

MASTER THESIS
COGNITIVE ARTIFICIAL INTELLIGENCE

**Adaptive Decision Support
Systems using Cognitive Models
of Trust**

Deny Raatgever (3154106)
denyraatgever@gmail.com
February-November 2011 (30 ECTS)
Version 4.0 Final (November 13, 2011)

External Supervisor:

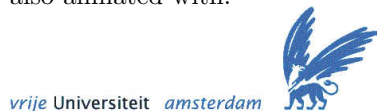
dr. P.-P. van Maanen
TNO Defensie en Veiligheid
Kampweg 5
Postbus 23
3769 ZG Soesterberg

Internal Supervisor:

prof. dr. J.-J. Ch. Meyer
Universiteit Utrecht
Department of Information and
Computing Sciences
Princetonplein 5
3584 CC Utrecht



also affiliated with:



Universiteit Utrecht

Third Reviewer:

prof. dr. V. van Oostrom
Universiteit Utrecht

Abstract

Decision support systems are becoming more important and more frequently used in every day life. These systems are also getting more complex and are providing more information at the same time. It is therefore more difficult for users to trust these systems appropriately, when working with the decision support systems. An improvement of these so-called human-computer teams could be to make the system more aware of how it is being trusted and whether this is appropriate or not. It could then intervene when this is considered to be necessary. In this thesis a way of estimating trust is described and possible interventions based on this estimation are discussed. The interventions are an adaptation of the provided decision support: 1) advice argumentation and 2) advice censorship. The trust models and two types of adaptive support strategies were implemented for a classification task. A pilot experiment showed that the trust models predicted trust well, but no significant differences in team performance have been found of the adaptive support types compared to static support. The different possible reasons for this are discussed.

Contents

1	Introduction	3
1.1	Research Question	5
1.2	Thesis Overview	6
2	Background	7
2.1	Trust	7
2.1.1	Social Trust Theory	7
2.1.2	Towards Strong Delegation	9
2.2	Decision Support Systems	9
2.2.1	Decision Support System Requirements	10
2.2.2	Enhancing Decision Support Systems	11
2.2.3	Trust-Dependent Task Environments	12
3	Trust Models	14
3.1	Descriptive Model	15
3.2	Prescriptive Model	17
3.3	Appropriateness Model	18
4	Adaptive Support	20
4.1	Argumentation Strategy	22
4.2	Censorship Strategy	24
5	Hypotheses	26
5.1	Model Validation	26
5.2	Support Evaluation	27
6	Method	28
6.1	Task Description	28
6.2	Task Support	31
6.2.1	Calculation of Threat Values (Ground Truth)	32
6.2.2	Addition of Noise (System Threat Values)	34
6.3	Trust Models Description for the Task Environment	36
6.3.1	Prescriptive Model	36
6.3.2	Descriptive Model	39
6.3.3	Appropriateness Model	42
6.4	Participants	43
6.5	Design	43
6.6	Independent Variables	43
6.6.1	Static Support	44
6.6.2	Argumentation Strategy Support	44
6.6.3	Censorship Strategy Support	46
6.7	Dependent Variables	47
6.7.1	Performance	47
6.7.2	Human Trust Feedback	50
6.7.3	Trust Appropriateness Ratio	50
6.7.4	Questionnaires	50
6.8	Task Implementation Issues	50
6.8.1	Server and Task	51

6.8.2	Noise Function	51
6.8.3	Argument Strategy Implementation	54
6.8.4	Censorship Implementation	55
6.9	Procedure	55
6.10	Materials	55
6.11	Data Analyzes	55
6.11.1	Model Validation	56
6.11.2	Support Evaluation	58
7	Results	60
7.1	Model Validation	60
7.1.1	Descriptive Model (In-Task)	60
7.1.2	Descriptive Model (Out-Task)	61
7.1.3	Prescriptive Model	62
7.1.4	Prescriptive Model per Scenario Part	63
7.1.5	Appropriateness Value	64
7.1.6	Appropriateness State	65
7.2	Support Evaluation	67
7.2.1	Performance	67
7.2.2	Trust Appropriateness Behavior	68
7.2.3	Trust Appropriateness Experiences	70
7.2.4	Results Feedback Participants	71
8	Conclusion	72
9	Discussion	73
9.1	Model Validation	73
9.2	Support Evaluation	75
9.3	Support Preferences on Different Task Characteristics	75
10	Future Research	78
	List of Abbreviations	79
	Appendices	81
A	Questionnaires (Dutch)	81
B	Task Description (Dutch)	88
C	Written Tests (Dutch)	96

1 Introduction

Computer systems are being used more extensively to support humans in various domains. Automation used to be just a mere representation of certain inputs to a human, while more recently developed systems are more intelligent and able to reason with input data. This change comes with many consequences; situations which used to be run by human agent teams only, are being replaced by human-computer teams. These teams need to manage working together in an appropriate way. However, the relation between humans is completely different from the relation between humans and machines. An important difference is the way people trust each other compared to how they trust machines. Human trust in other humans is a very natural process, people can negotiate with each other in order to gain trust and come to an optimal decision. With human-computer teams this relation is not very intuitive, since the only negotiation partner is the human. The human trust in the machine is lacking due to the absence of this negotiation possibility (for example the machine convincing the human of some advice).

An example is the usage of a navigation system. Many people are familiar with such systems and use it daily to support them finding their way. The user can have several problems with this support systems that are somehow related to trust. A familiar problem is for example that a user has no idea about the route himself. He sets his navigation system to his destination, and follows its instructions without looking for clues at the road itself (like signs, detours etc.). This leads to problems when the navigation system is wrong. Despite other information people might still only rely on their navigation system. In this specific case the user has overtrust in the navigation system.

When the user finally discovers the support was wrong, he could think the navigation system was badly programmed, and then stop using the advice from this system. In this case he could choose not to rely on the navigation system at all, while the system gives valuable advice. In this case, the user has undertrust in the navigation system.

The previous two cases could have been avoided (or at least improved) if the user had a more appropriate trust relation with the advice system. A way of improving this problem could be by altering the navigation system to give the user a better insight in its reasoning process. The user is able to have a better understanding why the system was wrong in the first place, making the decision to rely or not easier to make for the user.

Since the distribution of the tasks of human-computer teams has changed, the way we interact and look at computers has changed too. Humans are often excluded from the process of analyzing raw input, and left with high order representations of the environment. For example in naval contact threat classification, where the support system could advise the human to send a helicopter to a specific ship because it looks threatening. Sometimes the machine has completely taken over tasks, leaving the human in a mere monitoring position. An example of this type of automation can be found in the aviation field, where an auto pilot can completely take over control from the pilot, needing the pilot only to monitor the situation in case something goes wrong. When a human works with intelligent systems, his trust in these systems could play a significant role. It is impossible to know exactly how the system reasons and if the system comes to the right conclusion (output). Therefore the question whether to rely

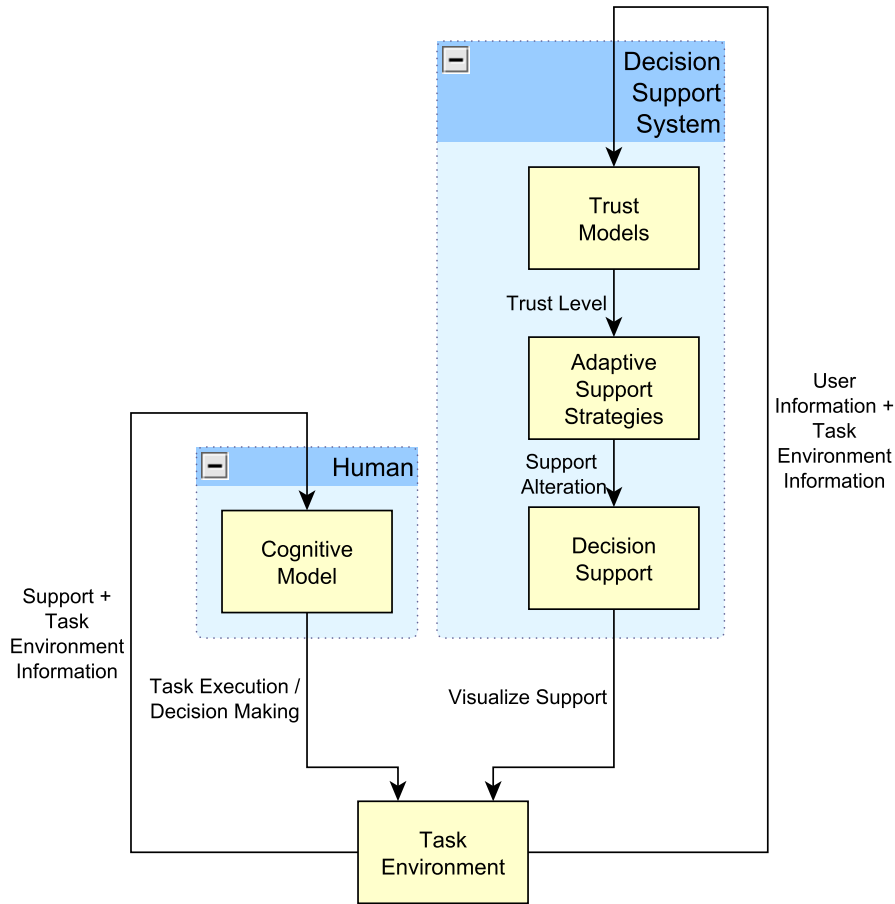


Figure 1: Adaptive Human-Computer Structure. Nodes represent the different objects used in the team structure. The edges represent the information which is transferred between the different objects.

or not rely on the system becomes more relevant. The human could for instance choose to neglect useful advice from the system because he does not trust it, while the system is correct (undertrust leads to disuse). He could also trust a system while it is giving wrong support (overtrust leads to misuse). These are trust-related errors which might have been prevented when the human has a better understanding of the system, and is able to have a more accurate trust in the system. One way to deal with this problem is trying to influence human trust to an optimal level, this could be the key to improving performance of human computer teams [11].

Several elements are needed to be implemented to enable an automated system to influence trust levels of humans. One crucial element is a possibility for the system to measure how it is being trusted by other agents. In this thesis a method will be explored of measuring trust that a user has in a system and

how appropriate that trust is [13]. Knowing how appropriate trust in a system is, a system could react to differences in trust appropriateness and have some kind of intervention. Interventions could differ from a change in the information representation, a representation of the trust appropriateness level to a complete task allocation to itself or other agents. One common example where such a system are being used is in teams that consist of one human who is performing a task, where he is supported by one system (people driving cars, flying airplanes, using computers, using mobile phones etc.). The goal is designing an advice system that is able to keep human trust into account. A basic trust based advice system could be designed like the model in figure 1. The structure in this figure consists of a human-computer team where the human does the actual task execution while being supported by a system. In this setting the human is still in control of the final decision making process and he can choose to take advice from the support system into account. However, he can also choose to leave the advice out of his decision making process.

Present are the elements which are needed in order to give advice to a user with the consideration of the user's trust in the system. One requirement (as given above) is a way for the system to measure human trust in the system. In the model structure this is done by different *Trust Models*. These models get input from the user himself (for example decisions he made) and from the task environment.

The ultimate goal is not to only measure the user's trust in the system, but to somehow use this information to give better advice. There are different ways of improving the system's advice. In the model structure this is referred to as the *Adaptive Support Strategies*.

In the end an enriched form of advice will be given to the user, with the consideration of his trust in the system. This could enable the user to get more appropriate advice, and have a better consideration of relying on the system.

This research is especially relevant for the field of Cognitive Artificial Intelligence (CAI). On the one hand a goal of the thesis is to find a way to model cognitive processes of humans (trust). On the other hand this thesis is concerned with using this cognitive model of trust in an artificial system. This could enrich the system with trust information, making it more intelligent. Therefore it is quite clear why this research is relevant for the field of CAI.

1.1 Research Question

This thesis is focused on human task execution, supported by a trust-based adaptive support system. This system will not perform always correct (as in real life), meaning it will give the best support it can using all the available resources. The system will collect evidence to estimate the appropriateness of the human trust in the system. The system is then able to adaptively change its advice strategy according to the human's trust. The question is whether the addition of adaptive trust-based support improves the performance of the team, compared to providing only static advice. The main research question is:

Can team performance of a human-computer team be improved by changing the system's advice strategies adaptively, based on trust appropriateness of the human in the support system?

To answer this question, an experiment will be performed, where subjects will perform a task while being advised by a support system. They can either follow the systems advice and answer accordingly, or not follow the system and come to their own conclusion. The support system will be able to adaptively change its advice (or the advice representation) by taking trust into account. The environment of the task is controlled, thus a performance measure can be constructed to test the performance of the human computer team. An important factor is that the support system is also not always right, and also not always knows when it is giving correct or incorrect advice. So the question is whether giving adaptive support based on noisy inputs will still lead to a better performance of human-computer teams.

While there could be many advantages to using such systems (such as more transparency of the system to the user), there could also be disadvantages: during adaptive support, more information could be presented (i.e. trust appropriateness information), what could lead to a higher cognitive load of the user. Other disadvantages could be when the system is adaptive, it could lead to a more complex support system and could be more difficult for a user to understand (since it is giving different support in different trust situations). This could also turn into an advantage, because the support system could adaptively give additional information of its reasoning at certain time points making its reasoning more transparent. The question is if the advantages of using adaptive support will overrule the possible disadvantages, and ultimately, whether trust-based adaptively changing the advice can improve the system's performance, keeping aspects like user's contentment and cognitive load into account.

1.2 Thesis Overview

In the next section the background of this study will be given. The background includes an explanation of the concept trust, what types of decision support systems might be subject to trust and what types of enhancements can be made to such systems, taking trust into account.

Furthermore all separate elements of the decision support system will be explained (from figure 1) in the sections after the background. The trust models will be explained based on existing literature and different kinds of adaptive advice strategies will be described.

After generally describing the different types of possible adaptive support, the support is taken to the test. In the method section will be explained how the models that were introduced were applied to a task environment. A structure will be given where all necessary elements of this human-computer team are further elaborated. The adaptive advice is used in the task environment, a description will be given how this is implemented. Furthermore the remaining experiment setup is given.

Following the method are the results, conclusion and discussion of the experiment and the research question will be discussed. The thesis will be concluded with a future research section, where the relevance of this research will be discussed, along with many possible and interesting future research directions.

2 Background

In the background section, first the concept *trust* will be described. Then the domain of decision support systems will be explained, with possible enhancements of such systems where trust can be influential.

2.1 Trust

Before being able to reason with trust, one first has to define the concept trust itself. Since trust is such a high level concept in human cognition, trust is not easy to define. Often the distinction is made between social trust and non-social trust, where non-social trust is about trusting mere objects. It is for example possible to trust a chair to not break when you want you sit on it. These are non-cognitive and autonomous agents (in the example the chair is the agent which is being trusted, the agent is autonomous because it is not able to reason with data). Social trust is trusting agents which are not completely autonomous [2]. In this paper the focus will be on social trust models, especially trust in decision support systems which are not completely autonomous. Human trust relations can often be compared with human to computer trust relations. Social trust can be explained in several ways. Coleman gave a reasonable description of trust [4]:

Trust involves voluntarily putting resources in the hands of another, who in turn will use them to his own advantage, the trustor's advantage, or both, which only works if the trustee is trustworthy.

This description contains some of the key features of trust, like trusting can be seen as a relation between two agents. Another issue dealing with trust is the vulnerability of the trustor when trusting another agent. Furthermore to determine the trustworthiness of the trustee, the trustor needs information of the trustee. Trust relations are based on certain goals the trustor wants to reach, and the agent can reach his goals by depending on another agent (the trustee). To grasp all these elements in one definition, a more detailed definition is needed, which can be found in a paper from Lee et al. [7]:

Trust is the attitude that an agent will help achieve an individual's goal in a situation characterized by uncertainty and vulnerability

However, a more profound description is needed to *reason* with the trust. When does trust apply? Which beliefs and goals are required for trust? In section 2.1.1 a more practical description of social trust will be given.

2.1.1 Social Trust Theory

To get a better hold of the concept, social trust can be defined as a relation between two agents, a trustor and a trustee. This relation can be divided in different concepts. The following concepts are broadly accepted [3, 2, 7]. Trust can be seen as:

- a **mental attitude (disposition)**, which is a prediction and evaluation of the trustor to another agent (the trustee). One can say another agent looks trustworthy and he may trust the agent when necessary;

- **decision to trust (intention)**, an agent intends to trust another agent. He decides the other agent is useful for fulfilling his goals and he intends to use the agent to do this. Deciding to trust another agent makes the agent vulnerable, because he puts responsibility in another agents hands, leaving him with uncertainty;
- an **act of trusting (behavior)**, delegating certain actions to another agent, in order to fulfill his goal. In this state, the agent relies the trustee to execute his actions in order to fulfill his goals.

These different types of trust all refer to the relation between the trustor and the trustee, but the trustor has to be in a specific mental state in order to trust the trustee. Castelfranchi et al. formalized which beliefs and goals are necessary for trusting another agent in one of the three ways [3]. This can be useful when designing a system which is able to reason with trust.

The most important beliefs an agent needs for trust disposition are competence belief and willingness belief. Competence belief is a profound evaluation of the abilities of the trustee, which must result in the belief that the trustee is able to reach the goals of the trustor. In fact, to have trust as a mental attitude towards another agent, he must not only believe the opposite agent is *able* to execute the right actions for fulfilling his goals. The trustee must also be *willing* to execute the specific actions, which is the willingness belief.

The decision to trust is logically stronger than trust disposition, which means that trust disposition is necessarily a precondition for the decision to trust, needing at least similar beliefs before an agent can decide to trust. One more constraint is the dependence belief. In order for an agent to decide to trust another agent, he needs to believe he has to depend on the other agent. Depending on the agent is the only way to reach his goals. There are two different types of dependence beliefs. In some situations, an agent believes he is *better off* deciding to trust another agent (weak dependence), or he believes trusting another agent is *the only way* of fulfilling his goals (strong dependence). Either of these dependence relations is necessary for an agent in order to decide to trust another agent.

When an agent decides to trust another agent, it can perform the act of trusting, by delegating certain actions to the trustee. There are three forms of delegation an agent can perform to achieve his goals. Given a model where there are two agents (X and Y), where there is a trust relation (X Trusts Y), three different forms of delegation is possible:

- **Weak delegation** is when an agent X trusts another agent Y to perform certain actions so that X can achieve his goals. However in weak delegation Y is not aware that X trusts him to perform certain actions.
- **Mild delegation**, in this form Y is aware of X trusting and delegating him. However, Y does not know the intentions and goals agent X has, so X is able to influence the actions of Y , without revealing what he exactly intends.
- **Strong delegation**, is when Y is not only aware of X trusting him, but he also knows what the goals of X are and he knows he is being exploited by X to fulfill the goals of X .

Since trust and delegation (reliance) are strongly correlated, it is important to notice that they are not similar. Trust can be seen as a mental attitude,

where delegation contains necessarily an action. Also an agent can have trust in another agent without delegation and vice versa, because trust is graduated and delegation is not. One can have a certain degree of trust in someone else, but could decide not to delegate to him, because the trust level is too low (below a certain threshold). On the other hand, one can delegate an agent without trusting him for several reasons, for example there is no other agent to delegate (including himself), so he does not have another choice but delegating to the opposite agent.

2.1.2 Towards Strong Delegation

The social trust theory is general applicable for various types of agents, for example when the two agents are human. However interesting is the relation between a human (trustor) and a machine (trustee). The main difference of this combination is in delegation. When the human has decided to trust the machine, and goes to the act of trusting, certain problems occur. As mentioned before, there are different types of delegation. Normally this relation is always referred to as weak delegation, where the human delegates certain actions to a machine (for example he trusts the machine to represent the right information), but the system is totally unaware of this delegation. The machine is often just programmed to represent information or give advice statically, based on programmed inputs.

To improve the performance of a human- computer team, this weak delegation relation could be changed to strong delegation, by giving the machine an awareness of his trustor. When this is achieved, the machine is able to get some indication of the human trusting him, and can anticipate on changes in this trust relation.

In order to achieve strong delegation from human to machine, the machine should be able to get an indication of trust and be able to reason with this trust levels. In the literature there are several models which are built to solve this problem. In section 3 tools will be proposed in the form of different trust models.

The question remains what exactly a system can do (types of interventions), based on his trust awareness.

2.2 Decision Support Systems

The domain of Decision Support Systems (DSSs) is a vast domain of different systems, from simple low level aid systems where sensory inputs are represented to a user, to sophisticated task allocation systems which not only represent information, but are able to reason with this information and can even choose to take over control, overruling the human [1].

The extremes on this automation scale have their own problems. On the one end a system which only represents information to the user is not able to reason with the data it represents. This results in a simple system where trust does not play an important role (the user either trusts or distrusts the system and chooses to use or discard the system, where the system cannot anticipate on this behavior). When systems are getting more complex, the question of how it is being trusted becomes more relevant.

However, the other extreme is also undesirable in risky situations. When a system is not only able to reason with information, but can also take over control of the situation (by prioritizing the system's decisions over the human's) could cause many problems. In most situations, the user wants to take the final decision and does not like a support system which takes over control when it concluded it was best. Even when the human is merely in a monitoring role, he wants to be able to intervene whenever necessary, leaving the final decision to him (for instance an autopilot on a plane, the pilot wants to be able to take back control when he wants). Machines taking over control also brings ethical objections, for example concerning responsibility. Normally the user is responsible for the decisions he makes, but is he for example still responsible when the system decides to take over control? Can a computer ever be responsible for its decisions? People have struggled to answer this questions for quite some time and these questions remain relevant in the search of adaptive systems.

In this paper, a DSS will be evaluated which is somewhere in the middle between the previous extremes, a system which is able to reason with information (instead of just representing information), with a human who has final control of the decision making process.

2.2.1 Decision Support System Requirements

A DSS can be used in many different ways. The relation between the human and the DSS can be compared to a negotiation process [12]. Negotiation processes are interesting for this study, because trust plays an important role during negotiation processes.

Any advice a system gives to the human can be seen as a type of negotiation, where the human has certain goals and beliefs, the system has certain goals and beliefs (generated by reasoning with input information), and they both want to make the best decision (according their calculations). When their solutions are similar they only support each other, but if they come to different conclusions there needs to be a negotiation process between the two. In this process the system needs to convince the human why it has chosen the advice.

With the shift of roles between the human and the support system, from the system not only representing input data to the user, to the system being more complex and able to reason with the input data, the difficulty for the human understanding and using the systems information increases. Appropriately trusting its information and use the system's support could get more complex for the user. Therefore some kind of negotiation is needed. The system does not merely give advice, but is able to convince the user by for example giving him more information about the reasoning steps [12]. What a system actually needs to do is to get the user to trust its reasoning and therefore its conclusions (the advice). When the user trusts the system he will consider its advice. There are several purposes of trust during the negotiation process:

- Determine what offer to send next. This can be translated to decision support as the question what information to show next.
- Select negotiation partners on the basis of trust. Which can be used to determine whether the human is still using the systems advice or not.
- Simplify negotiation dialogues. When the trust level is very high, the system might leave out much information about the advice (as long as the

user uses that advice appropriately in his decision making). However when the user has too little trust in the system, it might be the case that the system displays more information for the user to get a better understanding of the advice. This could in turn increase the trust of the user in the system.

2.2.2 Enhancing Decision Support Systems

Many support systems are not adaptive in their information representation. When advice is given, it is always represented in a similar way, with too little information or with too much information (even irrelevant information). Often the whole reasoning process of a DSS is like a "black box" to the user, where it is difficult for the user to keep trusting the system when the system gives complex advice. To deal with this problem, enhancements to the support system can be made on multiple levels [10].

The first level are content-based enhancements, which include choosing to show more or less relevant contents on the basis of the reasoning steps of the advice. This could include information of why the advice is given and how the system came to the particular conclusion. Content based enhancements can be achieved by presenting different explanations to the user [14, 10], leaving out information or giving extra information of the reasoning process.

First type of enhancements are adding rule traces. These traces are answers on questions like "why not?" and "what if?", which need to be displayed at the right level of detail. To quantify the importance and probability of different traces, certainty factors can be added to indicate how strongly the premisses have been confirmed. Secondly information about strategic knowledge could be shared, where the system displays its underlying problem-solving strategies. One content-based level deeper are the deep justifications. This type of enhancements aims at enriching the information with relations to other domains (for example X must be true because Y is relevant and Y holds in this environment), and how this can be used as heuristics to come to the given conclusion.

Furthermore the interface can be adjusted to enhance the DSS. This can be done in many ways, for example adding the ability to recover chosen settings or give immediate feedback on the actions the user takes.

Lastly the advisory strategy can be adjusted. Should the system adjust its advice to different users with for example different experience levels? How should the amount of content be adjusted in different situations? These questions could be related to trust, where different information is presented when the degree of trust in the system is changed. Furthermore the advice strategy contains information about the timing of the advice, who invoked them (the user, or triggered by the system), the amount of advice, the advice is safe to use, the advice might lead to harmful consequences, etc.

Need for adaptive information representation is based on previous studies, where users do not always like the extra information which is given to them, or even completely ignore the information because there is too much [14].

Arguments can be used to enhance the advice that the DSS is presenting to the user. Basic arguments can be divided into several components. A well formed argument does for example not only consist of a claim, but is also backed up by other information. An argument can be divided into the following components according to Toulmin's model of argument [14] (figure 2): an argument consists of a *claim*, which is shaped by a *qualifier* which is based on *data*. The

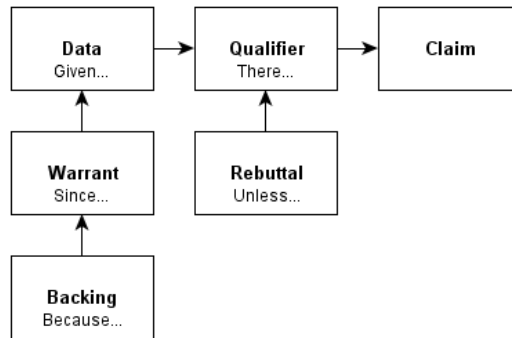


Figure 2: Toulmin’s model of argument, represented by Ye et al. [14].

reasonableness of the data is checked by a *warrant*, which is again substantiated by a *backing*. Furthermore a qualifier can be contradicted by a *possible rebuttal*, which is a kind of safety valve, where situations can be excluded (the claim does not hold for these situations, but it does for all other situations) for the validity of the claim.

2.2.3 Trust-Dependent Task Environments

The domain of the human-computer teams is very large, where in certain cases trust is a more relevant factor than other ones. An objective is to define environments where trust is of significance. Below several factors are given. These are restrictions to the tasks in the vast domain of human-computer teams, which consider tasks which are more dependable on trust.

A factor for example could be the complexity. Complex support systems could be more sensitive for human trust than basic systems, because the human has less knowledge of the reasoning process of the system. This can be compared to human trust in another human agent, giving complex or simple advice. The complex advice will be more difficult to understand and could be more dependable of trust.

Another factor in the evaluation of such a system is performance. If the system always performs much better or worse than the human, the reliance decisions could be easier to make (always rely or always not rely). Only in situations where the reliability of the advice is not always clear to the user and the user also doubts his own conclusions, trust plays an important role.

An implication of the system making mistakes is that the system does not always *know* if its support is correct or incorrect. This could influence the determination whether the system should be trusted or not. Therefore support which is too intrusive (for example task allocation, where the system could decide to take over control) will be risky and could lead to more errors. The adaptive support which is relevant only influences the advice. This means whenever the support system adaptively changes its advice, the user could always choose to follow or neglect it. So whatever the advice is, the user always has final control of the task execution.

The domain of relevant human-computer team environments have been nar-

rowed down by the given factors. In the remaining environments, trust could play a significant role. These environments could be useful to test the main research question.

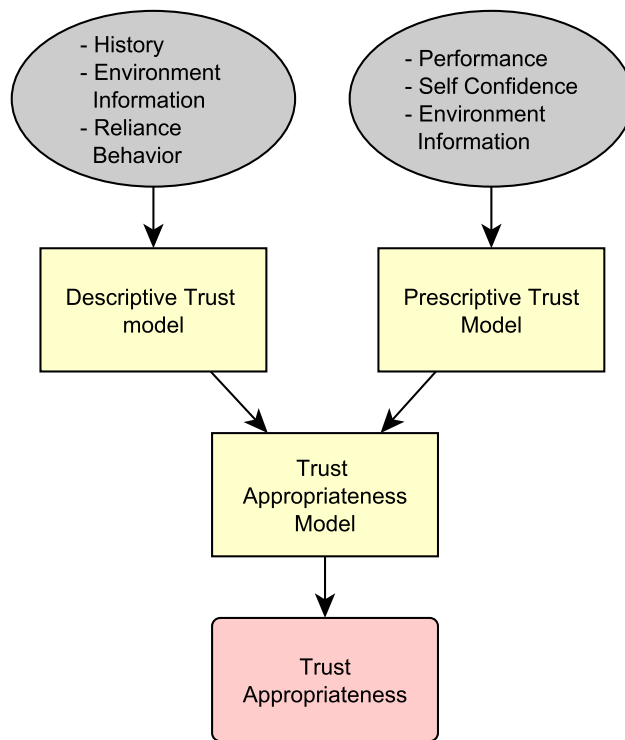


Figure 3: Trust model structure. Inputs are grey ellipses, models are yellow rectangles and the model output is a red rounded rectangle.

3 Trust Models

Trust can be seen in many forms, like trust as mental attitude, trust as intention or trust as a behavior (section 2.1.1). When one needs to make a decision, people are often influenced by many other agents which all have different views on the subject. In this position people have to make a decision who to rely on (this could of course also be on their selves). This reliance decision is based on the trustworthiness which the agent estimates of the other agents. To make the decision on who to trust, the agent is more probable to choose the agent who is most trustworthy, and the agent which he trust least will be chosen with the lowest probability.

In this context, trust can be referred to as a behavior, it can be observed or asked to the trustor who he has been or will be trusting in the future. This information is in human situations quite intuitive, where people try to indicate how well they are being trusted by the other humans, and reacting on this information when necessary. For a machine gaining this information (evidence) is more difficult, while it could be beneficial when a machine could use this information. To get information of how the system is being trusted, certain models are needed which can estimate this trust information. Below trust models are proposed that are capable of calculating actual trust estimates based in the information which is available for a machine.

The trust model is used to estimate whether the system is trusted and used in an appropriate way. For this estimation different factors are needed. First the system needs an indication of its own trustworthiness (in an optimal situation). This is what the trust level of an infallible agent would be in the system. The trustworthiness is called prescriptive trust, so what the trust level in the system *should* be.

The second factor which is needed is the trust a trustor actually has in the system. This could be referred to as descriptive trust, or what the trust level *is*. When a system is able to calculate both factors, it is able to compare them, leading to a trust appropriateness level (see figure 3). This structure was introduced and used by van Maanen et al. [13], and will be expanded to work on different inputs.

When for example the prescriptive trust is high (the system should be trusted) and the descriptive trust is very low (the system is not trusted), trust is not appropriate. This would result in misuse of the system, namely it is not used (because it is not trusted) in situations where it should be used. Another situation could be when prescriptive trust is low (system should not be trusted) and the descriptive trust is high (system is trusted by the trustor), leading to disuse of the system. The optimal way of trust appropriateness is when the level of prescriptive trust is equal to the descriptive trust. When for example both levels are low, the system should not be trusted and is not trusted by the trustor. When both levels are high, the system should be trusted and is trusted by the trustor. In these situations trust is appropriate, which is desired.

When the system detects a difference in prescriptive and descriptive trust, a reaction from the system could be helpful. It can for example change its support strategy, trying to decrease the distance between the prescriptive and descriptive trust, this method is called trust calibration. Another reaction could be the allocation of tasks to other agents, or taking over control from the trustor [13, p. 103-115].

3.1 Descriptive Model

The descriptive model is an indication of how the agent is being trusted by the other agents. The estimation can only be made with the internal model of the opposite agents trust (for example by asking the agent each decision who he trusted).

The ideal situation would be the system knowing exactly how it is being trusted by the other agents. This information is however in reality only partly available, so information about this trust is needed. Sometimes this information is available looking at the past. It can check whether the other agents has been relying on him in the past. When the agent (trustor) made similar decisions to the system, it could sometimes be concluded the trustor has relied on the system. Note that this is also an indication (not a certainty). Since the system does not know the motivations of the trustor, it could have come to similar conclusions without considering the system instead of following the systems advice. Asking the agent for every decision he makes whom he trusted is in real life not feasible. Because of the lack of a complete world knowledge, the agent cannot calculate descriptive trust, but has to estimate this, based on information which is available to him (which is also how humans estimate descriptive trust of other humans). In descriptive trust estimation, reliance data can be very

useful. It cannot conclude the trustor certainly relied on the system, but it can conclude that if the user is following the support system consistently, the probability is higher that the system really is being relied on. User information could be used in combination with the environmental data and possibly other inputs to give an indication of descriptive trust (figure 1).

Cues of the past reliance in combination with environmental data for instance could be useful in estimating future reliance with the knowledge of the environment. When someone relies heavily on a support system, the reliance was advantageous and the environment stays constant, he is more probable to rely in the system in the future. Therefore the history, environmental data and reliance information could be used in estimating reliance in an agent [6]. On the other hand, when the experience an agent has in another agent is negative, he is bound to rely less on this agent in the future. This could be used in estimating descriptive trust.

Different models based on experiences have been constructed, for example independent trust and relative trust models. These models are both compared in experiments by Hoogendoorn et al. [13, p. 85-99], who concluded relative trust models estimates descriptive trust more accurately than independent trust. This could be because the relative trust model is a richer and more complex model than independent trust. Independent trust just estimates current trust levels in a specific agent, without looking at the other agents (so the other possible reliance decisions). Relative trust on the other hand not only describes trust in one agent at a specific time, but compares this trust level with trust levels of all other agents. So if an agent has low trust in an agent, but even lower trust in himself, he would probably decides to rely on the agent even when its trust in this agent is low.

However, in some environments only two agents are available. Human computer teams for example often exist only of a human, being supported by a single support system. In this case the human only has two reliance options, rely on the support system or on himself. When the descriptive trust in the support system is high, implies that trust in himself is low (since the binary choice of reliance), and vice versa. This means looking at the other agents in this environment is not necessary so the relative trust model is not necessary to get an accurate descriptive trust estimation. In this case the independent trust model would be sufficient. In this thesis the setup of a human-computer team will consist of only two agents, so the focus will be on the independent trust model.

Independent Trust Model

Below the independent trust model is given as described by van Maanen et al. in [13, p. 72-81] and in [13, p. 83-99].

Independent trust is a model to estimate trust levels of one agent (for example the human) in a possible trustee (for example a decision support system). Trust is influenced by current experiences the trustor has of the trustee. Also the past experiences have been influential. Experiences can consist of different factors, like agents performance history (high performance could lead to positive experience, especially when the trustor relied on the trustee when its performance was high). However, performance feedback is not always available to the agent. In such cases experiences could be defined as the trust behavior

of the agent. Experiences ($E_i(t)$) of agent i at time point t can range from -1 (meaning a negative experience) to $+1$ (meaning positive experience). In between is 0 , which is a neutral (or no) experience. Estimating trust with the use of experiences can be done using the following function:

$$T_i(t) = T_i(t-1) \cdot \lambda_i + \left(\frac{E_i(t) + 1}{2} \right) \cdot (1 - \lambda_i) \quad (1)$$

Note that experiences $E_i(t)$ lead to trust $T_i(t)$, so all current experiences combined with the past trust lead to an updated trust value in a current time point. The influence of past experiences are different for each agent, and can be varied by the decay factor (λ_i). This factor is defined as $\{\lambda \mid \lambda \in \mathbb{R} \text{ and } 0 \leq \lambda \leq 1\}$. A lambda value near 1 means current trust level is less influenced by past trust levels (the agent forgets past experiences easily and focusses on new experiences), and a value of 0 means the agent is not greatly influenced by new experiences, but keeps to the past trust level (see function 1).

The function is generally applicable, only the decay function parameter should be determined for the specific agent truster, and the experiences need to be specified for the environment the agent is in.

3.2 Prescriptive Model

The prescriptive trust model is an estimation of the level of trust a rational agent has in the trustee, meaning it is an estimation of the optimal trust level of the agent. This estimation can be calculated based on different measures, dependent on the information the agent has access to. One possible measure is performance, where feedback on the past performance can be used to estimate future performance [13]. When the performance of the last time steps were high, the estimation of future performance would be high and prescriptive model would result in a high rational trust level in the next time step. However performance estimation is not always available in real world tasks, where feedback is often not given, or cannot be mapped directly to the performance of the agent alone. In this research another approach to the prescriptive model will be used, which could lead to a more general trust model. This model is able to perform with less explicit information about the the environment in order to estimate trust. When performance information is not available, different information could be used to estimate an agents performance, like self trust (self confidence) and environmental data (model input in figure 3).

In some environments, self trust could be estimated by the agents own confidence in the decisions it it about to make. By using his internal model and environmental data, information of the agents reasoning is available to the agent itself, making it possible to estimate how confident he is in his own decisions (or decision support). If the agent is confident about his decisions, it would be rational that his self trust is high. This means the prescriptive trust in the agent is high (the rational agent will have high trust levels in the agent). When the confidence of his answers is low, the agent will have low self trust which will result in low prescriptive trust (he would not trust himself with the decisions he is about to make).

Self trust can be seen as an internal measure of performance, an estimation the agent calculates with the lack of explicit performance information. The agent is often good in self trust estimation, since it can use its internal (cognitive)

model. When explicit performance data is not available, self trust is one of the best measures of prescriptive trust. This could be compared to a human telling a possible trustor "I am sure of my conclusion, so trust me!", or "I might be wrong, please reconsider my advice before using it."

3.3 Appropriateness Model

The trust appropriateness model is based on both the output of the prescriptive model and the descriptive model (as can be seen in figure 3), the trust appropriateness model will compare the output of these two models, to result in a trust appropriateness level. When the natures of the prescriptive and descriptive models are considered, it becomes clear why this comparison is made, and why this leads to a trust appropriateness level. The prescriptive model indicates the ideal trust level in the trustee, while the descriptive model indicates the current trust in the trustee. When these two values differ more, the trust appropriateness level also comes more out of balance. When the prescriptive and descriptive model have more similar output, the trust in the trustee will be more appropriate. The equation below is constructed to make this comparison:

$$\alpha_i(t) = \tau_i^d(t) - \tau_i^p(t) \quad (2)$$

Meaning trust appropriateness α in agent i at time point t is calculated by the descriptive trust $\tau_i^d(t)$ and the prescriptive trust $\tau_i^p(t)$ of agent i at time point t . Both τ_i^d and τ_i^p have to be scaled in a similar way, which is usually achieved by using real numbers between 0 and 1: ($\tau_i^d, \tau_i^p \mid \tau_i^d, \tau_i^p \subseteq \mathbb{R}$ and $0 \leq \tau_i^d, \tau_i^p \leq 1$). Assuming the boundaries of the descriptive and prescriptive models, it also holds that α_i is a bounded real number: ($\alpha_i \mid \alpha_t \subseteq \mathbb{R}$ and $-1 \leq \alpha_i \leq 1$). The resulting α_i is the trust appropriateness of i , which is negative when the agent is expecting the trustee to make worse decisions than the trustee is actual capable of (undertrust) and positive when the agent thinks the trustee will perform better than the trustee will (overtrust) according to the rational agent.

The $a_i(t)$ value stands for a certain trust appropriateness of an agent in another. However, this value is a continuous value which needs to be interpreted somehow by the agent to be meaningful. Three different trust appropriateness states can be distinguished: undertrust (leading to disuse of the system), appropriate trust (leading to appropriate use), and overtrust (leading to misuse). These three different states are related to the outcome of the trust appropriateness model, where low values (near -1) lead to undertrust (low descriptive trust and high prescriptive trust), values near 1 lead to overtrust (high descriptive trust and low prescriptive trust), and values near 0 lead to appropriate trust (similar prescriptive and descriptive trust values). The scale of $a_i(t)$ must be discretized to be mapped on the three different trust states, this can be done by using two thresholds (th_1 and th_2):

$$-1 \leq \text{undertrust} \leq th_1 \leq \text{appropriate trust} \leq th_2 \leq \text{overtrust} \leq 1 \quad (3)$$

The optimal trust appropriateness level is when descriptive and prescriptive trust are equal resulting in a $\alpha_i(t)$ value of 0. Constraints on the th_1 and th_2 are needed to ensure an outcome of 0 will always lead to appropriate trust state, leading to: $th_1 \leq 0 \leq th_2$. Furthermore these thresholds can be a value between -1 and 1.

The thresholds can be varied, depending on how cautious the agent is on his under- and overtrust verdict. When the agent is very cautious, so rather conclude the agent trustor of under- or overtrusting him, the thresholds could be set closer to the 0 mark. However, in some situations his verdict of under- or overtrust could have great consequences, where the agent needs to have a higher certainty when concluding the trustor is under- or overtrusting him. In this case the values could be further from the 0 mark.

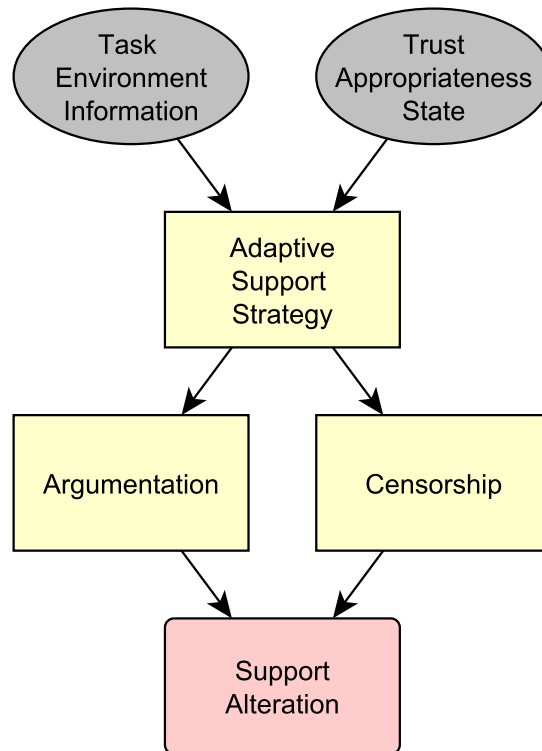


Figure 4: Adaptive Support Strategies, based on trust information. The inputs of the adaptive support model are given in grey ellipses, the model elements are yellow squares and the output of the adaptive support model is represented by a red rounded square.

4 Adaptive Support

The support strategy consists of different ways of altering the decision support. One type of modifications could for example concern the amount of advice, or the way the advice is given. Other modifications could be adding information about how the system calculated its advice. When information is not available the agent may not understand how the system works, the reasoning process is like a black box and it somehow came to a conclusion. This has been explained in sections 2.2.1 and 2.2.2. The opposite agent can only guess how the support system reasoned. If the agent does not know how the opposite agent came to a conclusion, it can be difficult to estimate the quality of the agents decision support and deem the trustee untrustworthy. This could lead to disuse of the system. On the other hand, a user mostly has to perform in restricted time, so he does not have enough time to process all information about the reasoning steps all the time. When he trusts the system, it means that he trusts the system to reason properly and come to a conclusion. The effect of giving too much information to the user can also lead to irritation, because the information which is represented is often already clear to the user. The system acts like the user does not know what he is doing. Giving too much information can also

lead to the discarding of valuable support from the decision support system (when there is too much information, the opposite agent can just ignore all information all together, while parts of this information could be valuable). The adaptive decision support system is constructed to prevent such a situation by representing the information in another way if the system is not properly trusted (disuse), or is trusted blindly (misuse). This way only the valuable information is presented to the user, so the decision making process will occur in optimal time.

The system's advice can be compared to a negotiation process between two agents (human or computer agents). The purpose of the negotiation is to give the right information (arguments or advice) at the right time, to convince the opposite agent of your reasoning (see section 2.2.1), without giving too much or irrelevant information. During the argumentation when an agent trusts a trustee, he does not want all the extra argumentation which is available, because he trusts the agent to reason correctly and come to the right conclusions (even without the extra information), meaning trust is appropriately calibrated. However when the agents does not trust his trustee, but the trustee thinks he is right, he will need to give extra arguments to prove his point and convince the agent to trust him. Similar reasoning could be used in a human agent using advice of a decision support system.

The argumentation of the trustee will have the right effect when it is trusted by the trustor in an appropriate way. Therefore, in real life, humans will adopt their argumentation structure according to the trust levels people have in them [14]. When trust levels are appropriate enough people will leave out information, when it is too low, people could give addition explanations to back up their argument and convince the negotiation partner. However, using the trust model from section 3, a decision support system can get an indication of how appropriate trust levels are in their argumentation (as can be seen in schema on figure 1 on page 4). This information could be used to change their advice strategy according to the trust states: undertrust, overtrust and appropriate trust (see section 3.3). The states are the input of the adaptive strategy model, which will be combined with the environmental data lead to different advice in different states, the structure can be found in figure 4.

Different types of support could be specified on the basis of this trust information. The system could for instance give direct information of current trust appropriateness levels, change the transparency of the support by adding more reasoning information or leaving out information for the trustor. Stronger support might be task allocation based on trust appropriateness levels. However, the trust appropriateness level might be inaccurate, since it is only an estimation of the real trust appropriateness (as explained in section 3). This leaves out strong interventions like task allocations considering the model might not be always correct. Two different types of support will be further explained: adaptive argumentation support and adaptive censorship support (see figure 4). Important is that the user must be able to do his task at all times, even when the support is not available or false. Therefore the system must be designed in a way where it only gives advice to the user. The interventions of the system (the adaptive strategies) should only change the advice strategy, which can be followed or not. This leaves the user in final control at all times, with the possibility to rely or discard the systems support.

4.1 Argumentation Strategy

As explained in section 2.2, a reason for an user to distrust a support system, could be due to the information the support system is giving to the user, or the possible lack of this information. This is also referred to as the transparency of the support that the system is giving to the user. When the system just represents the right answer, and the user does not know the reliability of the system (trust is not appropriately calibrated), he could find the information invaluable and discard it. This is a problem because the system might have evidence of the advice being correct, while the user discards its valuable advice. In such cases it could be really helpful that the support system is made more transparent. This system can be enhanced by for instance giving the user more information of how it came to the given advice (as explained in background section 2.2.2). With extra information the user might be able to use the support system more appropriately, by having more appropriate trust in the system's advice.

Therefore this type of support is called adaptive argumentation support. This support consists of changing the transparency of the support's reasoning process on different trust appropriate states. Changing the transparency can be done on various levels (see figure 2 from the Toulmin argumentation model), a claim (or advice) could be backed up by additional arguments, data which is consistent with the arguments and claim, backing of these data and further. This are all types of arguments the system could use, showing the user a more detailed advice than just the advice itself. This could give the user a better insight of the reasoning process of the support system, what could lead to a more substantiated decision to trust or distrust the system.

However, when the user already has knowledge of the reasoning process of the support system or he has knowledge of its reliability, a more transparent advice could be undesirable. First of all it could lead to an increase of the cognitive workload of the user, having more information to process before it can decide to trust or distrust the systems advice. Secondly sometimes the user might have very different reasons to make a different decision than the advice, which are unknown to the support system (or the support system might be wrong in some cases for different reasons). In this case the user does not want to be bothered by the support system, giving him additional information. An example of this effect is the navigation system. This system might not send you in the right direction because there are road works. You could therefore ignore the navigation support. The navigation system now repeats its advice: "Turn around". This could lead to irritation from the user, who could decide to disuse the system in future advices. So you want the right information when it is necessary, and no extra information when it is not needed.

Figure 5 is a description of the structure of the argumentation strategy. The adaptive strategy model receives the current appropriateness state from the trust models. For every possible appropriateness state, there is a different type of adaptation.

The first trust appropriateness state is undertrust, when the system detects it is not being used while it is confident of its advice, it changes its advice adaptively. More information should be given, getting the user to trust the system more when this is appropriate. The advice will be on full transparency. Since the system is certain its advice is right and it is not being used, arguments

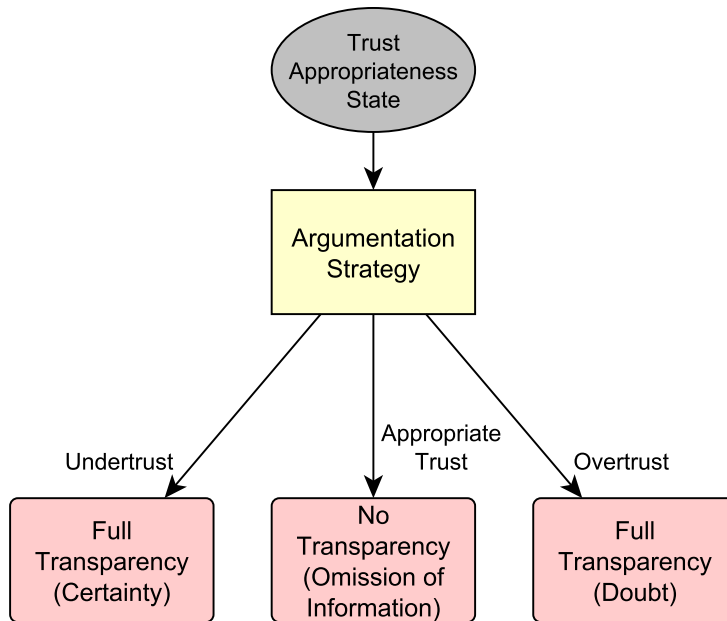


Figure 5: Argumentation strategy. Model is a yellow square, input is a grey ellipse and outputs are red rounded rectangles.

could be given backing the correctness of the advice. Why would the system be right, and based on what information would the system be right. These are questions the user might have and they should be answered by the support system.

On the other end of the scale is the trust appropriateness state overtrust. In this situation the support system is also not optimally trusted. While the system doubts its own advice, it is still being relied on by the user. In this case arguments are needed that the given advice might not be correct. In this state the support system needs to be fully transparent, with arguments why the user could doubt the correctness of the advice. Note that the transparency is similar to the undertrust state, but the arguments themselves are different (doubt instead of certainty arguments).

The optimal state is when the trust is appropriate. When the user is in this state systematically, the probability is high that the user has knowledge about the reasoning process of the support system, or at least he is able to estimate its reliability when considering the support. In this case too much information is not desired, because this could lead to communicating irrelevant information to the user (the user might already be aware of this information). In this appropriateness state the transparency should be limited, representing just the advice with no additional arguments.

The above given adaptive argumentation interventions (based on the different appropriateness states) are based on the way humans communicate to each other. As explained in section 2.2, when humans are negotiating, a sense of trust of importance, when one person tries to convince another and senses he is

not being trusted, he can give extra arguments to convince the other negotiator. When he is not certain of his solutions he could argue why he could be wrong and when he thinks he is being trusted appropriately, he could decide giving additional arguments will not improve his opponents opinion.

When the argumentation strategy is applied to the advice, the advice system is being more transparent to the user. The system gives a better insight of how the advice was calculated. The user can have a better consideration of the worthiness of the given advice. The decision to either trust or distrust the given advice is hereby made easier. The goal of the argumentation is to enable the user to trust the support system more appropriate, by steering the user's consideration into the right direction (using the different argument types). The user can reach more appropriate trust since he is aware of how he can trust the system in an optimal way (better calibrated trust). This means the user can get a better trust appropriateness *experience* by using the extra arguments.

4.2 Censorship Strategy

Where the argumentation strategy reasons about the advice itself, the censorship strategy is a more low level adaptive strategy: censorship does not work in addition of the advice, but it can change the advice itself. Recalling the background (section 2.2), a negotiation process consists part of the decision what to offer next to the negotiation partner. In the case of the human-computer team, this is referred to as what advice to offer to the user next. In some situations it could be beneficial to limit the amount of advice and in other situations it might be useful to give all the available advice. In human teams, trust is often correlated to censorship. To decide what offer to send next to the negotiation partner is often influenced by the trust the possible trustor has in the partner. When for example, I know that I am not being trusted, I can decide to leave out some arguments or claims, because I could be wrong and even lower the trust levels the partner has in me. Similarly it could be reasoned the situation involves a human-computer team. One goal of the adaptive advice is calibrating user's trust in the support system, where this could be influenced by censorship.

For all predefined trust appropriateness states, a type of censorship can be defined (figure 6). First state is undertrust, when this happens the user has little faith in the systems advice, while the advice reasoned its advice is reliable. In this case the system must have a strategy to regain user's trust in the system, so it would be trusted more than he does now. In a human team, a reasonable response would be modesty. Claims you are not really certain about will be left out, only claims with high certainty will be made. This is because you can imagine slightly doubtful claims will be rejected by the negotiation partner and only affect the trust he has in you more negative. Similar strategy could be applied in the support system. It can be modest in its advice by fully censoring advice with less reliability.

When the system detects overtrust, also a type of censorship can be used. Since overtrust is also not desired in the user's reliance on the support, something must be done to recalibrate trust. Overtrust indicates the user following the advice in situation where this is not correct. If advices are censored from the user, the possibility to overtrust is taken away from the user, forcing him to think for himself instead of letting him blindly follow the system's advice. This type of intervention is also imaginable in a human to human situation,

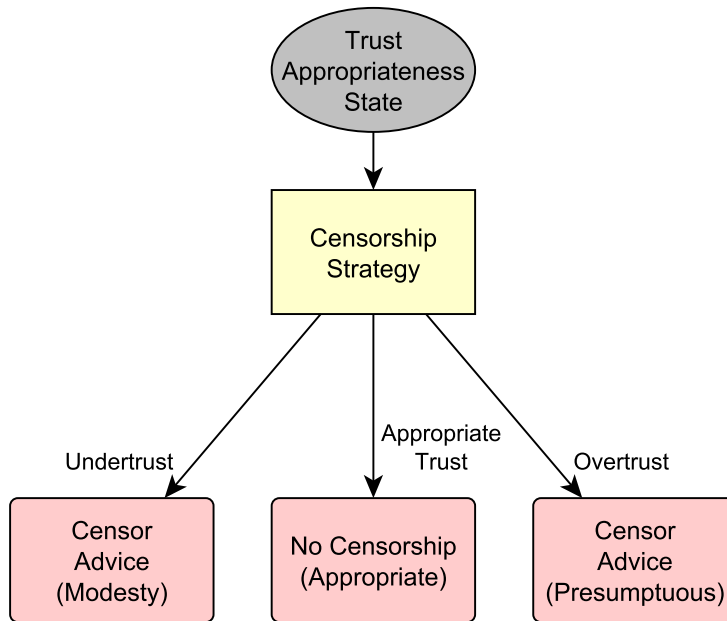


Figure 6: Censorship strategy. Model is a yellow square, input is grey a grey ellipse and outputs are red rounded rectangles.

when I know I am being trusted whatever I say, I can either claim nonsense (to let the partner find out and see for himself he is following nonsense), or leave out claims so the possibility of following them will be taken away. In the case of the support system, when the support system detects overtrust, it will get presumptuous and fully censor least obvious advice. When the least obvious advices are left out, it is not possible for the user to rely on possibly uncertain advices anymore. The user is forced to think for himself, hoping its trust in the system will be more optimally calibrated. Note that this the type of intervention during overtrust is similar to the intervention during undertrust, only the motivations for both interventions is different.

Finally there is a situation where the trust is appropriate, in this state the system detects the advice from the system is used in an appropriate way by the user, so it can just give all advice which is available. With no censorship applied to the advice and the user appropriately trusting the support system, the information can be optimally used.

Unlike the argumentation strategy, the user might not be aware that the system's support is being adaptively modified . Therefore the trust experience might not directly be influenced by the censorship strategy. Instead, this way of adaptive advice representation aims at influencing the trust *behavior* of the user. He might realize that the given advice is more reliable than he actually thinks (during undertrust) or the advice is less reliable than he imagined (during overtrust). This could lead to a better trust appropriateness of the user in the system, leading to a reliance behavior closer to the optimal reliance behavior. Optimal calibrated user's trust can lead to a better team performance.

5 Hypotheses

The main research question is "Can team performance of a human-computer team be improved by changing the system's advice strategies adaptively, based on trust appropriateness of the human in the support system?". This question can be answered using two different elements. The first element is whether a system can measure how it is being trusted by other agents, especially how it is being trusted by a human agent. The hypotheses concerning measuring and modeling trust can be found in the Model Validation section below.

The second element is whether the adaptive strategies (based on the measured trust appropriateness) have led to an improvement of performance. Hypotheses concerning team performance improvements of trust-based adaptive advice can be found in the Support Evaluation section.

5.1 Model Validation

The adaptive advice is based on several trust models. As explained in section 3, these models are estimations of human trust in the system (descriptive trust), self trust of the system (prescriptive trust) and the trust appropriateness.

The descriptive model is an indication of the levels of trust of the human in a support system, thus the output of the descriptive model should be correlated to the real trust levels of the human in the system.

Hypothesis 1. *The output of the descriptive model is a predictor for the human trust level in the support system.*

The prescriptive model is a model that outputs the self trust level of the support system. Self trust is an indication of how much the system is trusted by a rational agent, so what the optimal trust level in the system is. A measure of optimal trust can be system performance. When system performance is better, the system would be trusted more by the rational agent. If performance is worse, it would not be trusted by the rational agent. Therefore system performance and prescriptive trust could be correlated.

Hypothesis 2. *The output of the prescriptive model is a predictor for the performance of the support system.*

Trust appropriateness is measured by the difference in descriptive and prescriptive trust. The appropriateness model could be used to estimate human trust appropriateness in the system. The trust appropriateness value is used to determine the trust appropriateness state (recall section 3.3). Note that the trust appropriateness value is the difference between descriptive and prescriptive trust, and the trust appropriateness state is the related state. The possible states are undertrust, overtrust and appropriate trust. The trust appropriateness state is calculated to determine adaptive advice. The appropriateness state should be related to the real appropriateness state of the user (for example if he really has undertrust in the system's advice or not).

Hypothesis 3. *The appropriateness state of the trust models is a predictor for the real appropriateness state of the human in the support system.*

5.2 Support Evaluation

To answer the main research question, the actual team performance is important. As explained in the background (section 2.2.2), there are reasons (provided by other studies) to suggest that the addition of trust based adaptive advice could lead to a better team performance. Different types of adaptive advice can be used to optimize the user's trust in the support system's advice. As explained in section 4, the different types of adaptive support can optimize the trust appropriateness from the user in the system, by either influencing the trust appropriateness behavior of the user in the system's advice or the trust appropriateness experience. Better trust appropriateness can lead to performance improvements, when adaptive advice situations are compared to non adaptive situations.

The first possible adaptation that will be studied is censorship; advices that seem irrelevant to the user will be censored. This could lead to a better performance of the human-computer team. Therefore the following hypothesis is given:

Hypothesis 4. *The human-computer team will show an improved performance when the human is supported by censored advice as compared to static advice.*

Another possible adaptive strategy is argumentation. When argumentation is applied to the advice, the system could give additional arguments when it is trusted inappropriately. The human is able to make a better consideration of its decision possibilities, what could lead to a better performance.

Hypothesis 5. *The human-computer team will show an improved performance when the human is supported by argued advice as compared to static advice.*

6 Method

The previous given adaptive strategies will be applied on a task. The task should be chosen so that support is desirable and trust in that support is a factor (as explained in section 2.2.3).

The task that will be used is the Tactical Picture Compilation Task (TPC Task), which was first constructed by Heuvelink et al. [5]. The task was later adjusted and used by Lucassen [8, 9] and van Maanen et al. [13, p. 117-218] in several experiments. The task is executed by a human agent (the user), who is supported by a decision support system (advice system). The TPC Task is a naval classification task, which is dynamic, routine (support might be desirable in routine tasks, where the user can have concentration problems), difficult (so the participants are not likely to perform always correct) and a simplified version of a complex task which is done in real life. The simplification makes it possible to keep track of the exact performance of the human-computer team, since a form of absolute truth (ground truth) can be measured at all times.

In the introduction, a basic structure of a human-computer team was given, with the desired adaptive support structure (figure 1 on page 4). For the experiment, such a system will be implemented to work on the TPC Task. In figure 7 an expanded version of the model structure from the introduction is given, with a detailed plan of the elements needed to get such a system working. The *Trust Models* are elaborated (using the structures described in section 3) and the *Adaptive Support Strategies* are implemented using the structures from section 4. In this section these structures are implemented on the task environment.

As can be seen in figure 7, on the left hand side there is a human, who receives updated environment information and makes decisions based on this information. At the same time, there is a decision support system (on the right) advising the user (the *visualize support* edge in figure 7).

Support is calculated based on environment information, this results in static support. The system also determines human trust in its previous given advice. It can then modify the static task support and adaptively change this support using the adaptive support strategies. This leads to adaptive decision support, which is visualized to the user.

The purpose of the adaptive support is to provoke appropriate trust from the user in the system, so the human can make optimal decisions based on the support and the available information. In the coming sections first the task will be described, then all separate elements of the decision support system (in figure 7) will be further elaborated. This will be followed by the experiment setup.

6.1 Task Description

The TPC Task consists of a radar screen (figure 8), with in the middle your own ship (blue icon) and around your own ship other ships (contacts). Contacts are either friendly or hostile. However, there is no additional information of these contacts but the radar screen. The task of the user is to mark the *five most threatening contacts* using only the information on the radar screen. The total number of contacts on the screen is 24 during the experiment. There are different behavioral characteristics that determine how threatening a contact is,

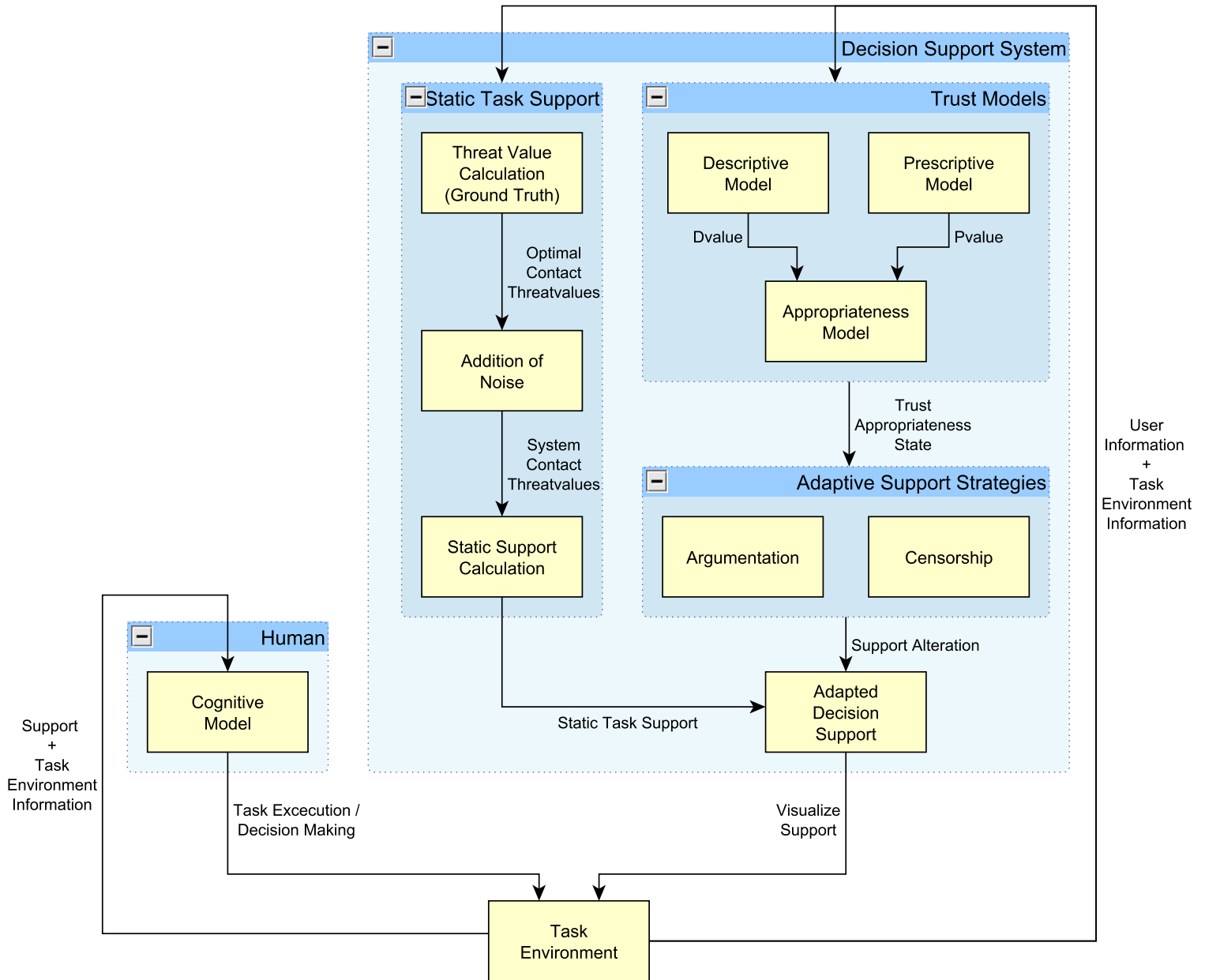


Figure 7: The structure of human computer team, as built in the experiment. Elaborated from the basic model from the introduction (figure 1 on page 4). Nodes represent the different objects which are needed for the team structure. The edges represent the information which is transferred between the different objects.

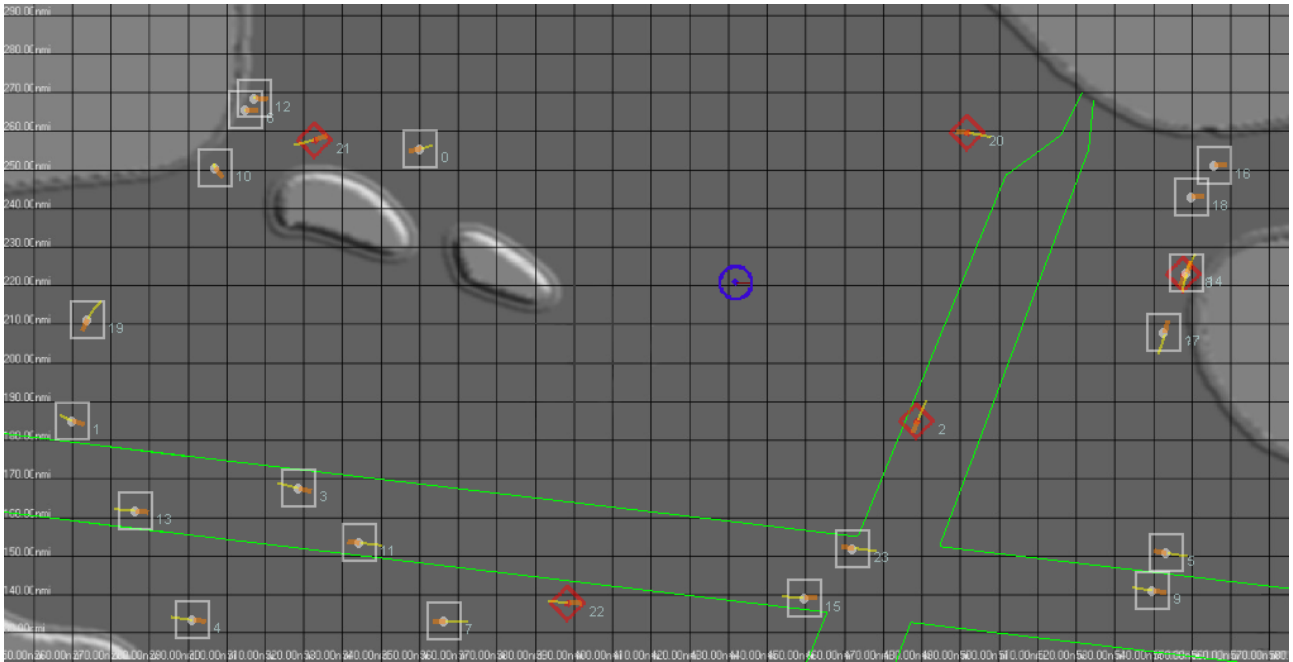


Figure 8: Screenshot of the task environment.

these characteristics are formulated in rules or identification criteria (idcrits). The four idcrits are:

- **Heading**, if the contact is moving in the direction of your ship (blue icon), it is more threatening than a contact which is moving in another direction.
- **Distance**, a contact which is closer to your own ship is more threatening than a contact further away.
- **Speed**, a faster moving contact is more threatening than a contact which is moving more slowly (it takes more time to reach your own ship)
- **Sea lane**, a contact which is moving in the sea lane (between the green lines) is more likely to be a normal ship thus less threatening. Contacts outside the sea lane are more probable to be a threat.

The idcrits are simplified rules of a real life situation to enable naive subjects to participate in the experiment with limited training. Combining these idcrits, one can estimate how threatening a contact is compared to other contacts. All idcrits are equally important.

In figure 9 different states of contacts are presented, where figure 9a represents your own ship. This will be stationary during the entire experiment. The other contacts will all be similarly represented, where an unmarked contact (status unknown or marked as not threatening) looks like the contact in figure 9b. When a contact is marked by the user as threatening it will turn into the icon given in figure 9c.

Each idcrit of a contact can be evaluated using only the radar screen. The combination of all idcrits can be used to determine which contacts are most threatening.

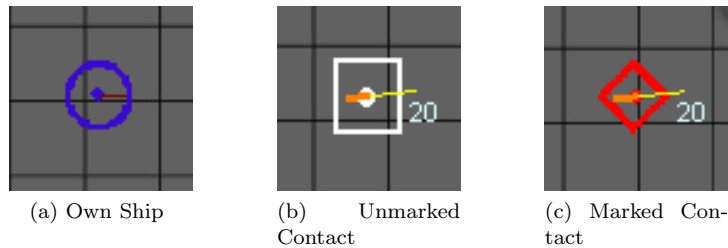


Figure 9: Different element representations.

The speed of a contact is represented by a yellow line. The yellow line is drawn from history points of the contact, it consists of the contact's previous path for a fixed number of past time points. This means the longer the line is, the faster the contact is moving (x time steps ago, the contact was further away from its current position). When a contact is stationary, all previous time points will be at its current position, so no yellow line will be visible.

The orange line of the contact represents its front, so the current heading. In the example contacts in figures 9c and 9b, the heading of both contacts is west.

Normally contacts will be moving within a predefined sea lane, which is represented by the green lines (in figure 8).

The last idcrit is the distance, which is represented by the distance between the blue icon (own ship) and the contact in question.

The contacts move according to predefined scenarios. Contacts can change from being relatively not threatening, to more threatening or vice versa. A contact could for example first be stationary, and then suddenly start moving towards the own ship at high speed. Several situation changes will be occur during one set. The goal of the participant is to keep track of the five most threatening contacts by marking them and constantly updating the marked contacts as the situation changes. The scenarios are designed by Lucassen [9]; he designed an easy and a hard scenario. Both scenarios were be used in the experiment.

6.2 Task Support

During the task execution, the user was supported by a decision support system. In time intervals, the support system generated advice statically, which could then be modified by an adaptive strategy (see figure 7).

The static task support is based on a determination of the five most threatening contacts, calculated by the support system. Its verdict will then be represented to the user on the task display. At each time interval, the five most threatening contacts which are calculated by the support system are compared to the actual classification of the user. Contacts which are wrongly classified according to the support system and would need the *action* from the user to change its state, will be made more salient (highlighted) to the user. This is done by making the other contacts more transparent and making the lines of the wrongly classified contacts thicker.

An example can be found in figure 10. In this example, contact numbers 3, 6 and 10 are highlighted, meaning the support system would change the state

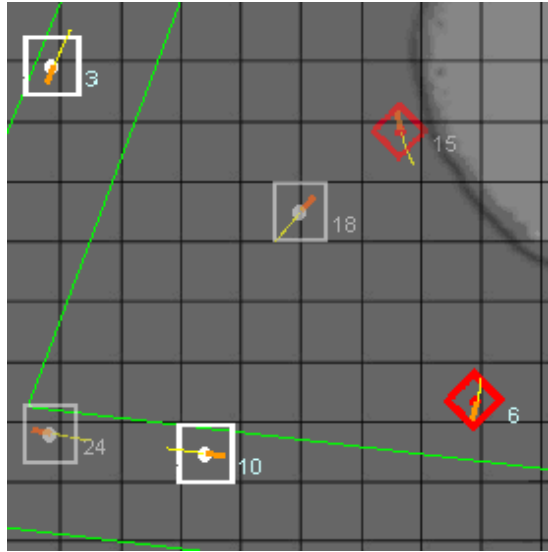


Figure 10: Screenshot of task environment, an example of the static task support representation.

of these contacts from threatening to non threatening or vice versa. The system advises to leave the other contacts unchanged, since they are not highlighted. The classification of the support system would be classify contacts 6, 18 and 24 as not threatening and 3, 10 and 15 as threatening.

Using the support visualization, the user can choose to follow the support system by clicking on the highlighted contacts, or choose to not follow the system's support. If there are no highlighted contacts, the user has classified all contacts exactly similar to the support system. The more highlighted contacts, the greater the difference between the user's classification and the system's classification is.

6.2.1 Calculation of Threat Values (Ground Truth)

The static support was determined by calculating a threat value for every separate idcrit and combining them to one global contact threat level. Since this is a simplified task, each idcrit can be calculated by one function, where the mean of all idcrits combined leads to the optimal contact threat levels (ground truth). Below for each separate idcrit its function:

- **Heading**, a contact which is sailing in the own ship's direction is more threatening than a contact moving in the opposite direction. This can be quantified by determining the angle between the contact's direction and the position of our own ship: coming straight at the own ship has a threat level of 1 and the opposite direction leads to a threat level of 0.

$$T_i^b(t) = 1 - \frac{|B_{i,o} - H_i|}{180} \quad (4)$$

Where $B_{i,o}$ is the bearing of contact i towards the own ship o and H_i is the heading of the contact. When heading and bearing are similar, the threat

Rank x	Contact i	Threat value $C_i^T(t)$
1	5	0.85
2	2	0.81
3	1	0.8
4	3	0.75
5	7	0.71
6	4	0.67
\vdots	\vdots	\vdots

Table 1: Example of a sorted contact threat list.

level is 1, when the contact is moving in the opposite direction, the threat level is 0, all levels in between leads to a threat score between 0 and 1.

- **Distance**, a contact closer to the own ship is more threatening than one further away. A function for the distance can be constructed, where contacts close by gets a threat level near 1 and ones that are far away gets a score near 0.

$$T_i^d(t) = 1 - \frac{dis_i}{dis_{max}} \quad (5)$$

Where dis_{max} is the (predetermined) furthest possible point from the own ship on the radar. The distance dis_i is the distance from the own ship to the contact C_i .

- **Speed**, a contact with higher speed is more threatening than slower moving contacts. This can be calculated by a function on the contact's speed. The faster a contact is moving, the closer the score of the contact is to 1. Slower moving contacts gets threat scores near 0.

$$T_i^{sp}(t) = \frac{sp_i(t)}{sp_{max}} \quad (6)$$

The threat value of speed is the speed of the current contact i at time point t , divided by the maximum speed a contact can have (maximum speed is predetermined).

- **Sea lane**, a contact moving in a sea lane is less threatening than contacts moving outside the sea lane. This can be modeled using the following function:

$$T_i^{sl}(t) = \begin{cases} 1 & \text{if contact is not in sea lane} \\ 0.1 & \text{otherwise} \end{cases} \quad (7)$$

These rules were defined by Lucassen [9] and were kept similar throughout this experiment. The given rules can be calculated for every contact i in the environment. The total threat score of a contact can be calculated by the mean of all four idcrits $R = \{b, d, sp, sl\}$, where $C_i^T(t)$ is a contact threat value and R is the set of idcrits.

$$C_i^T(t) = \frac{\sum_R T_i^r(t)}{|R|} \quad (8)$$

Rank x	Contact i	Threat value $C_i^T(t)$		Rank x	Contact i	System Threat value
1	5	0.85	→	1	5	0.91
2	2	0.81		2	1	0.89
3	1	0.8	→	3	2	0.79
4	3	0.75		4	4	0.72
5	7	0.71	→	5	7	0.68
6	4	0.67		6	3	0.65
⋮	⋮	⋮		⋮	⋮	⋮

(a) Ground Truth.

(b) Noisy Threat Values.

Figure 11: Example of noise addition to the contact threat values.

The resulting values of all contacts i can be compared to determine which contacts are most threatening. An ordered list of all contacts can be made, sorted by their threat level, an example is given in table 1. From this table the five most threatening contacts could be selected as the first five on the list, in the example from table 1 the five most threatening contacts would be $i = (5, 2, 1, 3, 7)$ respectively (the top 5 ranked contacts), the other contacts are not threatening. Note that these are the optimal contact threat values (ground truth). These values can be used to determine how many classification mistakes the agents make in each time point, so what their performance is.

6.2.2 Addition of Noise (System Threat Values)

The threat calculations from the previous section result in the ground truth of the situation, the optimal classification. The support system could not use this information directly otherwise it would know in each situation what the best solution is. This would make the user’s opinion redundant (following system’s advice would always be the best option). In real life the task situation is much more complicated, leading to a much more complicated set of idcrits. The inputs will be measured by noisy instruments, so the system would make mistakes. Noise was added to ensure the system makes mistakes.

The percentage of right classifications needed to be 70% on average. This was chosen because literature shows a lower system correctness could lead to total disuse of the advice system, and above this correctness people will tend to fully rely on the support system. Around the 70% is the correctness where people will still consider to rely or not rely on the system, making it possible to test the support system [13, p. 42].

To simulate a realistic environment, a noise function was used to make sure the system will not always perform correct. The noise works on the separate idcrits before they were used to calculate global threat levels of the contacts. The noise is a random factor added or subtracted from the ground truth threat values, this can be explained as a noise on the measurements of the support system (so sensory noise). The noisy values of the separate idcrits could lead to a different global threat level for every contact, leading to a different ordering of the threatening list, so different advice.

The noise that was added to the idcrits is a random factor determined by a normal (gaussian) distribution with the ground truth value as mean and a

predetermined variance. For each contact noise is being added to the threat values, resulting in different global threat values. In figure 11 an example of the noise calculation is given. In figure 11a the original table is given, containing the ground truth. Noise was added to the threat values, leading to different global threat values (given in figure 11b). As can be seen in the figure, certain contacts are ranked differently. For example contact 3 is now ranked 6th instead of 4th. This means it is classified as not threatening by the system, while in the ground truth it is classified as threatening. This way mistakes are generated. If the top-5 ranked contacts are compared (ground truth are $\langle 5, 2, 1, 3, 7 \rangle$ and system threat values are $\langle 5, 1, 2, 4, 7 \rangle$), can be determined that the system has made 1 mistake out of 5 (contact 3 is not on the list).

The system needs to make a number of mistakes which is still sensible for the user to believe (realistic system performance). When the situation is very easy (the top five contacts are far more threatening than the others), the probability of the system having the right classifications should be higher. To determine how difficult the classification is in a situation, a *situation difficulty value* was used. The situation difficulty is the distance between the threat values of the top-5 ranked contacts and the threat values of the other contacts. When the difference is greater, the classification must be easier, so the system should create fewer mistakes to be realistic. If, for example the top-5 ranked contacts have a threat value of 1, and the other contacts have a threat value 0, the classification would be easy, since five contacts are much more threatening compared to the others. In this situation the system should also make fewer mistakes to keep the support credible for the user (noise was less likely to disrupt the top five contacts).

Using predefined thresholds on the situation difficulty, the number of mistakes were determined. The thresholds are determined in a way that the average system performance throughout the experiment was 70% correct. It is predefined that the system will make 0, 1, 2 or 3 mistakes out of 5 (more mistakes leads to total incredibility and should be avoided), based on the situation difficulty.

The relation between the number of systems mistakes and the current situation difficulty can be trained on the task environment (by setting the thresholds). The task environment is constant (all experiments will be run using similar scenarios). This means all scenarios can be run in advance to determine the best threshold settings. The average correctness will be 70%, using the predetermined parameter settings. The thresholds were determined so that most of the times the system would make 1 or 2 mistakes (reliability of 0.8 and 0.6 respectively), but easier and harder scenario difficulties lead to a fewer or more mistakes. The thresholds were set in such a way that the average system reliability was 0.7 using the pilot data.

When the number of desired mistakes was determined (using the thresholds on the situation difficulty), a system classification had to be calculated (from the ground truth) containing that number of mistakes. To make this process easier, the variance of the noise calculation was determined by the situation difficulty. When more mistakes were needed, the variance was higher, so noise disrupted the ranked contact threat list easier, leading to more mistakes. When recalling figure 11 could be noticed that a greater variance could result in greater differences in the threat values of the system threat values compared to the ground truth. The order of the rankings of the threat list could also differ more

easily, leading to more mistakes.

The generated noisy data was used to calculate the static task support, which could be visualized to the user.

6.3 Trust Models Description for the Task Environment

For the advice to be adaptively changed, the trust appropriateness state needs to be estimated by the system. In section 3 the trust models that would determine the trust appropriateness state were introduced. The output of these models was then used in the adaptive support strategies (see figure 7 on page 29 for the complete structure of the decision support system). The trust models needed to be applied to the task environment to return the right output. In the sections below all trust models are implemented on the task environment. Explanations will be given of what information they use as inputs, how they calculate their output and how the trust models result in a trust appropriateness state for this specific task environment.

6.3.1 Prescriptive Model

One measure of prescriptive trust is self confidence (as explained in section 3.2). The prescriptive model does not have access to the ground truth, so given the lack of the performance data, self confidence was used as a prescriptive trust estimation. To apply the prescriptive model to the TPC Task, the system needs environmental information to estimate its self confidence. A way of measuring self confidence is by comparing the threat values of the separate contacts to each other. This is explained below.

The five most threatening contacts are determined by calculating all contact threat values separately and constructing an ordered list by their threat values (example is given in figure 11 on page 34). As explained in section 6.2.2, mistakes are made when noise is added to the threat values (noise on the separate idcrits leads to a different total threat value). These noisy threat values can result in another ordering of the contacts in the list. The contacts can be ordered differently in the noisy contact list than in the ground truth. The contacts that are not in the top 5 in the noisy threat value list (and were in the ground truth) are classified wrong, and vice versa. This way mistakes were made by the system.

The ordering of the contact threat list means the contacts ranked 1-5 are classified as threatening, while the other contacts are classified as not threatening. Mistakes are made when contacts who are threatening in the ground truth (ranked 1 – 5), are ranked lower than 5 in the system’s (noisy) threat value list.

Another characteristic of this list is that the contacts ranked 6-10 have threat values closest to the threatening classifications (closer than threat values of contacts ranked lower than 10). This means if mistakes are made, the probability is higher that by the noise, a contact ranked in the 6 – 10 range in the ground truth, is now ranked as threatening (1 – 5). The contacts 6 – 10 are the contacts that are most likely to influence the ordering of the top-5 in the list. In the worst case scenario, when all classifications are wrong, the probability is highest that the contacts from the ranks 6 – 10 are now classified as threatening (instead of contacts ranked lower than 10 being wrongly classified as threatening), since the threat values of the contacts ranked 6 – 10 are closer to the threat values

of the top-5 ranked contacts. Therefore contacts ranked lower than 10 are less relevant than the contacts ranked 1 – 10, since they are less likely to influence mistakes. Therefore the contacts 1 – 10 were chosen to be used to calculate self confidence.

Self confidence is dependent on the probability of wrong classifications being made. This can be done by comparing the threat values of the top-5 ranked (threatening) contacts to the threat values of the contacts ranked lower than 5 (non-threatening contacts). When values from the top 5 are much higher compared to the values of the contacts of ranks 6 and lower, probability is higher that the noise had less impact on the ordering of the list, so it is less likely that wrong classifications were made. However, when the values of the threatening classifications are very close to the non threatening classifications, mistakes are easily made when noise is added. This means that the more distinct threat values of threatening contacts are from contacts that are not classified as threatening (especially the contacts ranked 6 – 10 because they are more relevant), the more certain the system can be that its classification is correct. On the other hand, when threat values of threatening contacts are more similar to values of the contacts ranked lower, it is more likely that noise has caused mistakes. This effect can be used as a measure of self confidence.

The effect needs to be quantified in order to get the corresponding prescriptive trust. This is done by taking the boundary of the threat values (the average of the threat score of contacts 5 and 6) and calculating the distance from the contacts to this boundary. When this distance is greater, the threatening contacts are higher compared to the non threatening contacts. When the distance is smaller, the threatening contacts are less distinct from the other contacts, so it is more likely that noise has caused mistakes. So a greater distance leads to more self confidence and a higher prescriptive trust.

Since the contacts ranked 1 – 10 are most relevant (even in the worst case scenario, with 5 mistakes out of 5), only these contacts will be used to calculate self confidence. Another reason leave the lower ranked contacts out of the self confidence calculation is that the distance would be significantly greater in each time point, since the contacts at the bottom of the list have far lower threat levels (more distant from the boundary). The contacts ranked lower than 10 are not relevant, but they can negatively influence the self confidence measurement by being different in each situation. Therefore, only the contacts ranked 1 – 10 will be used.

The distance was used to calculate the task difficulty (self confidence) at each moment, using the distance of the contacts as previously described. Note that only the threat value that were calculated by the system (with the noise included) were used, the ground truth was not available to the model itself. The distance was calculated as follows:

$$d(t) = \frac{\sum_{x=0}^{10} |C_{i,x}^T(t) - b(t)|}{10} \quad (9)$$

Where $b(t)$ is the boundary of the threatening and non threatening contacts, calculated by the mean of the threat score of the contacts at rank 5 and 6 ($b(t) = \frac{C_{i,5}^T + C_{i,6}^T}{2}$). Furthermore $C_{i,x}^T(t)$ is a threat value T of contact number i with rank x at time point (t) . The outcome of formula 9 is the self confidence of the classification at the specified time point.

Thresholds on self confidence $d(t)$	prescriptive value	
$Z_{d(t)} \leq -2$	0.4	worst
$-2 \leq Z_{d(t)} \leq 0$	0.6	
$0 \leq Z_{d(t)} \leq 2$	0.8	
$2 \leq Z_{d(t)}$	1.0	best

Table 2: Translation from self confidence (z-scores) to prescriptive trust values, using thresholds.

The self confidence value only makes sense in comparison with other self confidence values. The value can be used to compare environment situation can be compared to other environment situations.

The self confidence value is an indication of the self confidence of the system in the advice it's giving in a specific situation. The self confidence can be used to compare different situations with each other. Since the self confidence is based on the scenario's of the task, the self confidence will be roughly similar when the scenario is repeated (since the situations are similar). The self confidence values could be sampled in advance by running all scenario sections once, having for each situation (time point) a sample of the self confidence. With a sample of all possible self confidence values, a measure can be constructed to make sense of this sampled data. Note that the self confidence is calculated by taking the system's *noisy* threat values (which differ slightly in similar situations), but since they are calculated in a similar way, the output is roughly similar each time for the similar scenario parts. Therefore the sample data is a good indication of all the different self confidence measurements during the experiment.

To compare the self confidence values, z-scores were used. The average and standard deviation were determined using the sampled self confidence values and were kept constant during the experiment. Using the average (μ_D) and the standard deviation (σ_D), a z-score was calculated for each newly determined self confidence value, using the z-score formula:

$$Z_{d(t)} = \frac{d(t) - \mu_D}{\sigma_D} \quad (10)$$

The self confidence (in the form of the above calculated z-score) could be used to determine prescriptive trust. However, prescriptive trust is scaled differently: a prescriptive trust value of 1.0 means an expected system performance of 100%. Similarly a value of 0.4 means a correctness of 40% is expected. As explained in section 6.2.2, the noisy threat values will result in 2, 3, 4 or 5 correct classifications out of 5 (40, 60, 80 or 100 percent correct respectively). The prescriptive trust value is the expected correctness of the system threatening classification. This means correctness of the model is divided into intervals of 0.2 (so the model could have possible prescriptive values $p(t) \in \{0.4, 0.6, 0.8, 1.0\}$). A translation has to be made from self confidence values to the possible prescriptive trust values.

The translation was made using thresholds. When the self confidence z-score is high (high compared to other self confidence values), the prescriptive trust value should be high. Lower self confidence values should lead to lower

Type	Highlighted	User Threatening
Hit (H)	0	1
Miss (M)	1	0
False Alarm (FA)	1	1
Correct Rejection (CR)	0	0

Table 3: Reliance signal detection. Highlighted is 1 if the contact is advised by the support system to change at the current time point, 0 otherwise. User threatening is 1 if the user has classified a contact as threatening, and 0 otherwise.

prescriptive trust values. Since the possible prescriptive trust values are predetermined, and the z-scores are already sampled for the scenarios, the thresholds were determined. The resulting threshold table is given in table 2. The thresholds $(-2, 0, 2)$ were set manually using the previous described data, so that the average expected performance of the system is 70%.

6.3.2 Descriptive Model

The purpose of the descriptive model is to estimate the user’s trust in the support system. As explained in section 3.1, it uses different input information to estimate this, by for example using user reliance information on the advice.

The experiment environment (TPC Task) was constructed in such a way that there are only two possible agents to trust, the user could either trust the system or himself. This means the decision on who to trust is binary. This property is crucial in order to decide what type of reliance model is appropriate for this task. The user could only trust one of the two agents and a decision to trust one agent, is automatically a decision to distrust the other agent (when the two agents would make different decisions). Since the reliance decision is binary, only estimating trust in the system is necessary. Trust in the human agent would not be necessary to calculate because this could already be derived from the trust in the system. In section 3.1 two reliance models were described: the independent trust model and the relative trust model. Since only the estimation of trust in the system is needed, disregarding the trust in the other agent, the independent trust model was needed.

The independent trust model is based on experiences, which needs to be defined for this specific task environment. The only appropriate environmental information which is available to the models at all times is the human reliance information. Therefore the experiences in the trust model for this task will be the user’s reliance on the systems support (static support). The decision to rely or not rely on the advice of the system is a binary choice. A contact is either highlighted (meaning the user has not relied on this piece of advice) or the contact is not highlighted (meaning the user has classified the contact according to the systems advice). If all contacts are combined, the total amount of highlighted contacts is a measure of the current reliance of the user in the system. More highlights means less reliance, if there are no highlighted contacts, the user has fully relied on the system.

Since the reliance decision on the system is binary, a signal detection can

be used to calculate human reliance on the system. The signal detection possibilities are defined in table 3. A hit is for example defined as the user having classified a contact as threatening, where he followed advice (the contact is not highlighted anymore when the classification is made). If he classified a contact as threatening and it is highlighted, he decided to not rely on the system, which is called a false alarm. If a user has not classified a contact as threatening, but it is highlighted (so the system would classify the contact as threatening), a miss has occurred. Finally when the user has not marked a contact as threatening, and the contact is not highlighted, the user has made a correct rejection.

However, the system always advises you to classify exactly five contacts as threatening. When the user also marks exactly five contacts as threatening, and they are similar to the advice system, there are 5 hits, 0 misses, 0 false alarms and the resting contacts are correct rejections. However, when the user changes the classification of one contact instead of another contact (situation remains similar), this not only results in a miss (he has not classified a contact that the system would classify as threatening), but also in a false alarm (another contact is classified as threatening that the system would not). The general reasoning is that as long as the user classifies exactly five contacts, contacts can only be swapped, meaning for every false alarm, there is a miss. This implies that the number of false alarms always equals the number of misses ($\#FA = \#M$).

The number hits and correct rejections are not equal to each other, but they are related. For every hit which is not made (fewer hits than the possible 5), there is also a correct rejection less (fewer than the maximum possible correct rejections, which is all contacts minus 5). Note that this only holds when the user and the system have classified (or advised) exactly 5 contacts as threatening.

For the descriptive trust measure, the hits and correct rejections were compared to the number of misses and false alarms. Hits and correct rejections are positive reliance experiences, and false alarms and misses negative. However, the maximum number of correct rejections is dependent on the number of total contacts. Even when the user has classified different contacts than the advised (total disreliance), there are still a number of correct rejections (5 misses and 5 false alarms, the remainder of the contacts will still be correct rejections). It is not desired to count the remainder of the contacts as possible positive reliance experiences, since these experiences are just a consequence of the total number of contacts. The correct rejections are related to the hits (as explained above), so a decrease of hits means also a decrease in correct rejections. Therefore the positive experiences can be calculated by using only the hits.

As explained above, the false alarms are always equal to the number of misses, so for the negative reliance experiences it is only necessary to use the false alarms. The total experience (using the hits and the false alarms) can be calculated using the following equation:

$$E_i(t) = \frac{\#H_t}{\#H_t + \#FA_t} \quad (11)$$

The experience $E_i(t)$ is the number of hits divided by the number of hits and the number of false alarms together.

This estimation is accurate when the user has classified exactly five contacts as threatening, leading to positive experiences (higher $E_i(t)$) when the number of hits is greater (with a maximum of 1.0), and lower experience when the number of false alarms is greater (with a minimum of 0). However, the user has

the possibility to classify more or fewer than 5 contacts as threatening. The reliance experience must be able to cope with this variation.

When the user has classified more than five contacts as threatening, the number of hits cannot be higher than five (since the number of advices is 5), only the amount of false alarms can increase. This means the $\#H_t + \#FA_t$ will increase and the reliance experience will decrease. So if the user classifies more than five contacts as threatening, the experience can only decrease, since only the number of false alarms increase. When more than five classifications are made, such a decrease in trust experience is desired.

If there are fewer than five contacts classified by the user, the reliance experience could get inaccurate. The number of false alarms is in this case not necessarily equal to the number of misses. If the user for example has only classified one contact as threatening (following the advice), there is no false alarm, but there are four misses. Using equation 11 this will result in an experience of 1.0 which is undesired. When the user has marked fewer than five contacts, the number of false alarms should be supplemented to be equal to the number of misses. Using the above example, if the user only has marked one contact, the number of false alarms is set to four (since there are four misses in the environment). This will lead to a more negative reliance experience (of $\frac{1}{5}$), which is desired since the user has not relied on on the missed advices.

With the experience applied to the task, recalling the complete independent trust model on page 17 (equation 1), the only constant that still needs to be set in the independent trust model is the decay of the past experiences (λ_i). The decay is a constant which is higher when past experiences are important in calculating updated descriptive trust, or lower when past descriptive trust is less influential. In the current task, the task environment is very dynamic. The support from the system also is dynamic. Therefore, past reliance decisions are not good indications for future descriptive trust (it could be beneficial to trust the system at one time point, while it was not beneficial before, or vice versa.). This means only the current experiences will be used to indicate current descriptive trust. The decay λ_i will be set to 0. This will also keep the descriptive trust model considerably less complex.

All parameters of the descriptive trust model are initialized, leading to a descriptive trust level of 0 when the user did not rely on the system, and 1.0 when he fully relied on the system, and all graduations in between. This is a reliance estimation of the user in the system. However, a problem can occur when the user decides similar to the systems advice. The system is unable to know the users' actual motivations, whether he followed the advice or he concluded similarly to it consulting the system itself. This could be a problem, but one can say that when a user consequently makes similar decisions to the system's advice, the probability increases that this is no coincidence, and the user is following the advice. The lack of knowledge of the motivation of the user's decisions might lead to less accurate descriptive trust in only this moment, but when multiple descriptive trust values are compared (new trust evidence is collected during each time interval), the descriptive trust estimation could get more accurate.

Another factor which is unavailable to the model is whether the user's reliance decisions were correct, so if he followed the system when its advice was right or wrong. Since there is no ground truth available to the support system, it can not know when advice is correct or incorrect, leaving the descriptive model

only with the amount of reliance decisions without their justifications. The model is in this way a representation of a real life situation, where justifications are also not always available to the model.

6.3.3 Appropriateness Model

The descriptive and prescriptive trust have to be compared with each other to get to a trust appropriateness level. The purpose of the appropriateness model is to make this comparison. In section 3.3 such a structure is given. This given appropriateness model is a general model, which needs to be applied to the experiment environment (the TPC Task).

The prescriptive and descriptive model are already structured in such a way so that they can be used directly in the appropriateness model. If the prescriptive model is an expectation of the system's performance (self trust), with the value literally being the expected correctness (a value prescriptive value of 0.5 means expected 50% correct). On the other hand, the descriptive value is an indication of the level reliance at the same time point of the human in the system. Since the models estimate the situation at the same time point, and are scaled in the same way, they can be directly compared with each other. This can be done by the formula given in section 2:

$$\alpha_i(t) = \tau_i^d(t) - \tau_i^p(t) \quad (12)$$

Negative $\alpha_i(t)$ values means the prescriptive value is higher than the descriptive values, meaning undertrust has occurred (as explained in section 3.3). Similarly, high $\alpha_i(t)$ mean overtrust. Values near 0 mean human trust in the system is appropriate.

However, the task is very dynamic. Since the prescriptive model strongly depends on the task environment, the values of the prescriptive model are also dynamic. The reliance of the user could also be dynamic, since the system changes advice and also its reliability (recall the prescriptive model section 6.3.1 and the descriptive model section 6.3.2). Given this dynamic properties, calculating the trust appropriateness as in the equation above would lead to high fluctuations of the trust appropriateness levels. The adaptive advice needs to be stable, so it is undesirable to change trust appropriateness state in every recalculation. To solve this problem a decay is added to the trust appropriateness calculations. Human trust also does not vary every second, but new evidence is needed to change the trust appropriateness. When a person distrusts another agent, much evidence is needed to change its opinion and start trusting the other agent again (one piece of evidence mostly is not enough for the agent to change its trust appropriateness). The trust appropriateness level calculation is done by:

$$\alpha_i(t) = \alpha_i(t-1) \cdot \lambda + \left(\tau_i^d(t) - \tau_i^p(t) \right) \cdot (1 - \lambda) \quad (13)$$

Mostly when a certain trust appropriateness level is reached by the human, it is difficult to influence this level. When a human distrusts some one or something, he is unlikely to switch to fully trusting it in the next time step. The lower the decay is, the less influential past experiences are. However, most people stay pre-cautious in trust related situations, making it harder for a person or system to be trusted appropriately again. It will take time and new evidence to change the trust appropriateness level. Therefore a higher decay is desirable.

The decay is characterized by λ in equation 13, and is a constant which was determined before the models was used. The decay is strongly dependent on the task dynamics, and also on the personality of the human. To keep the complexity of the experiment low, the decay will be predetermined using pilot data and stay constant throughout the experiment, disregarding the personality factor. The decay was set manually using pilot data. It was set quite high (close to 1), so that much evidence is needed to change the appropriateness state. This made the trust appropriateness state determination much more stable than lower values. Note that all factors in the trust models are be predetermined and kept constant during the experiment.

The calculated trust appropriateness *value* was used to determine the trust appropriateness *state*. As explained in section 3.3, the state was determined using thresholds. In the implementation of the trust models on the task environment, these thresholds have been determined manually using pilot test data. The trust appropriateness value can then be used to determine one of the three trust appropriateness states: undertrust (value lower than the lower threshold), overtrust (value higher than the upper threshold) and appropriate trust (when the value is in between the two boundaries). The thresholds were set so the model would output undertrust in 25% of all cases, overtrust in 25% and appropriate trust in the other cases of the pilot data. These percentages were set to let the adaptive advice be present in enough cases. This way the adaptive advice could make a difference in performance, but was not too obtrusive.

6.4 Participants

A total of ten participants were used for the experiment. Average age is 23.5 year (standard deviation 5.75). From the 10 participants, 80% was male.

The participants were naive to the task in advance, they were in fact not trained for executing naval tasks such as the TPC Task at all. The only requirement for the participants was computer experience (average computer usage per week is 27.3 hours). Since the task is quite complex, the factor of it being executed on a computer should not have led to any extra learning effects. Every participant was used to working with computers.

6.5 Design

The experiment had a repeated measures design with Support Type as factor. Three levels of support types were used. The order of the conditions was balanced throughout the participants. The conditions were preceded by a training session (of equal length as the conditions).

There were four different scenario parts available (the easy and hard scenario's were divided into two parts each), one part for each condition and one part as training exercise. The scenario parts were balanced over all conditions (including the training). A repeated measures ANOVA analysis has been done to check for differences between the scenario parts. No differences were found.

6.6 Independent Variables

The independent variable of the experiment is the Support Type. Two types of adaptive support were given: Argumentation Support (AS) and Censorship

Support (CS), based on the adaptive strategies of section 4. The adaptive support types need to be compared to a static type of support, so the third support type is the Static Support (SS). The three different types of support form the conditions of the experiment. The conditions will be described below.

6.6.1 Static Support

the SS condition is the control condition, where the adaptive strategies were compared to. The SS condition was only based in static support, which is explained in section 6.2. The system will highlight contacts, which need to be changed according to the systems calculations. This will be calculated regardless of different trust appropriateness states of the participant in the system.

6.6.2 Argumentation Strategy Support

The argumentation strategy is based on the SS condition, but it will also take the trust appropriateness state into account. In section 4.1, the different trust appropriateness states are explained, with their consequences for the argumentation. They were *certainty* arguments during undertrust, *doubt* arguments during overtrust and *omission of information* in the appropriate trust state. Different states are applied to the task using the SS condition as the standard support, which will be visible at all times. The static support is further elaborated by presenting extra arguments to the user, substantiating the given advice (change/don't change the contact's threat state because ...). The arguments are based on the idcrits, where the static support is also based on. This could give the user a better transparency of how the static advice was calculated. For each trust appropriateness state, different arguments are given.

The arguments consist of a qualifier of the advice itself (the advice to change or not change). The arguments are given to show why the advice is given, which is the first step of backing up a claim in Toulmin's model of arguments (see figure 2 on page 12), referred to as the qualifier.

When the system detects undertrust it can give certainty arguments. This is visualized in the task environment as green flickering cues, distinctive for each of the four idcrits. The selected contact is colored light blue inside to clarify which contact the arguments are given for. The support system is giving additional information why it is certain its advice to change or not change the state of the contact is correct, based on which idcrits. The representation is shown in figure 12.

In figure 12a the advice is to let the contact be not threatening. The additional argument is that the distance is a factor that makes this contact relatively not threatening. In the situation of figure 12b, the system advises the human to change the state of the contact in question to threatening. The contact is threatening because it has manoeuvred out of the sea lane. In figure 12c the contact is not threatening because its direction is not threatening, to show this to the user the orange heading line is colored green. In figure 12d the yellow line has turned green. The system reasons the contact is not threatening because his speed is not very high. It can occur that more than one argument is given where the support system has based its advice on, or that no sensible arguments can be given.

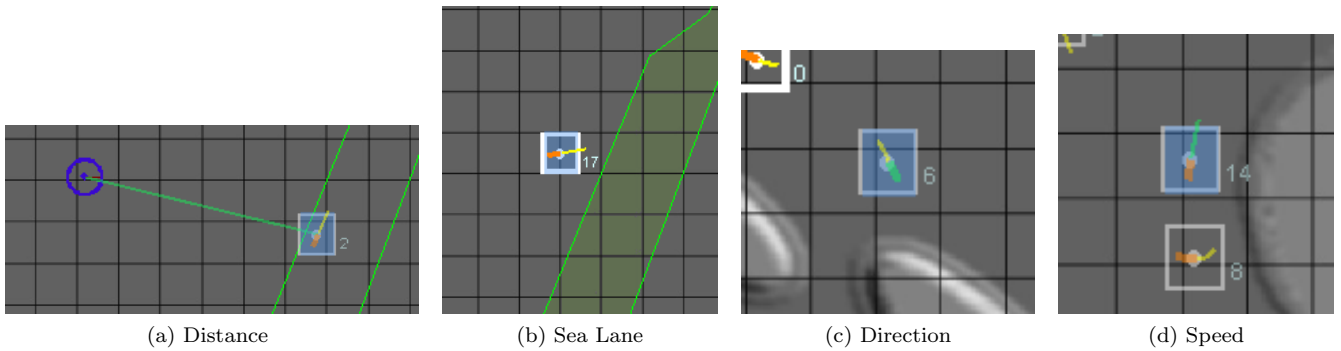


Figure 12: Visualization of certainty arguments based on idcrits.

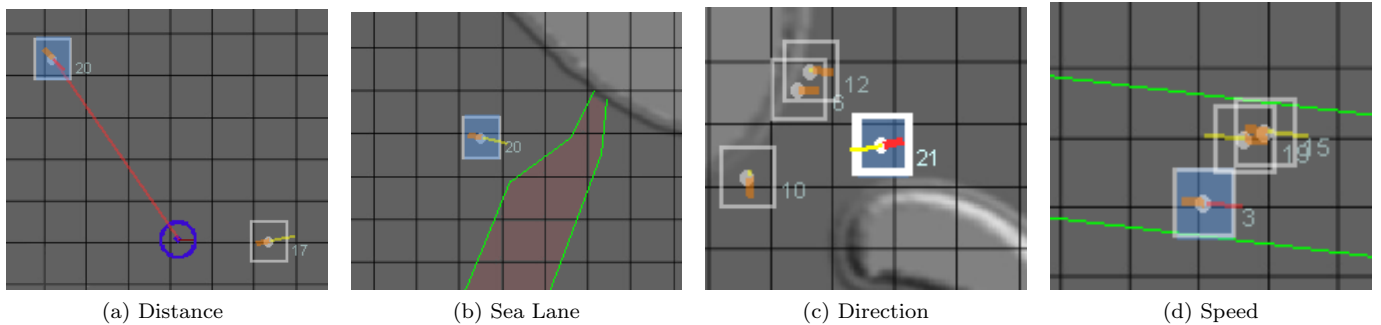


Figure 13: Visualization of doubt arguments based on idcrits.

In the appropriate trust state, the adaptive support has the omission of information. This means the participant uses the support system appropriately, and should not be interfered by extra information from the support system. In this state, no additional information will be present but the static support.

The last state is when overtrust is detected. In this situation extra arguments are needed, since the user's trust in the support system is not appropriate. According to the argumentation strategy, doubt arguments are in order to be presented to the user. This will be presented in a similar way as the certainty state, but with a different color. While certainty is green, doubt arguments are red. Red means, watch out, maybe the advice is not right, these elements cause uncertainty. In figure 13 examples of all arguments can be found, based on their idcrits.

The first figure shows an advice of leaving the contact as not threatening, while the system thinks it might be not true since the contact is fairly close to the own ship. Figure 13b shows an uncertain advice of leaving a contact not threatening, but the contact is moving out of the sea lane. The direction of the contact in figure 13c makes the contact not threatening, while the advice of the system would be changing the contact to threatening. In the last picture the advice is to mark contact 3 as not threatening, but its speed is actually quite high, making the system doubt its given advice.

The system has no access to the ground truth, also not in the argument calculation. The system threat values are based on the noisy separate idcrits

(as explained in section 6.2.2). This means the arguments are (like the advice itself) not always correct, which is realistic. So a percentage of the arguments will be valuable, but the argument can also be incorrect, which could give the user a better insight why and when the system gives wrong advice sometimes.

In each time step, the threat values of the separate idcrits of all contacts can be compared to each other. When a threat value is high or low compared to the threat values of other contacts, the specific idcrit can be used as an argument for this contact. Therefore, not the raw system threat values will be used for the determination of arguments, but the z-score will be used instead. When t_i^r is the value of the idcrit r of contact i , and T^r is the set of threat values of all contacts for idcrit r , then the z-score of an idcrit can be calculated by the formula:

$$z(t_i^r) = \frac{t_i^r - \mu_{T^r}}{\sigma_{T^r}} \quad (14)$$

The z-score will result in a normalized threat value for the idcrit r of contact i , where this value will be lower or higher than 0 when the value is more distant from the values of other contacts and closer to 0 otherwise. Using the z-scores can be assured something is an argument, only when it outstands from the idcrit values of the other contacts, or is an uncertainty argument when it does not. The arguments are given when a value of an idcrit exceeds certain thresholds. When it is below a lower threshold, it would be an argument why it does not make it threatening (it is not threatening in comparison to other contacts). When it is between thresholds, it is not notable from the other contacts, which could make the support system doubt its advice.

$$\text{certainly not threatening} \leq th_1 \leq \text{doubt} \leq th_2 \leq \text{certainly threatening} \quad (15)$$

The thresholds (th_1 and th_2) will be determined manually using pilot data. In pilot testing, the z-scores of the idcrits can be visualized separately, to determine what the best values are for the thresholds. This is task dependent, and will be kept constant for all participants.

6.6.3 Censorship Strategy Support

The Censorship condition is based on the SS condition. Unlike the argumentation condition, the CS will influence the static advice itself rather than providing extra information.

There are three possible trust appropriateness states, where according to the adaptive strategy model from section 4.2, reacts in three different ways: *modesty, appropriate and presumptuous*

The first trust appropriateness state is undertrust. When under trust is detected, the system should be modest with its advice. Since the user does not use the information from the support system, the system would have to be modest by only displaying advice with high certainty. This could help trust calibration of the user in the support system. In this state the advice will be censored as much as needed (a fixed percentage of the advice), forcing the user to think for himself and only supporting him with sensible (obvious) advice. The censoring of information is done by controlling the amount of highlights. Normally the system will highlight the contacts which need attention from the user, because these are the contacts that must be changed. The censored contacts will simply

not be highlighted before the next advice consideration has been calculated. This leads to less highlights on the screen, so less interference from the support system and less desired actions for the user to follow.

When the trust is appropriate, the system has detected that the user is using the system as it should. Therefore the system will give full advice in this appropriateness state.

Finally the system could detect overtrust of the user in the advice. In the censorship strategy this is explained as the user being presumptuous. He follows the support system too much, without having enough doubt in the system's advice. In this case a percentage of the advice will be censored, disallowing the user to literally follow all the advice, stimulating him to think for himself. During overtrust, the most obvious advices will be censored. The user is given only the advices which could be valuable, but leaves out the obvious ones. This could induce the user to think for himself and stop blindly following the support system. What would mean a better trust appropriateness in the advice.

6.7 Dependent Variables

The dependent variables were performance, human trust feedback, trust appropriateness ratio and different aspects of the experiment by questionnaires.

To validate the different trust models, different dependent variables were used.

The hypothesis concerning descriptive trust (hypothesis 1) states that descriptive trust is a measurement of real human trust. The descriptive model output can be compared to the human trust feedback.

The prescriptive trust is an estimation of the system's own performance. Hypothesis 2 can be tested by measuring the actual performance of the system.

The appropriateness model outputs a trust appropriateness state. This state is according to hypothesis 3 a predictor for the real trust appropriateness state of the human in the system. To test the hypothesis, the trust appropriateness ratio is used.

Hypotheses 4 and 5 are concerned with the performance of the human-computer team. This can be tested by using the actual performance of the human and system. Below the performance measure is further described.

As explained in section 4, possible performance improvements using the different adaptive support types could be explained by the trust behavior and the trust appropriateness experience of the user. To test these effects, the trust appropriateness ratio and the human trust feedback measures can be used.

6.7.1 Performance

The performance is a natural number between 0 and 1 (0 meaning 0% correct, 1 meaning 100% correct). This was calculated in every task update interval, which was once a second. The performance measure was similar in each condition.

Performance can be calculated for both the human and the support system. Both agents are estimating what the best classification would be for all contacts, what the five most threatening contacts are. The performance is calculated comparing the system or participant's classification to the optimal classification (ground truth).

Signal Type	Contact Threatening	
	Truth	Agent
Hit (H)	1	1
Miss (M)	1	0
False Alarm (FA)	0	1
Correct Rejection (CR)	0	0

Table 4: Possible signals, for every possible state of a contact.

The performance can be calculated using signal detection, where the top five ranked contacts from the optimal threat list are compared to the participant or the system’s classification. In table 4 the possible signals can be found. If for example, a contact is threatening (ground truth) and it is classified by the agent (user or support system) as threatening, results in a hit. When the contact is threatening and it is not classified as threatening by the agent, is a miss. Contacts which are not threatening and classified by the agent as threatening or not threatening will result in either a false alarm or a correct rejection.

In a normal situation, both the ground truth and the the agents will have five threat classifications and the other contacts are classified as not threatening. In these cases a threatening classification of the *agent* will either be a hit or a false alarm. Since there are only five classifications, the number of hits and false alarms will always be five in total. If there is a false alarm it implies there also is a miss (there are only five classifications, when one is wrongly classified as threatening, there is another threatening contact which is not classified as threatening). This means the false alarm rate is proportionate to the miss rate. In a similar way the hit rate is proportionate to the correct rejection rate (if the hits are decreased by one, the is also one less correct rejection). For a performance measure, only the hit and false alarm rate are needed, leading to the following penalty function:

$$P(t) = \frac{\#FA}{\#FA + \#H} \quad (16)$$

The hits and false alarms can be used to calculate performance. However, one mistake is more probable to be made than another. As explained in section 6.2.2, one situation can be more difficult than another, when the threat values of the threatening contacts are more alike the non threatening contact values. Also when contacts are classified as threatening while they are certainly not, is a greater error than contacts which are less conveniently threatening.

This is accounted for in the penalty calculation, by adding weights to the hits and false alarms of the contacts. The boundary threat level is calculated by the average of contacts at ranks 5 and 6. Contacts with a higher threat score are threatening, others are not. When a false alarm has been made by classifying a contact with a threat score near the boundary, the mistake is subtle. The task of distinguishing the contact from the threatening contacts was much harder, therefore mistakes should not be punished very hard. However, when the distance between the boundary and the incorrectly classified contact is large, the misclassification is more convenient. This should lead to a greater penalty.

However, when a hit has been made, hits near the boundary are more conve-

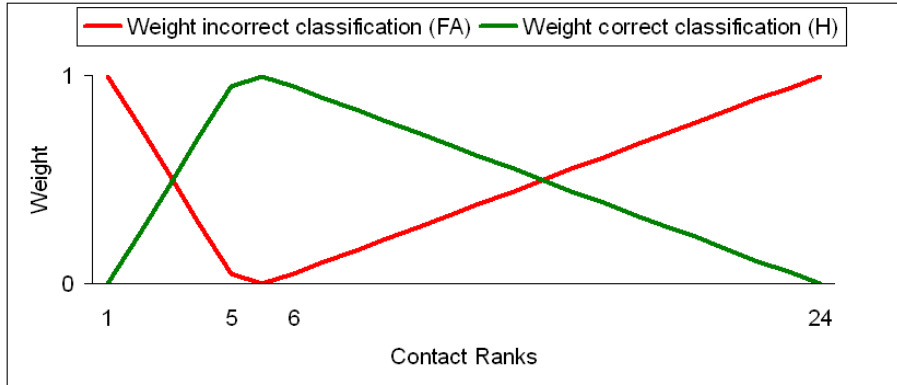


Figure 14: Performance, weights of contacts for different situations.

nient than the obvious classifications. Therefore hits are weighted the opposite way as the false alarms. For both hits and false alarms the distance is a measure of their weights. In the experiments a scaled version of the weights is used in task performance calculation (figure 14). The figure shows the weights are scaled. On both ends of the scale the maximum will be reached, with maximum penalty having the outer most contacts classified as threatening while they are not, and minimum penalty having the most difficult contacts classified wrong. On the other hand the most difficult classifications will be rewarded most and the most obvious classifications will be rewarded least.

The weights need to be applied on all contacts, therefore equation 16 (given above) needs to be modified. Instead of just counting the number of hits and false alarms, each hit and false alarm has to be multiplied by its weight and summed. The penalty using the weights can be described by the following equation:

$$P(t) = \frac{\sum_{fa} w_i C_i^{fa}}{\sum_{fa} w_i C_i^{fa} + \sum_h w_i C_i^h} \quad (17)$$

The above given equation is an accurate performance measure in most cases. However, the human is able to classify more or fewer than 5 contacts as threatening. In this situations the penalty must be heightened, because the user is supposed to select precisely 5 contacts. In the case of too many contacts, there will only be more false alarms (there can only be 5 hits, furthermore only false alarms). This will increase the penalty as desired. When too few contacts are classified, a situation could lead to a too high performance. When for example only one contact is selected and it is correct, the penalty will be 0, while the resting four possible misses are not accounted for. The penalty could be made more accurate to add missing hits to the false alarms. This will increase the penalty when items are not classified as they should have been. In this way the total hits and false alarms will always be 5. The weight of the false alarms should not be too high (classification will lead to maximum penalty) and not be too low (mistakes will not be taken into account). Therefore the weight will be the average of all weights of the contacts. Using this mechanism, too few selected contacts will lead to a higher penalty, as desired.

Finally using the penalty function (function 17), the performance can be calculated by:

$$C(t) = 1 - P(t) \quad (18)$$

Since the penalty $P(t)$ is a number between 0 and 1, the final correctness $C(t)$ is also scaled from 0 (worst performance) to 1 (best performance).

6.7.2 Human Trust Feedback

The human trust feedback is the degree of trust the participant has in the support system. The participant is asked to answer the question "To what extent are you using the support system" during the experiment with a one minute time interval. The result is a number between 0 and 9 (the numbers on the numpad of the keyboard). The measure will be similar for each condition.

6.7.3 Trust Appropriateness Ratio

At each update interval (one time each second), the complete task world environment was logged for later analysis. The classification of the advice system and participant can be evaluated. A determination can be made for each contact whether the participant followed the system advice (or not) and whether this decision was correct (using the ground truth). This will result in a number of over reliance decisions (followed incorrect advice), under reliance decisions (did not follow right advice) or appropriate reliance decisions (followed right advice and rejected false advice). A ratio can be calculated of trust appropriateness, under- and overtrust per time step.

The participant only marks 5 contacts and the advice system also only marks 5 contacts, the focus of trust appropriateness only depends on the marked items. The other contact are less relevant and will be left out of the calculation. If 5 contacts are marked and 5 contacts are advised, the other contacts will automatically result in appropriately rejected, which is not desirable to take into account (similar effect as for the descriptive model signals in section 6.3.2).

The trust appropriateness ratio is calculated at each time interval (once a second) and will result in three proportions (between 0 and 1): proportion undertrust, proportion appropriate trust and proportion overtrust.

6.7.4 Questionnaires

During the experiment, the participants were given various questionnaires. These questionnaires were used to determine subjective system trust (after each condition), estimation of system performance, fatigue, concentration and other factors. A complete list of all questionnaires can be found in appendix A.

6.8 Task Implementation Issues

This section shows task implementation details. Why certain choices have been made in order to get the experiment running as desired. Some parts will be elaborated by code fragments of the implementation. The program is partly created using visual cues, therefore the complete code section could not be included in this thesis. The complete experiment implementation can be requested by either sending an email to me, or dr. P.-P. van Maanen.

6.8.1 Server and Task

The task is designed to work using two computers: a server machine and a task machine. Since the experiment could use much resources it was decided to divide it on two machines. However, structural problems (i.e. connection delays) and task simplifications (all model parameters are kept constant during the experiment) lead to an implementation that could run on a single computer.

Two separate programs were used for the experiment. The task was programmed in GameMaker, a program made for designing graphical interfaces like games using an object oriented programming language. This program could accept visual entries (clicking on objects to visually change their properties) or just plain code to describe objects. The GameMaker program was used to program the task environment itself (what the participants would see), as well as the graphical adaptive advice.

The server program was programmed in C#, using visual studio 2005. The server accounted for the trust models, the calculations behind the model and the logging of the whole task environment. The server and task communicated with messages using TCP/IP.

Initially, the models had to be tuned for each participant using personal training data. This required a fast machine to do this in reasonable time. The task also needs many resources to run properly. Therefore two machines were needed to run the experiment. However, after pilot testing the models were simplified and the parameters were set using pilot data. In the experiment the parameters were kept constant. This made the server much less complex, so the server and task could be run on a single computer.

The task was initially too complicated to be run properly on one computer. Since the refresh rate of the task (approximately 100hz.) would vary depending on the background processes of the computer, results varied (the speed of the contacts would vary), leading to different results each run. The problem has been tackled by lowering the frame rate to 50hz. (a normal computer could easily maintain this frame rate) and doubling the speed (so the total scenario time will stay similar in half the refresh rate).

In the end the server and task run stable and identical each time on a single computer. This guaranteed the data between subjects could be compared.

6.8.2 Noise Function

The noise function was one of the hardest problems to solve during the experiment set up. The task itself is quite dynamic due to the scenarios. This means also the desired threatening classification is dynamic, so is the advice. However, the advice system should be programmed to make mistakes. Making random mistakes on an already heavily varying system advice was a problem. The system must make mistakes, and when the advice is recalculated, it should make the same mistakes with a higher probability. If this was not the case, and the system was to make for example 3 mistakes (out of 5), then in the next step it could randomly have calculated three other mistakes. This would lead to too many advice changes in a single step. The participant would be unable to keep up with the changes, making the advice useless for him.

The problem would also be the participant not being able to decide whether to follow or not follow the systems advice, since the advice will already have been

```

1 public void CalculateSystemThreatValues ()
2 {
3     if (!CheckSystemReliability () && (Environment.TickCount >
4         lasttimesystemthreatvaluesrecalculated + 5000))
5     {
6         foreach (Contact contact in contacts)
7             for (int i = 0; i < contact.systemthreatvalues.
8                 Length; i++)
9                 contact.oldsystemthreatvalues[i] = contact.
10                    systemthreatvalues[i];
11
12         CalculateSystemThreatValues(0);
13         lasttimesystemthreatvaluesrecalculated = Environment.
14             TickCount;
15         systemthreatvalueschanged = 1;
16     }
17     else
18         systemthreatvalueschanged = 0;
19 }

```

./img/Code/SystemNoiseCalculation.txt

Figure 15: Control method for noise calculation. In the C# language.

changed when he wants to make a decision. So the descriptive trust measure will also be inaccurate.

The system makes mistakes by adding noise to the calculated threat values. The noise needs to be small enough to only change the lowest threats with a higher probability than the higher threats. This means the contacts near the border of threatening/ not threatening are most likely to be changed.

Furthermore the amount of mistakes have to be controlled, where a 70% system reliability was desired. The number of mistakes were made dependent on the system difficulty (as explained in section 6.2.2). When more mistakes were expected from the system, the whole noise addition calculation would be repeated until a threat distribution was formed with the desired mistakes.

A way to keep the advice from changing all the time is by the control algorithm in figure 15. The method in this figure shows that the server will check during every update whether it is necessary to update the system threat values. It could be the case the threat values are still correct. When the System Reliability check fails (the desired number of mistakes is not equal to the current mistakes) the system threat values will be recalculated.

An extra measure of keeping the advice steady is by controlling the minimal time between system threat value updates. Only when a certain time has past between the last update and the current time, the noise will be recalculated. In this specific case this is 5000ms.

The steadiness of the advice will give the participant time to reason with the advice. However, this also means the model will make more mistakes, since the situation might change rapidly, the model will still keep the previous given advice.

```

1  if(global.SupportType == 3)
   {
3    if( curContact > 0 )
   {
5      //under trust
      if(global.TrustLevel < global.LowerTrustThreshold)
7        {
           if((curContact.status == 'onbekend' && curContact.
11          isadvice == 1) ||
           (curContact.status == 'bedreigend' && curContact
13          .isadvice == 0))
           {
               if((curContact.threatsealanezscore > global.
15          idcritupperlimit)) {idcritinfo += '-
               Vaarroute#';}
               if((curContact.threatspeedzscore > global.
17          idcritupperlimit)) {idcritinfo += '-
               Snelheid#';}
               if((curContact.threatheadingzscore > global.
19          idcritupperlimit)) {idcritinfo += '-
               Richting#';}
               if((curContact.threatdistancezscore > global.
21          idcritupperlimit)) {idcritinfo += '- Afstand
               #';}
           }
           if((curContact.status == 'bedreigend' && curContact.
23          isadvice == 1) ||
           (curContact.status == 'onbekend' && curContact.
           isadvice == 0))
           {
               if((curContact.threatsealanezscore < global.
25          idcritlowerlimit)) {idcritinfo += '-
               Vaarroute#';}
               if((curContact.threatspeedzscore < global.
           idcritlowerlimit)) {idcritinfo += '-
               Snelheid#';}
               if((curContact.threatheadingzscore < global.
           idcritlowerlimit)) {idcritinfo += '-
               Richting#';}
               if((curContact.threatdistancezscore < global.
           idcritlowerlimit)) {idcritinfo += '- Afstand
           #';}
           }
           }
       }
   }
27 }

```

./img/Code/ArgumentationImplementation.txt

Figure 16: Example argumentation support, only for under trust arguments. In the GameMaker language.

```

1  if(global.amountofadvice < 1.0)
   {
3     position1 = ds_list_find_index(tobemarked, obj);
     position2 = ds_list_find_index(tobeunmarked, obj);
5     size1 = ds_list_size(tobemarked);
     size2 = ds_list_size(tobeunmarked);
7     test = 0;

9     //under trust
     if(global.TrustLevel < global.LowerTrustThreshold)
11    {
        //censor most obvious contacts
13        if((position1 > numbertobemarked || position2 >
            numbertobeunmarked))
15        {
            obj.isadvice = 0;
            obj.iscensored = 1;
17        }
19    }
   }

```

./img/Code/CensorshipImplementation.txt

Figure 17: Example censorship support, only for under trust arguments. In the GameMaker language.

6.8.3 Argument Strategy Implementation

The argumentation strategy depends on arguments which will be given during under- or overtrust. To ensure the arguments make sense to the user, different threatening levels have been determined. An argument could be given while the system is certain a contact is not threatening, or an argument why he is certain a contact is threatening. In between there is an interval where the system could doubt this argument leads to either threatening or not threatening.

In the experiment the z-scores of contact threat levels (for each idcrit separately) will be used to determine how much influence the criterium has in the threat determination. When a z-score of an idcrit is high, it makes the contact threatening compared to the other contacts (as explained in section 6.6.2).

With the use of pilot data, thresholds were determined. Where lower than the lower limit means the idcrit makes the contact certainly not threatening, higher than the upper limit means certainly threatening and between the lower and upper threshold means the idcrit leads to uncertainty.

In figure 16 an example segment of the argument implementation can be found. Where "curContact" is the currently selected contact (where the participant hovered over with his mouse). Two advice cases will be distinguished: certainly threatening, certainly not threatening. Where in both cases the right arguments will be returned. Similar to this mechanism is the over trust, where all idcrits between the upper and lower boundary will be given as argument (uncertainty).

6.8.4 Censorship Implementation

The censorship strategy concerns the advised (the highlighted) contacts themselves. During over- or undertrust, an amount of the advice will be censored, so the participant will not see all desired modifications from the system.

Censoring items can be done during the update of each contact in the task environment. For each contact its threat rank (is the advice threatening or not threatening) is known. From this list an ordered list can be constructed (together with the user his threat classification) which contacts should be highlighted and which not.

During under- or overtrust a percentage of this advice should not be shown to the participant. In figure 17 a fragment of the implementation is shown. the *global.amountofadvice* is the ratio of censored contacts (1 means show all advices, 0 means censor all advices). The to be marked and to be unmarked lists are the contacts which should be advised by the system. However, a calculated ratio (all until the *position1* marker) will be censored and not be shown as advice to the user.

The order of the *tobemarked* list guarantees the censored contacts will be the least obvious threats (or non threats), leaving only the obvious advices visible to the participant. The contacts will stay censored until the next model update.

6.9 Procedure

The experiment procedure is given in table 5. On several occasions during the experiment, different questionnaires were given to the participant. The different questionnaires were: the start questionnaire, a questionnaire after each condition (independent on support type), questionnaires dependent on the support type and an end questionnaire. All questionnaires can be found in appendix A on page 81.

Written tests were used for training purposes, to get acquainted with the task and with the different support types. These tests can be found in appendix C on page 96.

6.10 Materials

The questionnaires were given on paper. Furthermore the experiment was run on a single computer. Important requirements for the computer were the screen, which was large enough (the task should be visible without having been cropped) and the computer was fast enough. The task was set to have a refresh rate of 50 Hz, when the computer could not keep this refresh rate constant, faulty results would have been outputted (more information on the implementation of the task in section 6.8.1).

6.11 Data Analyzes

The data analyzes are divided into two sections. First part of the analysis consists of model validation. In this section the different trust models analyzes are explained. Correlations were searched using standard (simple) regression tests. These regression tests are visualized using scatter plots.

The second part consists the support evaluation analyzes. Different analyzes comparing the conditions are explained in this section. The most important

Activity	Duration (min)		
Begin experiment	10		
Start questionnaire	5		
Experiment description	15		
Understanding test idcrits	5		
Understanding test static support	5		
Task practise session (computer)	10		
Questionnaire condition independent	5		
Understanding test dependent on condition	75	5	repeated 3 times
Condition SS, AS or CS		10	
Questionnaire condition independent		5	
Questionnaire dependent on the condition		5	
End questionnaire	10		
Experiment end	5		
Total duration	145		

Table 5: Experiment time table, the exact procedure with their durations.

support evaluation analysis is the comparison of the performance data. Furthermore differences in trust appropriateness are tested. The different variables were averaged per participant per condition. The generated data could then be analyzed using repeated measure ANOVA tests. Any found differences were further analyzed using a Bonferroni tests.

Both sections are used to answer the main research question. A more detailed description of all separate data analyzes are given described below.

6.11.1 Model Validation

Three different models were analyzed: the descriptive model, the prescriptive model and the appropriateness model (as explained in section 6.3).

The descriptive model is the model that estimates the human trust level in the system. Hypothesis 1 was constructed to test this model. It stated that the descriptive model should be able to predict the real human trust level in the system. The hypothesis can be confirmed when a correlation between descriptive trust and real human trust is found. The human trust was measured using two different types of data: in-task and out-task human trust data. The in-task human trust data is based on the human trust feedback measure. As explained in section 6.7.2, the human trust feedback is a measure of human trust in the system, by asking the participant's opinion during the task.

The second measure is the out-task human trust measure. This data is extracted from the given questionnaires (explained in section 6.7.4) at the end of each condition. In the questionnaire the participant was asked in what degree he agreed with the statement: "I trusted the system was giving me correct advice". If the participant trusted the system at a higher degree during the set, it is assumed that he agrees stronger with the statement. The found human trust data (both in-task and out-task) can be compared to the descriptive model data. A correlation between the model output and the human trust data means

the descriptive model was a good predictor for human trust, confirming the hypothesis.

The prescriptive trust was used to estimate the optimal trust level of an agent in the system. Hypothesis 2 can be confirmed when a correlation is found between the prescriptive model values and the system's performance, since the optimal measure of prescriptive trust is the system's performance. Therefore, the prescriptive trust model was validated using the system's performance data. (the performance measure is explained in section 6.7.1).

The outcome of the prescriptive trust model was based on the task environment only at that time during the task. Different scenario parts were presented to the participant during each condition. As an extra check to see if the scenario parts influenced the prescriptive model performance, the performance of the prescriptive model was calculated for each scenario part. This way differences in model performance between scenario parts can be distinguished.

The appropriateness model used the descriptive and prescriptive trust as input. The comparison of these two model outputs was the appropriateness value (as described in section 6.3.3). As explained above, the descriptive and prescriptive models were validated using the above described validation data. Using the given validation data instead of the descriptive and prescriptive model data, an optimal trust appropriateness value could be calculated, using the appropriateness model similar as in the experiment. For the descriptive data the in-task human trust feedback was used and for the prescriptive data the actual system's performance data was used. The appropriateness model functioned as expected when the appropriateness model output is correlated to the optimal appropriateness model output. Note that this is just the appropriateness *value*, which is used to determine the trust appropriateness *state*.

The appropriateness state is the final output of the appropriateness model. Hypothesis 3 states that the appropriateness state which is calculated by the model should be a predictor for the optimal trust appropriateness state. To test the appropriateness model, an optimal trust appropriateness is needed to be calculated.

First recall that the outcome of the trust appropriateness determination is either undertrust, overtrust or appropriate trust. The determination is done using thresholds: when, for example, the appropriateness value is below the lower threshold, undertrust will be the given state (as described in section 2.2.3). The determination of the thresholds is a parameter which can be set based on the desired cautiousness. When the model was set to be very cautious (meaning determining under- or overtrust is dangerous, mistakes have greater negative implications), the thresholds can be set closer to the 0. The further away the thresholds are from 0, the less cautious the model is in determining either under or overtrust.

To validate the trust appropriateness states, validation data which is dependent on the cautiousness factor was used. The validation data is based the trust appropriateness ratio (described in section 6.7.3), with different penalties for different situations. A situation can be for example returning overtrust, while appropriate trust was the optimal state for the largest ratio of contacts. Each situation can be set with a different penalty. If returning overtrust while appropriate trust is optimal is more dangerous than vice versa, the penalty for this situation should be higher. When in this example the ratio of overtrust is high, it will be penalized harder, because appropriate trust was optimal and the

penalty of returning overtrust was higher. This way returning overtrust is more dangerous, and will be the optimal case in less cases.

Using different penalties for different states, the cautiousness of the model can be regulated. Using the penalties, the optimal appropriateness state can be calculated using the trust appropriateness ratio's as follows: The ratio's can be used to determine the total penalty for each possible state, where the state with the lowest penalty is the optimal state (the total penalty of the undertrust state is the ratio of appropriate trust and overtrust times their penalty). If for example all situations are penalized similar and in the current situation most contact states lead to undertrust, the optimal state is undertrust. The resulting optimal states can then be used to validate the model's appropriateness state.

The penalties of the appropriateness situations (i.e. return overtrust while appropriate trust is optimal), represent the cautiousness of the model. The penalties are set in such a way that they fit the appropriateness model data best (so the difference between the model output and optimal state determination is minimal). Using this data can be determined how cautious the model was, given the penalties. The model's performance can be analyzed, using the comparison between the given model output state and the optimal state (a total percentage of correct state determinations can be calculated). When the model performed better than random, hypothesis 3 can be accepted.

6.11.2 Support Evaluation

The support was analyzed using the performance of the participant (using the support types). This analysis was used to test both hypotheses 4 and 5, which state that the argued and censored advice lead to an improved team performance. How the performance was calculated is described in section 6.7.1. When differences in performance was found between the static task support condition and the adaptive support conditions, the hypotheses can be confirmed.

Possible found performance improvements could be explained by the trust appropriateness behavior or experience of the user in the system (as explained in section 4). To explain any found performance differences by these factors, they were analyzed.

As explained in section 4, the adaptive support types could lead to performance improvements because the user trusts the system more appropriately. To check whether the user trusts the system more appropriately during the adaptive support (as compared to static support), the trust appropriateness was measured. The trust appropriateness behavior of the participants was measured using the trust appropriateness ratio (explained in section 6.7.3) and was used to evaluate if the trust appropriateness ratio different during the different support types.

The adaptive support types could also have led to performance improvements because the user can be aware of how to appropriately trust the system (section 4). This effect can be checked by the trust appropriateness experience of the participants in the system. The trust appropriateness experience is calculated by comparing the reported human trust feedback (as described in section 6.7.2) and the rational trust in the system (by means of the system's performance, explained in section 6.7.1). When differences are smaller in a condition, the participants have reported their trust to be closer to the optimal trust level, meaning their trust appropriateness experience was better.

Possible differences between the support types could be elaborated by the participants themselves. Therefore using questionnaires (explained in section 6.7.4), several useful details were asked, like support preference, comments on the support types and comments on the experiment itself.

7 Results

The results are divided into two parts, first the experiment validation results are given. In this section the influences of the support system are analyzed. Secondly the results of the trust model evaluation are given.

7.1 Model Validation

In the experiment, the adaptive support was based on different trust models (explained in section 6.3). The performance of these models (descriptive trust model, prescriptive trust model and trust appropriateness model) are validated in the sections below. The types of analyzes are explained in section 6.11.

7.1.1 Descriptive Model (In-Task)

The output of the descriptive model should have been an indication of how the system is being trusted by the user. A comparison was made between the output of the descriptive model and the objective trust (section 6.7.2). The in-task human trust was what the participants reported as their trust in the system, at several moments during the task. The measurements were averaged for each participant, per set. A correlation means the model worked as expected. The results are given in figure 18.

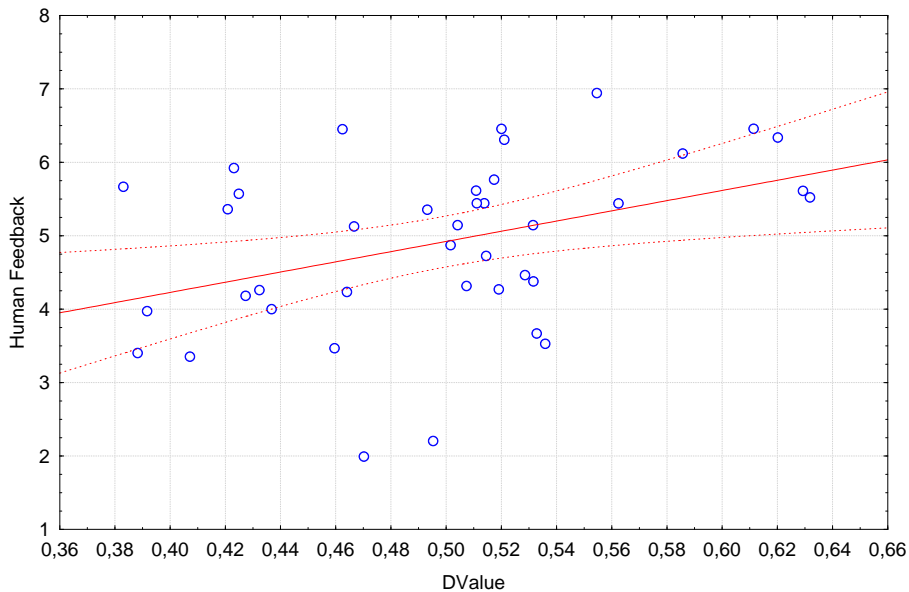


Figure 18: Scatter plot on the Descriptive Model output (DValue) being a predictor for Human Trust (by Human Trust Feedback).

The descriptive model significantly predicted human trust, $b = 0.39, t(40) = 2.62, p = 0.01$. The descriptive model explained a significant proportion of variance in human trust, $R^2 = 0.13, p = 0.01$. This means hypothesis 1 is accepted.

7.1.2 Descriptive Model (Out-Task)

The descriptive model was also evaluated using out-task human trust data. The human trust in the system was asked one time after each condition. The human trust data of the participant after the conditions was compared to the average descriptive value during the condition. In figure 19 the results are presented.

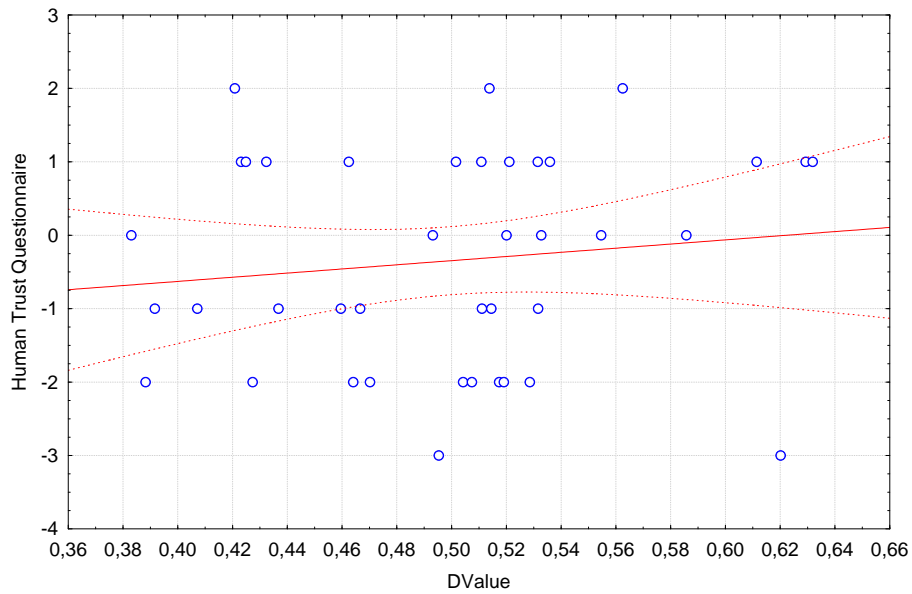


Figure 19: Scatter plot on the Descriptive Model output (DValue) being a predictor for the Human Trust (by Trust Question in the Questionnaire).

No correlation was found between the descriptive model output and the answers of human trust in the questionnaires ($b = 0.13, t(40) = 0.80, p = 0.43$). Therefore, using the out-task human trust data, hypothesis 1 is accepted.

7.1.3 Prescriptive Model

The prescriptive model was analyzed using the system's performance data. The averaged prescriptive model output (per participant, per condition) was compared to the average performance of the system during the participant's condition. The results can be found in figure 20.

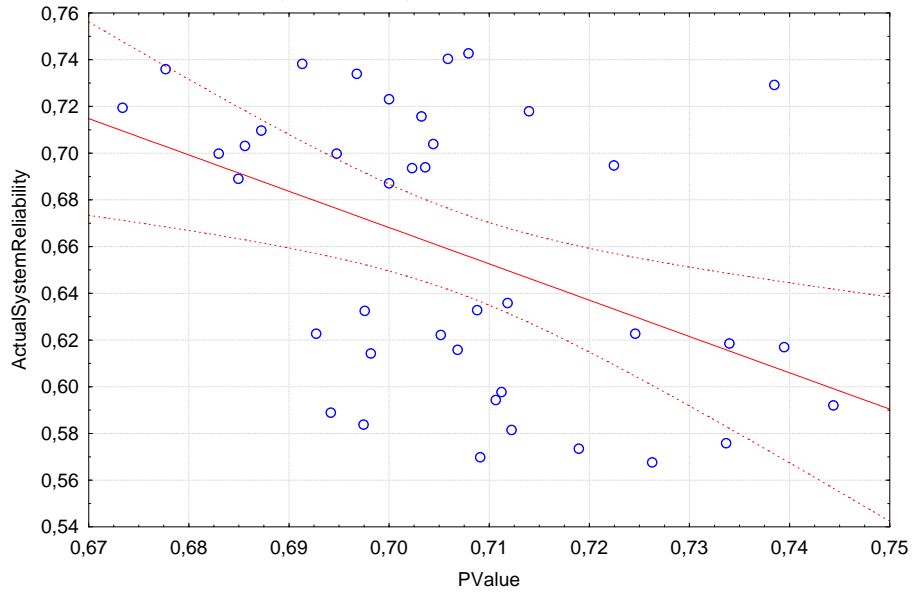


Figure 20: Scatter plot on the Prescriptive Model output (PValue) being a predictor for the Actual System Performance (by ActualSystemReliability).

The prescriptive model significantly predicted actual system performance, $b = -0.44$, $t(40) = -3.04$, $p = 0.004$. The prescriptive model explained a significant proportion of variance in actual system performance, $R^2 = 0.17$, $p = 0.004$. A problem is that the found correlation is negative. Therefore, based on the results hypothesis 2 is not accepted, since a positive correlation was expected.

7.1.4 Prescriptive Model per Scenario Part

The prescriptive model was only dependent on the task environment (the scenario's), not on the participant's behavior. A determination of the performance of the prescriptive model was used to check whether the prescriptive model was more reliable at different scenario parts. At each time point, an error was calculated of the prescriptive model output, compared to the actual system reliability. This distance was averaged over the set per participant. The averages of the different scenario's were then compared. Results can be found in figure 21.

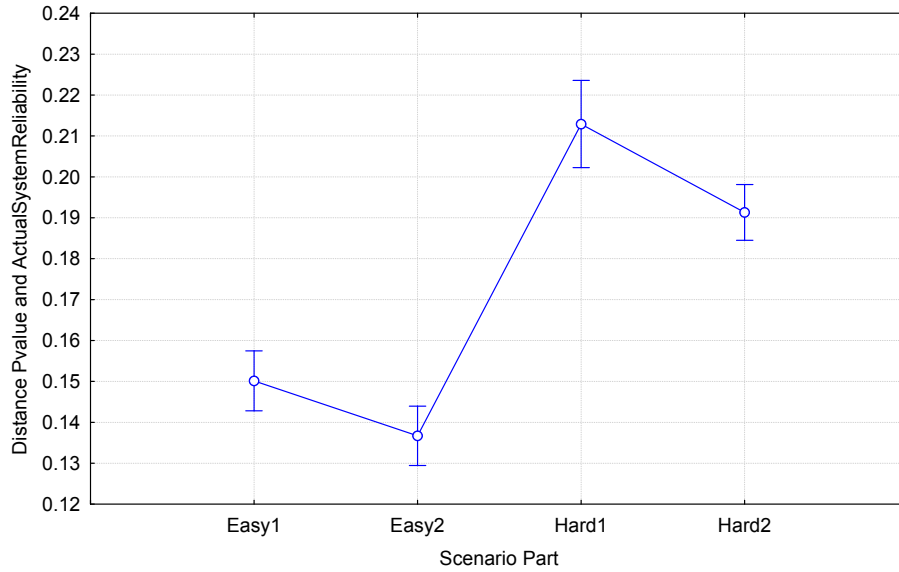


Figure 21: Accuracy prescriptive model per Scenario Part.

Scenario Part	Mean	SD
Easy1	0.15 ^A	0.003
Easy2	0.14 ^A	0.003
Hard1	0.21	0.005
Hard2	0.19	0.003

Table 6: Prescriptive Model accuracy for different scenario parts. The means with the same letter in their superscript did not differ significantly according to a Bonferroni test (with $\alpha = 0.01$).

Different prescriptive trust performance indications have been found between the different scenario parts. A Bonferroni test (undirected) shows differences between all sets, only a difference between easy set 1 and easy set 2 was not found (table 6).

The differences mean the prescriptive model was better in estimating system performance in certain scenario parts. In the easy scenario parts the prescriptive trust estimation was significantly better than in the hard scenario parts.

7.1.5 Appropriateness Value

The appropriateness value was the comparison of the descriptive and the prescriptive model values. The appropriateness values were compared to the optimal appropriateness values, which were constructed of comparing the optimal descriptive model values (by the human trust feedback, described in section 6.7.2) and the optimal prescriptive values (actual system performance, section 6.7.1). The averages were compared per set per participant. The results of the comparison can be found in figure 22.

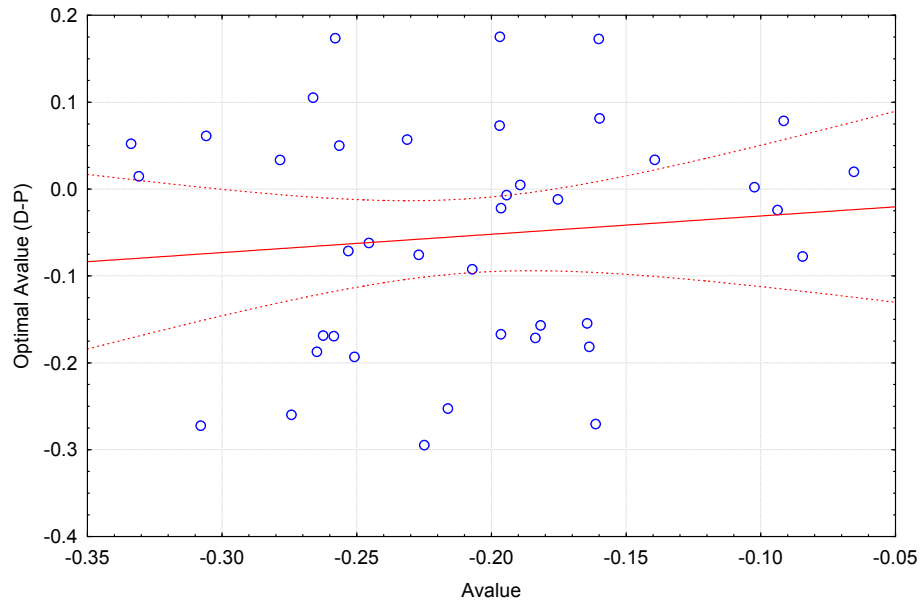


Figure 22: Scatter plot on the Appropriateness Model output (AValue) being a predictor for the Objective Appropriateness (by Objective AValue).

No correlation were found between the appropriateness model and the optimal trust appropriateness ($b = 0.16, t(40) = 0.66, p = 0.51$).

7.1.6 Appropriateness State

The trust appropriateness state (the output of the appropriateness model) was validated using the optimal trust appropriateness state. The optimal trust appropriateness state was calculated using the trust appropriateness ratio and penalties for different trust appropriateness situations, taking the cautiousness factor of the model into account. How the optimal trust appropriateness state was calculated and why penalties were used in this calculation is described in section 6.11.1.

The penalties were based on the trust appropriateness ratio, where for each situation the proper penalty was determined. In table 7 the determined penalties are given which have been used to generate the optimal appropriateness states. The penalties were set such that the distance between the optimal trust appropriateness state and the output of the appropriateness model is least, meaning the penalties describe the obtained model data optimal. In this way the penalties in the penalty table represent the cautiousness of the model.

Model Output	Case	Penalty
Appropriate Trust	Under / Over Trust	1
Under Trust	Appropriate Trust	0.8
Over Trust	Appropriate Trust	0.2
Under / Over Trust	Over / Under Trust	1.25

Table 7: Penalty table for the different possible appropriateness situations.

The differences in penalties indicate the model was not cautious its under and over trust verdicts, since these penalties are much lower than vice versa (what means wrong under- or overtrust determination was not penalized as hard as vice versa).

The penalties of undertrust while appropriate trust and overtrust while appropriate trust are also different. This means the model was less cautious in its overtrust verdict than the undertrust verdict.

Using the above given penalty table, the optimal trust appropriateness state was calculated in each time step of the experiment. The optimal states were compared to the actual appropriateness model outputs. In table 8 the results are presented.

ModelOutput	Optimal		
	Under Trust	Appropriate Trust	Over Trust
Under Trust	16.56	13.02	1.40
Appropriate Trust	18.14	23.82	2.45
Over Trust	8.85	14.10	1.65

Table 8: Differences between Optimal Trust Appropriateness and the Appropriateness model output.

The total percentage correct was 42.04%, where a random distribution with three possible states would lead to a percentage correct of 33% on average. Since the model performs better than random, hypothesis 3 is accepted.

Given the model's thresholds, the trust model was set to output under- and over trust in 25% of all cases (recalling section 6.3.3). The data in table 8 shows that the model returned undertrust in 30.98% and overtrust in 24.60% of all cases.

However, optimal trust appropriateness was divided differently. The optimal undertrust rate was 33.55% and the overtrust rate was only 5.50%. Given the optimal trust appropriateness states, undertrust occurred far more than overtrust.

7.2 Support Evaluation

The results of the support type analyzes are presented in the following sections. Furthermore feedback of the participant are presented. The feedback is based on comments of the participants on the experiment and their answers to the questionnaires. A detailed description of the analyzes can be found in section 6.11.2.

7.2.1 Performance

Performance was calculated for the Argumentation Support (AS), the Censorship Support (CS) and Static Support (SS) condition. For each condition, the average performance per participant was used. In figure 23 the results are presented.

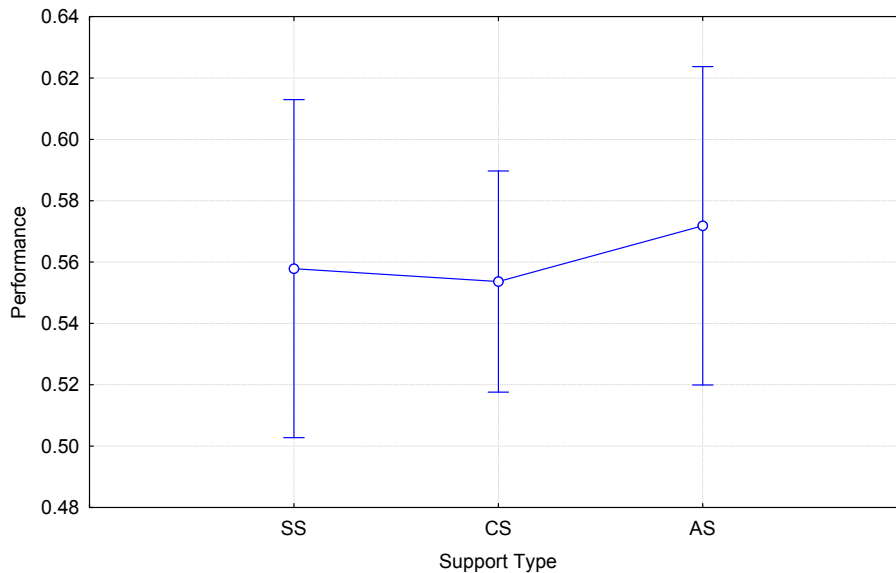


Figure 23: Performance plot per condition.

The average performance of the all participants together for all conditions combined was 58% correct (SD 7%). The average performance of the system was 66% correct (SD 17%). As explained in section 6.2.2, the desired system performance was 70%, this was almost reached, given the average and the large standard deviation.

No significant performance differences were found in the data between support types ($F(2, 18) = 0.37055$, $p = 0.69550$). Therefore hypotheses 4 and 5 are not accepted.

7.2.2 Trust Appropriateness Behavior

Possible performance differences could be explained by participants trusting the system more appropriately. As explained in section 6.11.2, trust appropriateness behavior was measured using the trust appropriateness ratio measure. The trust appropriateness ratio's were averaged for all participants for each condition. The resulting comparisons can be found in figures 24, 25 and 26.

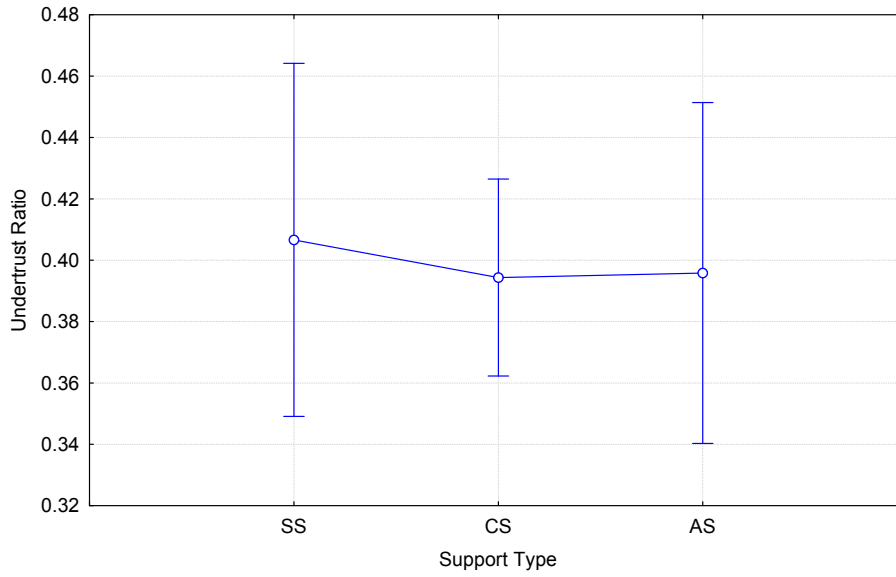


Figure 24: Undertrust ratio's per support type.

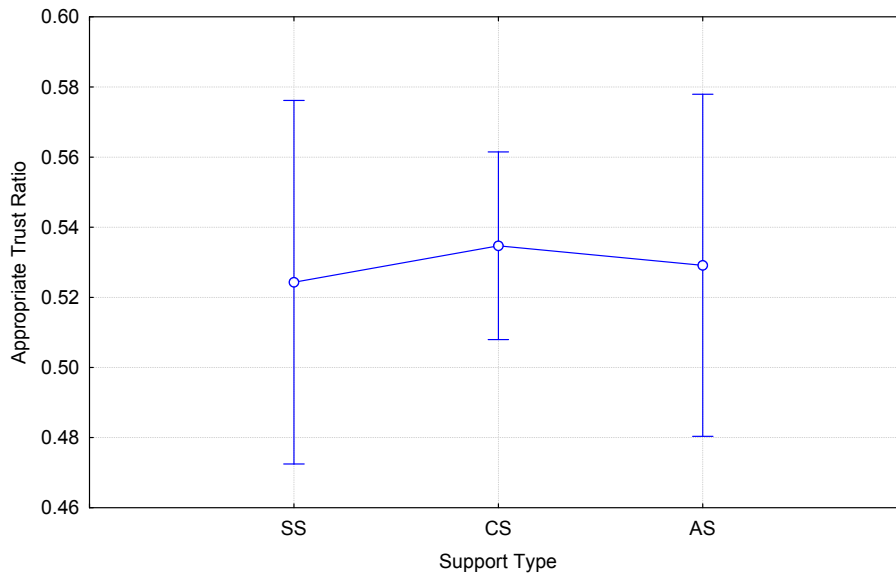


Figure 25: Appropriate trust ratio's per support type.

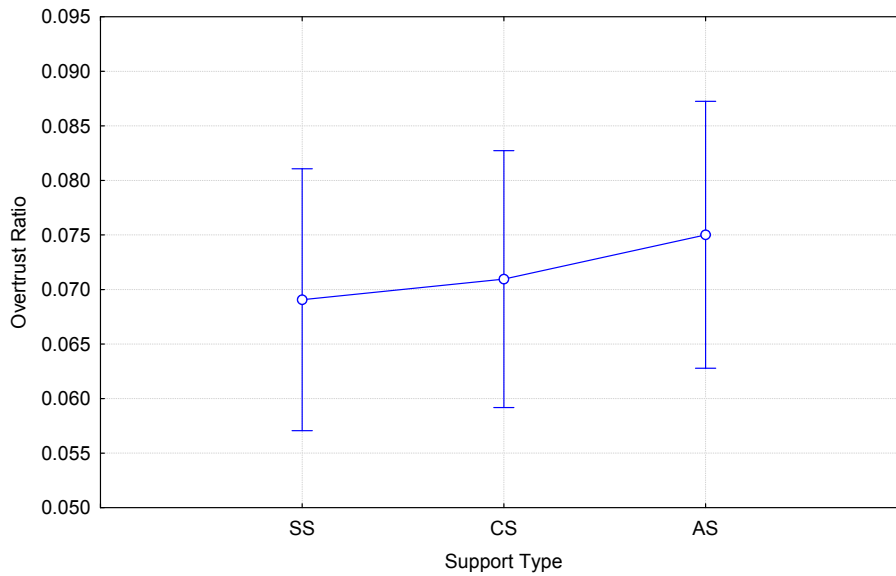


Figure 26: Overtrust ratio's per support type.

No differences were found between the undertrust ratio ($F(2, 18) = 0.13, p = 0.87$), appropriate trust ratio ($F(2, 18) = 0.13, p = 0.88$) and overtrust ratio ($F(2, 18) = 0.28, p = 0.76$) between the support types. If there were any performance improvements, they could not be explained by the measured trust appropriateness (user's behavior).

7.2.3 Trust Appropriateness Experiences

The trust experience was the human trust (Human Trust Feedback) which was reported by the participants, compared to the rational trust (system's performance). The trust appropriateness experience was calculated by scaling both trust values similarly (ranging from 0 to 1) and calculating the descriptive minus the rational value. The resulting values were averaged per participant per condition. For the different conditions the values were compared. Average values closer to 0 means the trust has been more appropriate. In figure 27 the results are given.

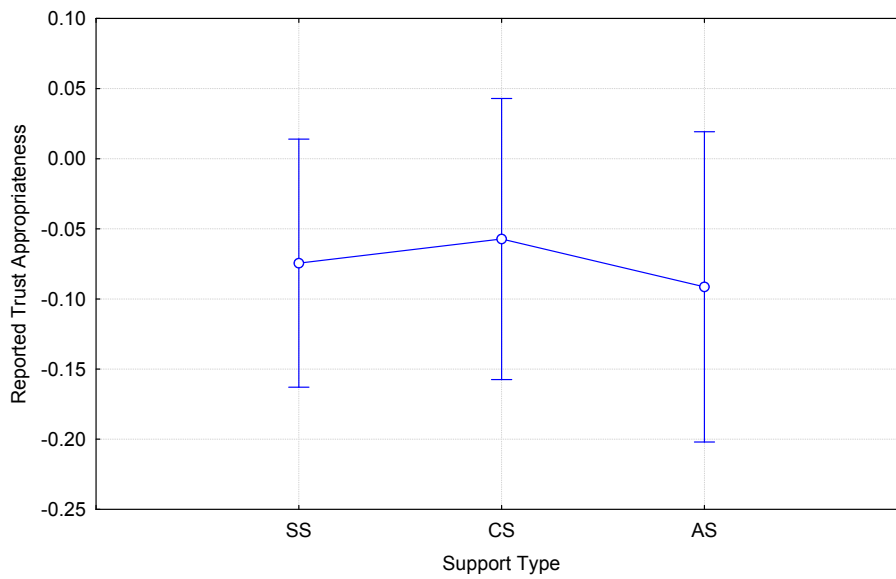


Figure 27: Optimal Trust Appropriateness per condition.

The results shows no differences in trust appropriateness experiences were found ($F(2, 18) = 0.23, p = 0.80$) in the different experiment conditions. This means the participants did not report any different trust appropriateness between the different conditions.

7.2.4 Results Feedback Participants

When the participant finished the experiment, he was asked several questions. These questions were designed to give them the ability of commenting on the conditions and the experiment in general. Many sensible comments were made, therefore they are reported in the results of the experiment (table 9).

The questions were whether the adaptive strategy was useful to them (explained in section 9a), and which adaptive strategy they preferred (explained in section 9b).

Strategy	Comments	
	Positive	Negative
AS	<ul style="list-style-type: none"> - No "black box" advice - Knowledge on why advice system was wrong - Knowledge of how advice was calculated - Helped making final decisions 	<ul style="list-style-type: none"> - Hard to analyze advice - Text on side display was distracting - The advice system often made mistakes
CS	<ul style="list-style-type: none"> - Less intrusive advice, more time to reason by yourself - Calmer advice - Clearer advice 	<ul style="list-style-type: none"> - No difference from SS condition noticed

(a) Comments on Adaptive Strategies.

Strategy	Votes	Comments
AS	4	- With false system advice, argumentation helps
CS	3	- Calmest support, lead to highest performance
SS	1	- Calmest support, kept my concentration optimal
AS + CS	1	- Censored argumentation would be best

(b) Preferred Support System.

Table 9: Qualitative results, common answers from the participants. Differences for the three different conditions: Static Support (SS), Censorship Support (CS) and Argumentation Support (AS).

Finally the participants were asked for comments on the task or experiment in general. Most participants said the task was clear and interesting to them.

Other comments concerning the task were that they found it very difficult to execute the task by only following the specific rules (idcrits), without considering their own criteria (for example "a contact is threatening because it *could* turn around and face your own ship at any time", or "a contact that moves in a circle or zigzag motion is more likely to be a threat"). When these criteria were used to determine system threatening, they would be punished by the system because they did not follow the given rules. Participants argued that because of this task imagination, the task was often hard to execute.

Furthermore participants commented on the support system. The support system seemed very precarious, performing well at one time and bad at the next. This led to systematic undertrust from some participants in the system.

8 Conclusion

The main research question of this thesis was "Can team performance of a human-computer team be improved by changing the system's advice strategies adaptively, based on trust appropriateness of the human in the support system?". In order to answer this research question, a trust based adaptive support system was implemented on a task environment and used in an experiment. The system was constructed to measure trust appropriateness of the user in its advice with the use of trust models. The trust appropriateness was then used to adaptively change the system's advice, based on an argumentation or a censorship strategy.

The performance of the system was separated in two sections. First the way the system performed in measuring trust of the user in its advice was analyzed. The second section is concerned with the adaptive decision support and the possible influence of the adaptive support on the performance of the human-computer team.

To determine how the model performed in measuring human trust, the performance of the different trust models have been analyzed: the descriptive model, prescriptive model and the trust appropriateness model. The conclusions concerning the trust models are given below.

The descriptive model was correlated to the human trust feedback. Hypothesis 1 was accepted. The conclusion is that the descriptive model was able to predict the human trust in the system.

The prescriptive model was tested by comparing the model output to the performance of the system. The comparison showed a negative correlation between the two measurements. However, a positive correlation was expected, therefore, hypothesis 2 was not accepted. The conclusion is that the prescriptive model did not function properly.

The appropriateness model was used to determine the right trust appropriateness state. The determined states were compared to the optimal trust appropriateness states. The data analysis showed that the model predicted the states in more situations than when states were chosen randomly. Therefore hypothesis 3 was accepted. The conclusion is that the trust appropriateness model was able to predict the actual trust appropriateness state properly in many situations.

The trust appropriateness state was used to adaptively change the system's advice by either argue or censor the given advice. The second section, and final step in answering the research question, is whether the adaptive advice lead to an improved performance of the human-computer team, in comparison with non-adaptive advice. An improved performance was expected for both the censorship support (hypothesis 4) and the argumentation support (hypothesis 5), compared to the static support. Both hypotheses were not accepted. The conclusion is that the addition of trust-based adaptive support has not lead to an improvement in the team performance.

The conclusions mean that some parts of determining the right adaptive decision support worked well, but the whole adaptive support system did not lead to an improved team performance. Therefore the main research question cannot be answered with "yes", based on this experiment.

9 Discussion

The research was divided into two parts, first the trust models were analyzed, the discussion of the trust models are presented in the model validation section. The second part was the evaluation of the different support types. These conclusions are discussed in the support evaluation section below. Using the experiment results, different task characteristics were especially suitable for the adaptive support types, in section 9.3 these characteristics are discussed.

9.1 Model Validation

The measurements of trust by the trust models were reliable in some aspects and were less accurate for others.

The descriptive model was able to predict the human trust levels in the system. This was confirmed by the in-task human trust data of the experiment. Only a small effect was found, but this could be caused by the large human trust feedback update intervals (leading to a small number of measurements). The feedback was only asked once every minute of the participant (described in section 6.7.2), while the descriptive trust model was updated once every five seconds (as described in section 6.8.2). The lack of human trust feedback data could explain the small correlation effect on the descriptive model data. The large intervals of the human feedback measurements were chosen deliberately, because according to pilot data, the participants were too often interrupted from their task.

Another consequence of the low frequency of human trust feedback measurements is that only the global effectiveness of the descriptive model was analyzed (averages of a whole set). The local effectiveness of the descriptive model could not be analyzed due to the lack of validation (human trust feedback) data, while local user's trust level fluctuations in the system may have occurred. Local effects might for example occur when advice is changed. To validate local effects, measuring human trust could be done by increasing the number of human feedback questions, but as explained before, increasing the measuring frequency led to more distraction of the participant from the task. Because the lack of the human trust feedback data, only the results of a whole set were considered in this experiment. Testing local descriptive trust effects could be done by an experiment designed to test this factor.

The out-task human trust data didn't confirm that the descriptive data was able to predict human trust. The out-task data was gathered using the questionnaires (as explained in section 6.11.1). While the descriptive data was updated every five seconds, the out-task human trust measurements were only done at the end of a condition, resulting in one measurement for each condition. This data might have been not enough to validate the descriptive data appropriately.

The prescriptive model was validated using the actual system reliability (the system's performance). The actual system reliability was calculated in each time step. However, the system threat values were only updated once every five time steps (as explained in section 6.8.2). Since the prescriptive model values only depend on the system threat values, the prescriptive model was also only updated once every five time steps. The real threat situation might have changed in these time span, therefore the prescriptive model might be less accurate. This mechanism of waiting five time steps before the system's threat

values were updated, was constructed to keep the advice steady for a certain time, giving the participant time to reason with the advice. This came at the expense of the accuracy of the trust models.

The found negative correlation between the prescriptive model output and the actual system reliability was not expected. The prescriptive model outputs were based on the noisy system threat values. Therefore, the negative effect could have been caused by the artificial noise function, which was built to force the system to make mistakes (explained in section 6.2). The noise function was constructed to make mistakes in a realistic way, leading to more system mistakes as the situation became more difficult. However, to enforce more mistakes of the system, the noise variance for each contact threat value was higher in situations where more mistakes were desired (recall section 6.2.2). The algorithm was constructed to enforce a specific number of mistakes. If this number was not reached using the noise, the system would recalculate noise repeatedly until a situation was calculated with that number of mistakes. The variable threat value variance was used to keep the number of recalculations at a minimum, because the calculations needed to be done in real time during the task. However, the variable threat variance might not only have helped to enforce more mistakes, it could also have led to a wider distribution of the contact threat values. This means that if the situation is more difficult, more mistakes are desired, the variance is greater, leading to a more scattered contact threat value distribution. In conclusion, a difficult situation could lead to calculated noisy system threat values that are more scattered. The model would use this greater distances to determine that the system was more certain of its classification (as described in section 6.3.1). This means a greater error could lead to higher certainty, which is the opposite of the desired prescriptive value calculation.

This undesired effect is an effect of this specific task implementation (based on the way noise is calculated). This does not imply the principles of the prescriptive model (using a self confidence as prescriptive trust indication) would not work in a different situation or different task implementation.

The appropriateness model outputs a comparison of the descriptive and prescriptive model (as explained in section 3.3). No correlation between the appropriateness model output and the measured trust appropriateness was found (results are given in section 7.1.5). This could be explained by the noisy outputs of the descriptive and prescriptive models. Especially the prescriptive model, which was not accurate enough to contribute to a correct trust appropriateness in all cases. Since the prescriptive model worked opposite as it was expected, and this effect was strong, could have resulted in an inaccuracy of the appropriateness model output.

The trust model being not correct in all cases was expected. Since the model is only an estimation of trust appropriateness it could make mistakes in certain cases, like it could in real life. One of the questions was knowing the model output was not always correct, could it still provide valuable adaptive support in most cases. The results show that keeping in mind certain trust cautiousness parameters, the model was still able to predict trust appropriateness in 42% of all cases. This seems not so much, but given the above noisy model outputs (even the trust appropriateness value was not accurate) it is still a substantial proportion. One of the problem were inaccuracy of the prescriptive model, and the fact that all trust models were only updated once every five measuring points. This resulted in inaccurate trust appropriateness values, from which

the trust appropriateness states were determined. If these factors in the trust models were improved, the model could have a even higher accuracy, estimating trust appropriateness correctly in more cases, possibly making the advice more useful.

9.2 Support Evaluation

The interventions of the system, in the form of the different support types, have not led to a significant improvement in the team performance. This could have been caused by the small number of participants that were used in the experiment. Due to the time span that was available for the experiment, only ten participants were tested. The results also shows quite a large standard deviation in the performance data between participants. This could have been caused by the complexity and the highly dynamic properties of the task. Therefore ten participants might have been too few to see any significant differences in performance. The results imply the main research question can not be answered positive after this research only.

Another factor might have been the model its of the trust appropriateness state, as discussed above. The small amount of correct state estimations means that the model gave unnecessary argumentation or censorship in many cases, or no intervention when it actually was necessary. When the accuracy of the trust models increases, support will be given when this is appropriate, possibly leading to a higher relevance of adaptive advice, resulting in an increase of computer-team performance.

Not only were there no differences in performance, also the participants have not *experienced* (reported) trusting the adaptive support types more appropriately. The results can be found in section 7.2.3. The lack of trust appropriate experiences can possibly be explained by the feedback analyzes of the participants, using the comments of the participants on the different support types (given in section 7.2.4). According to the votes of the participants, only four participants preferred the argumentation and three preferred censorship, so the participants were divided in their support preference. Furthermore most participants liked one support type, and were very negative about the opposite adaptive support type. This could be balanced out to no significant performance effects at all.

Aside from the trust appropriateness experiences participants had in the different support conditions, there were also not any measured trust appropriateness differences found (results shown in section 7.2.2). The participants were not appropriately relying on the system more often in the different conditions. This could have been a factor that influenced the performance of the participants using the support types, but no differences were found.

9.3 Support Preferences on Different Task Characteristics

Performance differences of human-computer teams could also depend on the task environment where it is used in. The results of the experiment along with the qualitative analysis bring forward interesting task properties, where certain adaptive support could be beneficial or not. In table 10 these properties are given, and they are further elaborated below.

Properties	Nonadaptive Advice	Censorship	Argumentation
Task Properties			
Routine Tasks		V	V
Longer Time Span		V	V
Quick Decision Task		V	
Great Decision Impact	V		V
Imaginative Task			V
Advice Properties			
Highly Dynamic Advice		V	
Less Dynamic Advice			V
High Quantity of Advice	V	V	
Complex Advice			V
Trustworthy Advice		V	V

Table 10: Task environment properties where different types of support are suitable for.

There are several task properties where trust is sensitive for. When a task is for example a *routine task*, the user is performing this task with great repetition. This means it is possible for the user to lose concentration or get blind for changes in the repetition pattern. In this case the support system might not be trusted appropriately anymore (i.e. systematic overtrust is likely to happen). Extra arguments can be useful to recalibrate the users trust in the system when this occurs. The user often has much knowledge of the situation in situation tasks, censorship can be used to filter out any irrelevant advice for the user in certain situations. Other purposes of the adaptive support strategies is to break the routine when the system calculates its necessary. This will keep the human-computer team performing optimal.

Similar principles hold for tasks with a *long time span*, to keep the user reasoning appropriately with the advice, it might be beneficial to change the way the advice is presented when the system measures it is not being trusted appropriately.

When the task is about making decisions in a *limited time span*, the user might need advice. This advice should be easy to obtain, since time is a factor. Therefore it would be useful to censor advice which is not correctly used by the user anyway. This could also recalibrate trust in the support system, so the advice is used in an optimal way. Argumentation might not be desirable, since this is even more support to process by the user.

When *impact* for the decisions to be made by the team is great, the user needs to know as much as possible about the systems reasoning. This means argumentation would be a good addition to the static support. However, the system might not be able to always determine the right trust state. When censorship is applied to the task and the determination is wrong, possible valuable advice is not presented to the user. When the decisions are for example a situation of life and death, all possible advice should be taken into account. In this case censorship is not desirable.

The TPC Task turned out to be quite *imaginative*. Experiment data (ta-

ble 9) showed that even though there were clear decision rules specified, the participants used their own interpretation. Since performance depended on the given rules, people using their own imagination performed bad. A problem with imaginative tasks is that people use their own interpretation instead of the system's (the system's interpretation could be right in some cases). When the rule sets of the user and the system differ from each other it might be crucial that the system shows more of its reasoning process. Therefore, argumentation is desired for such tasks.

One of the comments of the participants TPC Task is that the TPC Task was *highly dynamic*. Both the task environment and the advice changed rapidly during the conditions. The censorship in this situation was desirable, since it would limit the amount of advices, when the system was not properly trusted. This would leave valuable (but less) advice in these situations, making the advice easier to use. However, the already highly changeable advice was even more dynamic when argumentation was added. Therefore many participants did not like this type of support on this specific task environment. However, at moments where the scenario's were *less dynamic*, they would want to have arguments. So on less changeable advice, argumentation would be welcome.

When the advice is already quite *extensive*, there is often already too much information for the user to consider. Argumentation would even lead to more advice, which could lead to the user not using the advice at all (because selecting valuable information takes too long). Filtering this advice in certain situations could be useful in this type of tasks.

Complex advice often cares for a better explanation how this advice is being relied on by the user. Therefore argumentation is useful, keeping the trust in the advice more appropriate.

The *trustworthiness* of the system is an important factor in human-computer teams. Often people are biased in trusting computers. System errors are often penalized disproportional in comparison with the credits for system correct advices (this is an automation bias described in [7]). During the experiment, people tended to under-rely on the system far more than over-rely (this effect can be found in the trust appropriateness ratio's in section 7.2.2 and section 7.1.6). The results could have been the above described bias in effect. Crucial is to keep the trust of the user in the system at an optimal level. When the system is in most cases correct, it is important for the user to have high trust levels in the systems advice. Fluctuations in system trust should be corrected as soon as possible, both adaptive support types could be useful to recalibrate trust appropriateness.

10 Future Research

In this thesis, a trust-based adaptive decision support system was constructed, which could intervene when human trust in the system was not appropriate. In an experiment, no performance improvements have been found with the addition of adaptive support (compared to static support). However, many aspects of the experiment went well (e.g. descriptive model). Therefore the structures given in this thesis can be used as ideas for further research. The research in this thesis can be used as first step in designing a trust-based adaptive support system. Different improvements on the experiment are given below.

Improvements could be made on the trust models to increase their accuracy. Especially the prescriptive model and then of course the appropriateness model. With a higher accuracy, the advice will be given more appropriately for the real situation.

The task environment could also be improved, by for example choosing a completely different task. The comments of the participants (section 7.2.4) and property table 10 (page 76) can be helpful in determining which adaptive strategy will have the most effective on a task. The adaptive strategies could be used in various automation domains.

In this experiment, a basic form of argumentation has been used. The reason for this is that the advice itself was also basic. The advice was only in the form of the idcrits, which are only qualifiers and data in Toulmin's model of argument (figure 2 on page 12). However, when more complex advice is given, it could be beneficial to explore arguments deeper in the argument structure. When overtrust is measured it could for example give rebuttal argument backup. Or it could give extra backings to the original advice in certain situations. there are many possibilities to expand the advice arguments.

Censorship was also applied at a basic level. By pilot testing was determined that half of the advice needed to be censored to get any effect, without censoring too many advices. Much more research could be done on what the best way of censoring is.

Another possibility is to combine the adaptive strategies. As proposed by one of the participants (given in table 9b), it could be beneficial to have some form of adaptive censored argumentation. Using this strategy, given adaptively given arguments as extra advice could be censored in different situations.

Most research which have been done on this topic focussed on modeling trust itself and intervening at a basic level. Differences with the research in this thesis and past experiments, is that the system had access to the ground truth in most past experiments, so in the theoretical task environment the system always knew if it gave correct advice. It also had much reliance information (i.e. by asking the participant its reliance behavior after each action). One of the points of interests of this research have been that the model had no access to such information. The system's estimations were not correct in all situations. This experiment environment could be mapped to a realistic task, where the strategies were based on incomplete evidence. This research has been one of the first attempts to construct an adaptive decision support system, which could be used in such a realistic situation. This experiment was an attempt to improve decision support systems by using trust based adaptive advice. Only much more extensive research can be done before the usage of trust-based adaptive advice can be used to improve team performance in a real life situation.

List of Abbreviations

AS	Argumentation Support.
CAI	Cognitive Artificial Intelligence.
CR	Correct Rejection.
CS	Censorship Support.
DSS	Decision Support System.
FA	False Alarm.
H	Hit.
idcrit	identification criterium.
M	Miss.
SS	Static Support.
TPC Task	Tactical Picture Compilation Task.

References

- [1] A. Bisantz, J. Llinas, Y. Seong, R. Finger, and J.-Y. Jian. Empirical investigations of trust-related systems vulnerabilities in aided, adversarial decision making. 1998.
- [2] C. Castelfranchi and R. Falcone. Principles of trust for mas: cognitive anatomy, social importance, and quantification. In *Multi Agent Systems, 1998. Proceedings. International Conference on*, pages 72–79, July 1998.
- [3] C. Castelfranchi and R. Falcone. *The open Agent Society*, chapter Social Trust Theory. John Wiley & Sons, pres.
- [4] P. W. de Vries. *Trust in systems: effects of direct and indirect information*. PhD thesis, Technische Universiteit Eindhoven, 2004.
- [5] A. Heuvelink and F. Both. Boa: A cognitive tactical picture compilation agent. *Intelligent Agent Technology, IEEE / WIC / ACM International Conference on*, 0:175–181, 2007.
- [6] C. Jonker and J. Treur. Formal analysis of models for the dynamics of trust based on experiences. In F. Garijo and M. Boman, editors, *Multi-Agent System Engineering*, volume 1647 of *Lecture Notes in Computer Science*, pages 221–231. Springer Berlin / Heidelberg.
- [7] J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. 46(1):50–80, 2004.
- [8] T. Lucassen. Adaptive support of human attention allocation using cognitive models. Master’s thesis, University of Twente, 2008.
- [9] T. Lucassen. cognitive models of attention, internship report (TNO-DV S218), TNO Soesterberg Defense & Security. TNO Defense & Security, april 2008.
- [10] R. T. Nakatsu. Explanatory Power of Intelligent Systems: A Research Framework. In *Decision Support in an Uncertain and Complex World: The IFIP TC8/WG8.3 International Conference*, 2004.
- [11] R. Parasuraman and V. Riley. Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, 39(2):230–253, 1997.
- [12] C. Sierra and J. Debenham. An information-based model for trust. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS ’05, pages 497–504, New York, NY, USA, 2005. ACM.
- [13] P.-P. van Maanen. *Adaptive Support for Human-Computer Teams*. PhD thesis, Vrije Universiteit Amsterdam, 2010.
- [14] L. R. Ye and P. E. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Q.*, 19:157–172, June 1995.

A Questionnaires (Dutch)

Below the questionnaires, given to the participant during the experiment. The questionnaires were given at different moments during the experiment:

- Initial questionnaire
- Questionnaire condition independent (after each set)
- Questionnaire dependent on condition (after specific support type)
- End questionnaire

The questionnaires are presented below, in the above given order.

Vragenlijst

Beantwoord deze vragen voor de training

Algemene informatie

1. Naam _____

2. Leeftijd _____ jaar

3. Geslacht M / V

	mavo / vmbo	havo / vwo	mbo	hbo	wo
4. Hoogst behaalde opleidingsniveau	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. Aantal uren per week dat u gebruik maakt van een computer (ongeveer) _____ uur

6. Aantal uren per week dat u spelletjes speelt op een computer of game console (ongeveer) _____ uur

7. Voelt u zich onwel of ziek? Ja Nee

	heel slecht	slecht	redelijk	goed	heel goed
8. Heeft u goed geslapen vannacht?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Geef aan in welke mate deze stellingen betrekking op u hebben

	Helemaal Oneens	Oneens	Neutraal	Eens	Helemaal eens
9. Ik vind het niet erg om mijn eigen mening door te zetten binnen een groep.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
10. De ideale situatie voor mij is als mijn mening overeenkomt met andere mensen.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
11. Ik hecht meer waarde aan meningen van anderen dan aan mijn eigen mening.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
12. Ik zal snel iemand anders volgen, ook al verschilt zijn mening van mijn eigen mening.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Set nr. _____

Nummer. _____

Proefpersoonnr. _____

Vragenlijst

Beantwoord aan het einde van de set, geef aan in welke mate u het eens bent met deze stellingen over de afgelopen set.

Vragen over eigen prestatie

1. Mijn prestatie was hoog tijdens deze set.
2. Ik vond de set moeilijk.
3. Ik vond het moeilijk om geconcentreerd te blijven tijdens de taak.
4. De taak was dynamisch, ik heb dus vaak mijn antwoorden moeten aanpassen.
5. Ik twijfelde vaak aan mijn antwoorden.
6. De manier waarop ik tot mijn selectie moest komen, was voor mij duidelijk.
7. Mijn prestatie was hoger door de voorgaande oefeningen (training/voorgaande sets).

← Oneens	Neutraal				Eens →	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Vragen over prestatie systeem (de adviezen, dus dikgedrukte contacten)

8. Ik heb veel gebruik gemaakt van de ondersteuning.
9. De prestatie van het ondersteunende systeem was hoog tijdens de set.
10. Ik vertrouwde erop dat het systeem goed advies gaf.
11. Het adviessysteem hielp op momenten waar ik onzeker was.
12. Het advies was duidelijk.
13. Zonder advies zou mijn prestatie lager zijn.
14. Het advies was vaak erg verschillend van mijn eigen inschatting.

← Oneens	Neutraal				Eens →	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

15. De hoeveelheid advies was:

← Te weinig	Genoeg				Te veel →	
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Prestatie schatting

16. Schat uw eigen prestatie:
17. Schat de prestatie van het systeem:

Percentage correct										
0	10	20	30	40	50	60	70	80	90	100
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Set nr. _____

Nummer. _____

Proefpersoonnr. _____

Meer/Minder Advies

Tijdens de afgelopen set heeft u gebruik kunnen maken van ondersteuning. De manier van ondersteunen werd hierbij beïnvloed, door soms niet alle adviezen weer te geven (de dikgedrukte contacten). Beantwoord de vragen over deze manier van ondersteunen.

Vragen over ondersteuning

1. Ik heb gemerkt dat er geen volledig advies is gegeven.
2. Op het moment dat ik het advies nodig had, waren er vaak weinig contacten dikgedrukt.
3. Ik had het idee dat het systeem vaker goede adviezen aangaf.
4. Het systeem gaf voldoende advies om mij te helpen bij het maken van mijn keuzes.
5. Het systeem veranderde vaak van advies

	← Oneens		Neutraal			Eens →	
1.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Set nr. _____

Nummer. _____

Proefpersoonnr. _____

Argumentatie

Tijdens de afgelopen set heeft u op sommige momenten extra argumenten gekregen. De argumenten waren of het systeem zeker of onzeker was van het gegeven advies. Beantwoord de vragen over deze manier van ondersteuning.

Vragen over ondersteuning

1. Ik vond de extra argumenten nuttig.
2. De argumenten waren overtuigend.
3. Het was moeilijk om de taak uit te voeren terwijl er veel extra informatie werd aangeboden.
4. De manier waarop het systeem op zijn argumenten kwam, was voor mij duidelijk.
5. Ik had tijd genoeg om de argumenten te kunnen lezen en te gebruiken in mijn oordeel.
6. De argumenten waren correct.
7. Door de argumenten wist ik beter hoe het systeem aan zijn adviezen kwam.

	← Oneens		Neutraal			Eens →	
1.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
2.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
3.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
4.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
5.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
6.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
7.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Vragenlijst

Vul deze vragen in aan het eind van het experiment
De vragen gaan over het gehele experiment.

De taak was

1.

← Oninteressant	Neutraal	Interessant →
○ ○ ○	○	○ ○ ○

2.

← Moeilijk	Neutraal	Makkelijk →
○ ○ ○	○	○ ○ ○

Vragen over eigen prestatie in samenwerking met het advies systeem

3. Mijn prestatie was beter in de conditie waar argumenten werden gegeven.
4. Mijn prestatie was beter in de conditie waar censuur werd toegepast.
5. Ik heb me laten beïnvloeden door advies systeem tijdens het experiment.
6. Het adviessysteem was betrouwbaar.
7. Aan het einde had ik moeite me te blijven concentreren.
8. Na de training was ik voldoende in staat de taak goed uit te voeren.
9. Naarmate ik meer sets heb uitgevoerd, gebruikte ik het adviessysteem steeds minder.
10. De prestatie van het systeem was gelijk gedurende het experiment.

	← Oneens	Neutraal	Eens →
3.	○	○	○
4.	○	○	○
5.	○	○	○
6.	○	○	○
7.	○	○	○
8.	○	○	○
9.	○	○	○
10.	○	○	○

Proefpersoonnr. _____

De extra argumentatie was handig / onhandig omdat:

De censuur was handig / onhandig omdat:

Ik zou het liefst de argumentatie / censuur / niet adaptieve conditie gebruiken, omdat:

Heeft u nog opmerkingen op de taak of in het algemeen op het experiment?

B Task Description (Dutch)

The task description is not only a description of the task itself, but also of all different support types. The support types were given right before the condition was to take place.

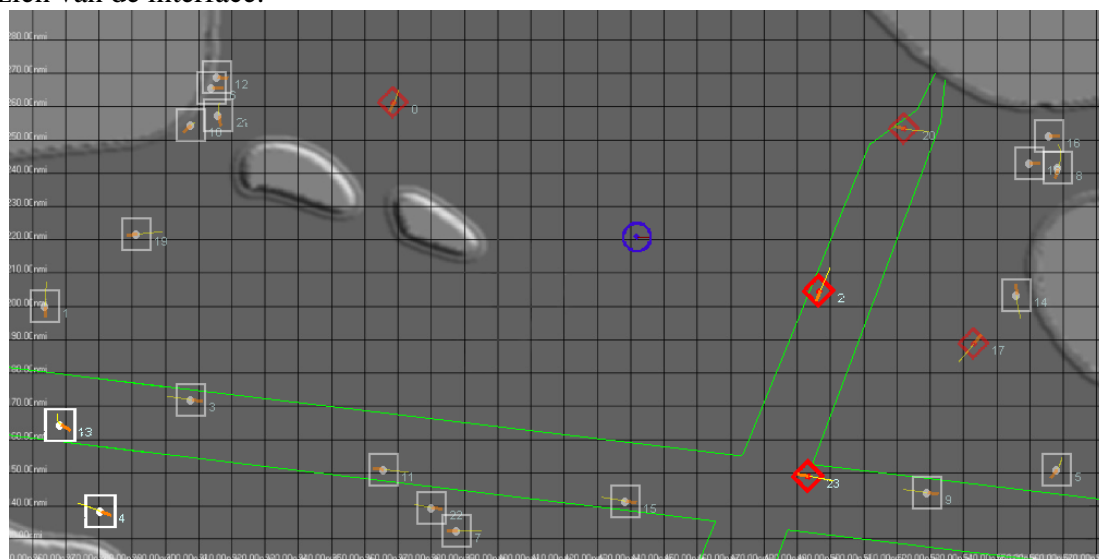
Taakomschrijving

Tactical Picture Compilation Task

Een belangrijke taak die bij de marine wordt uitgevoerd is het opbouwen van een situatieschets van de nabije omgeving. Wat gebeurt er allemaal om je heen, en wat ziet er eventueel uit als een bedreiging? Uw taak is om een situatieschets op te bouwen, door beelden te analyseren van een radar scherm. Op een radarscherm zijn contacten te zien die zich verdacht kunnen gedragen. Deze contacten willen we graag in kaart brengen om hier eventueel op te kunnen reageren. Bij de marine is dit een belangrijke taak die uitgevoerd wordt door een assistent CCO (commando centrale officier), waarna er bij detectie van mogelijke dreiging besloten kan worden om actie te ondernemen.

Het experiment bevat een radar scherm met contacten die rond uw eigen schip bewegen. Uw taak is om van alle contacten de **vijf meest bedreigende contacten** bij te houden. Hoe bedreigend een contact is, kan berekend worden op basis van enkele regels. U wordt tijdens het uitvoeren van het experiment ondersteund door verschillende computer systemen. Hieronder wordt eerst de interface uitgelegd, waarna de regels gegeven worden die een contact bedreigend of niet bedreigend maken.

Het experiment zal zich afspelen op de computer, waar u hieronder een voorbeeld kan zien van de interface:



Op het scherm ziet u in het midden het schip waarop u zich bevindt (blauw icoon), uw schip zal altijd op dezelfde plek blijven tijdens het experiment. Verder zijn er andere contacten te zien op de radar, die weergegeven kunnen worden op twee manieren. Allereerst zijn er neutrale (niet bedreigende) contacten, deze zijn de witte vierkanten op het scherm. In het begin zijn alle contacten op deze manier weergegeven, wat betekent dat u nog geen dreigingen hebt gemarkeerd. Als u een contact wilt markeren als gevaarlijk, klikt u op het contact (linker muisknop), waarop het contact in een rood ruitvormig element verandert. Als u het gemarkeerde element niet meer als bedreigend

ziet, kunt u het nog een keer aanklikken en verandert de status weer in neutraal of niet bedreigend. Binnen de blauwe lijnen is een vaarroute weergegeven, schepen blijven normaal gesproken binnen deze lijnen, maar kunnen hier ook van afwijken.

Het is uw taak om gedurende het experiment de vijf meest bedreigende elementen te blijven markeren. De situatie zal echter gedurende het experiment blijven veranderen, waardoor u continu zal moeten controleren of uw antwoorden nog kloppen en u uw antwoord moet aanpassen.

Nu u weet hoe contacten als bedreigend of neutraal kunnen worden gemarkeerd, is het nodig om te weten wat een contact bedreigend maakt. Voor dit experiment zijn versimpelde regels gebruikt om dit te kunnen doen. Het gebeurt op basis van zogenaamde identificatie criteria (idcrits). Hieronder zijn de vier idcrits gegeven waar bedreiging op gebaseerd wordt:

- **Snelheid**, als een contact sneller beweegt, kan het ook eerder in de buurt van uw schip komen. Een snel bewegend contact is dus bedreigender. De gele streep achter het contact staat voor snelheid
- **Richting**, Een contact dat zich recht naar uw schip toe beweegt is bedreigender dan een contact dat de andere kant op beweegt. De oranje balk aan de voorkant van het contact staat voor richting
- **Afstand**, een contact dat dichterbij uw eigen schip komt is bedreigender dan een contact dat verder weg is. Afstand is de afstand tussen een contact en het blauwe icoon (uw eigen schip)
- **Vaarroute**, zoals al eerder is aangegeven, varen schepen normaal gesproken in een vaarroute (het gebied binnen de blauwe lijnen). Een contact dat binnen deze vaarroute blijft is dus minder bedreigend dan een contact dat buiten de vaarroute is.

Met behulp van deze criteria is het de bedoeling dat u continu de vijf meest bedreigende contacten aangeeft. Belangrijk is dat deze criteria alle vier even belangrijk zijn, en dus even zwaar tellen in uw afweging. Let op dat zowel meer of minder dan vijf geselecteerde contacten leiden tot een negatieve score. Tijdens het wisselen kan het gebeuren dat er te veel of te weinig contacten gemarkeerd zijn als bedreigend. Houd deze periode dus zo kort mogelijk. De bedoeling is dat u altijd 5 contacten heeft gemarkeerd, ook al vindt u de situatie niet bedreigend is het belangrijk toch vijf meest bedreigende contacten aan te geven.

Tijdens het experiment wordt u ondersteund door verschillende soorten systemen. Per experiment wordt uitgelegd wat deze ondersteuning precies inhoudt.

Tijdens het uitvoeren van de taak, zult u een aantal keren worden gestoord met een vraag: ***“In hoeverre, op een schaal van 0-9 heeft u gebruik gemaakt van het advies van het systeem?”***. Als er een vraagteken rechts bovenin het scherm knippert, is het de bedoeling dat u deze vraag beantwoordt door op het toetsenbord het juiste cijfer in te voeren. Als u het advies systeem veel heeft gebruikt bij het bepalen van een antwoord kunt u het systeem een hoog cijfer geven, als dit niet het geval is een lager cijfer. Wanneer u dit gedaan heeft krijgt u een signaal dat de vraag beantwoord is en zal het vraagteken weer verdwijnen. De taak zal echter gewoon doorgaan, dus probeer de vraag te beantwoorden zonder te veel afgeleid te raken van de taak. Onthoud deze vraag goed!

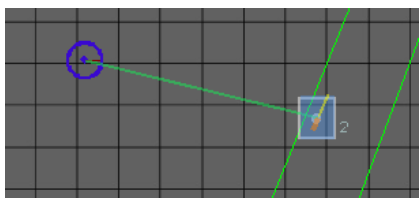
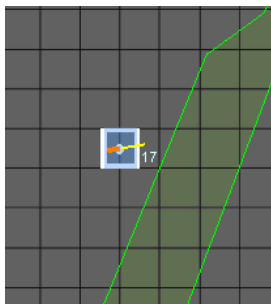

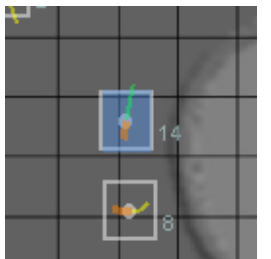


Het experiment zal bestaan uit een schriftelijke test, om te leren werken met de bovenstaande criteria en de verschillende soorten ondersteuning, daarna zal er een training volgen op de computer, om te wennen aan de interface van de taak. Vervolgens zullen er sets komen waar u op verschillende manieren wordt geholpen door een ondersteuningssysteem. Een set zal 10 minuten duren, waarna een vragenlijst ingevuld dient te worden. Tussen de sets is er een korte pauze. Het hele experiment zal uiterlijk 3 uur duren.

Ondersteuning: Argumentatie

Deze manier van ondersteuning gebruikt ook de dikgedrukte contacten om aan te geven waar het systeem markeringen zou veranderen. Alleen wordt er in dit geval extra informatie gegeven. U kunt informatie opvragen over de bedreigings status van een contact, door er met uw muis op te bewegen. De argumenten zijn redenen waarom het systeem vindt dat de markering van het contact moet veranderen of juist niet. Deze informatie wordt op het radar scherm getoond (zie onderstaande figuren). De extra informatie wordt gegeven op basis van de idcrits. Als een idcrit overtuigend is (het schip is bijvoorbeeld erg dichtbij), kan het systeem dit als argument gebruiken om aan te geven dat het contact mede daardoor gevaarlijk is.

Hieronder representaties voor de vier idcrits, een geselecteerd contact wordt aangegeven door een blauwe gloed. Hieronder voor iedere regel (idcrit) de bijbehorende argumenten.

	<p>Afstand In dit geval is het systeem zeker van zijn advies (contact 2 niet bedreigend laten), omdat de afstand tussen het contact en het eigen schip groot is.</p>
	<p>Vaarrichting Het contact beweegt buiten de vaarroute, daarom adviseert het systeem om dit contact als bedreigend te markeren (het contact is dikgedrukt).</p>
	<p>Richting De richting van contact 6 geeft het gegeven advies meer zekerheid. Het advies is om het contact niet bedreigend te laten, omdat het contact in een ongevaarlijke richting vaart.</p>
	<p>Snelheid De snelheid die contact 14 heeft is laag, daarom adviseert het systeem om dit contact niet als dreiging te markeren.</p>

Het systeem kan twee soorten argumenten geven: het kan namelijk aangeven waarom het systeem **zeker** is van zijn advies, wat weergegeven wordt door de argumenten in het **groen** aan te geven (zoals hierboven is uitgelegd). Ook kan het systeem aangeven welke idcrits het advies juist **onzeker** maakt, dit wordt weergegeven door de argumenten in het **rood** aan te geven. Het kan ook voorkomen dat het systeem geen goede argumenten kan geven voor zijn oordeel. Het contact zal dan wel geselecteerd worden, maar er zullen geen argumenten gegeven worden.

De informatie over de argumenten zijn niet alleen zichtbaar op het radarscherm, maar worden tevens weergegeven in een tekstvlak naast het scherm. Hierbij wordt aangegeven wat het advies is op de volgende manier:

GEEN Dreiging / Dreiging
ZEKER / ONZEKER:

- Vaarroute
- Snelheid
- Afstand
- Richting

- wat is het advies?
- is het systeem zeker of onzeker van dit advies?

- de argumenten

Het systeem kan ook fouten kan maken, probeer het systeem dus op de manier te gebruiken die u het beste lijkt.

Ondersteuning: Meer/Minder Advies

Deze manier van ondersteuning gebruikt ook de dikgedrukte contacten om aan te geven waar het systeem markeringen zou veranderen. In sommige gevallen kan het systeem er echter voor kiezen om niet alle adviezen weer te geven. Het systeem geeft dus in principe aan dat contacten van status veranderd kunnen worden, maar in sommige gevallen zal het systeem deze contacten niet dikgedrukt maken. Deze adviezen worden gecensureerd.

De manier van ondersteunen is gemaakt om het voor u makkelijker te maken het advies te gebruiken. Op momenten waar u het advies niet nodig heeft of niet gebruikt wordt het advies weggelaten. Wanneer u het weer nodig hebt zal het volledige advies worden hersteld.

Het bepalen wanneer er gecensureerd wordt is een automatisch proces, waar u verder geen extra informatie over zal krijgen. Let wel dat het systeem ook fouten kan maken, probeer het systeem dus op een juiste manier te gebruiken.

C Written Tests (Dutch)

To get accustomed with the different idcrits, a test was given. The participants were allowed to keep the task description when taking this test. The only purpose of this test was to ensure the experimenter that participant understood the task principles. This test was constructed by Lucassen [8].

Similar tests were given for understanding the argumentation support and the static support (advice in the form of highlights.)

Below the complete tests are presented, exactly how they were given to the participants (in Dutch).

Oefenen met de criteria (idcrits)

U ziet een aantal situaties met verschillende contacten. Probeer de vragen te beantwoorden aan de hand van de vier criteria die hiervoor zijn uitgelegd. Als er iets onduidelijk is kunt u het altijd aan de proefleider vragen.

Aan het einde van deze test zullen de antwoorden met u worden besproken.

Betekenis van de symbolen



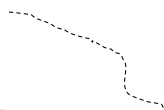
Eigen schip



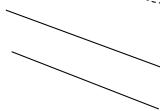
Andere contacten



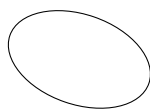
Pijl geeft vaarrichting aan



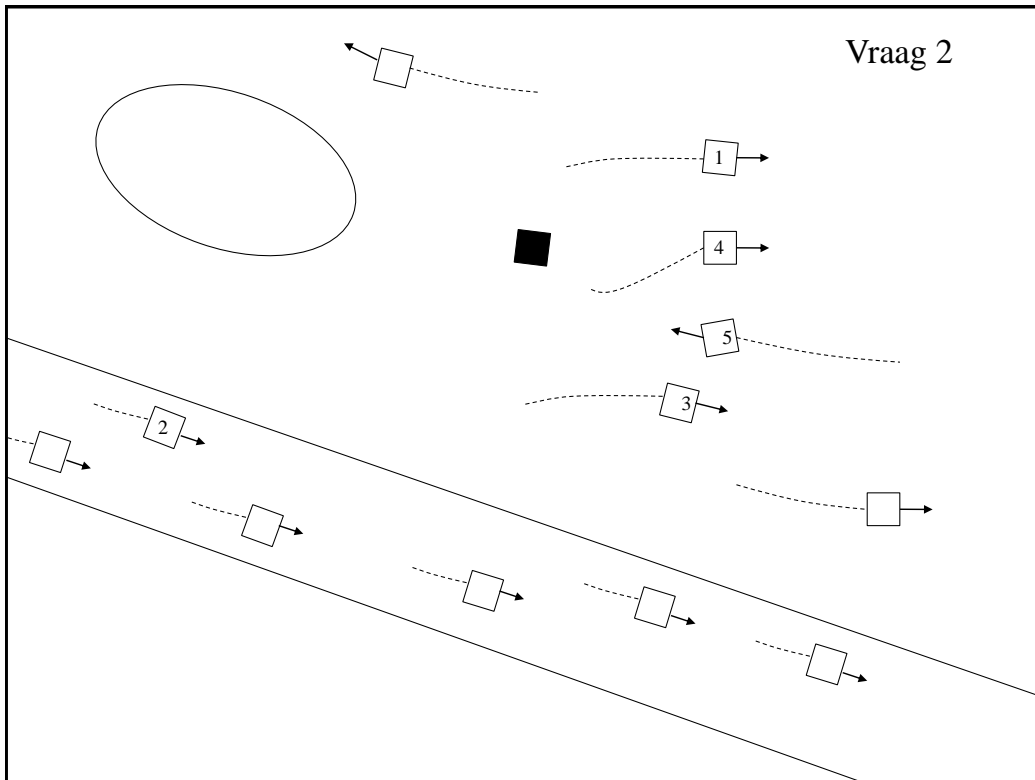
Spoor achter een contact, geeft aan waar schip heeft gevaren. Lang spoor betekent hoge snelheid en kort spoor een lage snelheid



vaarroute



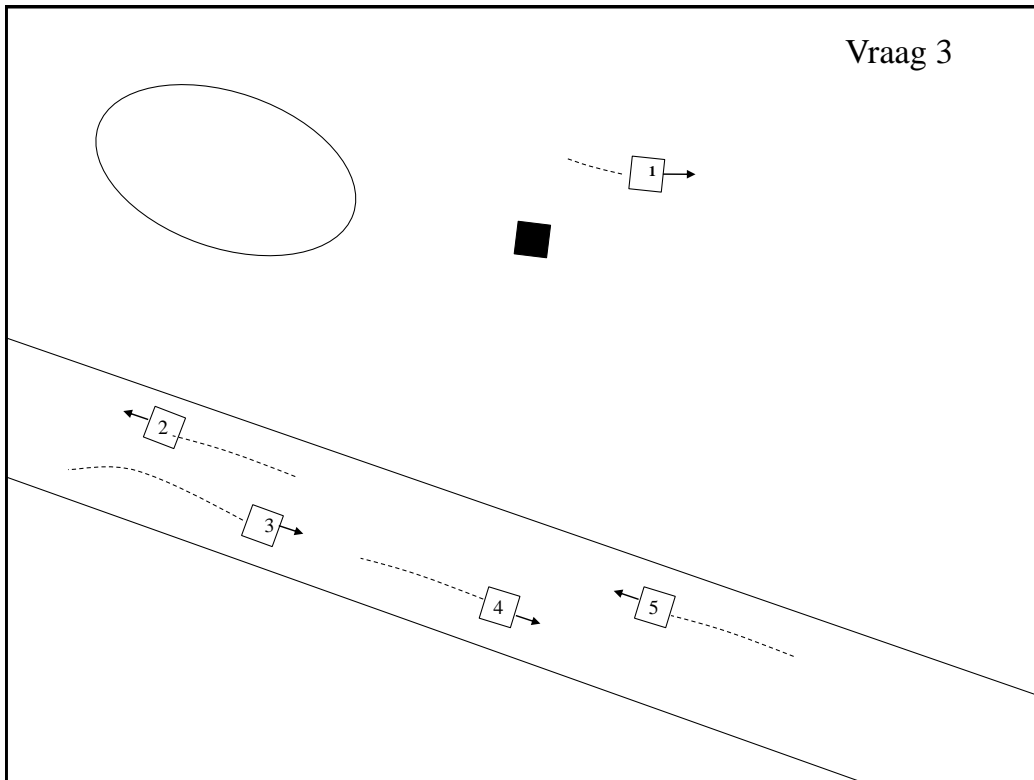
Land



2. Welk contact is het meest bedreigend?

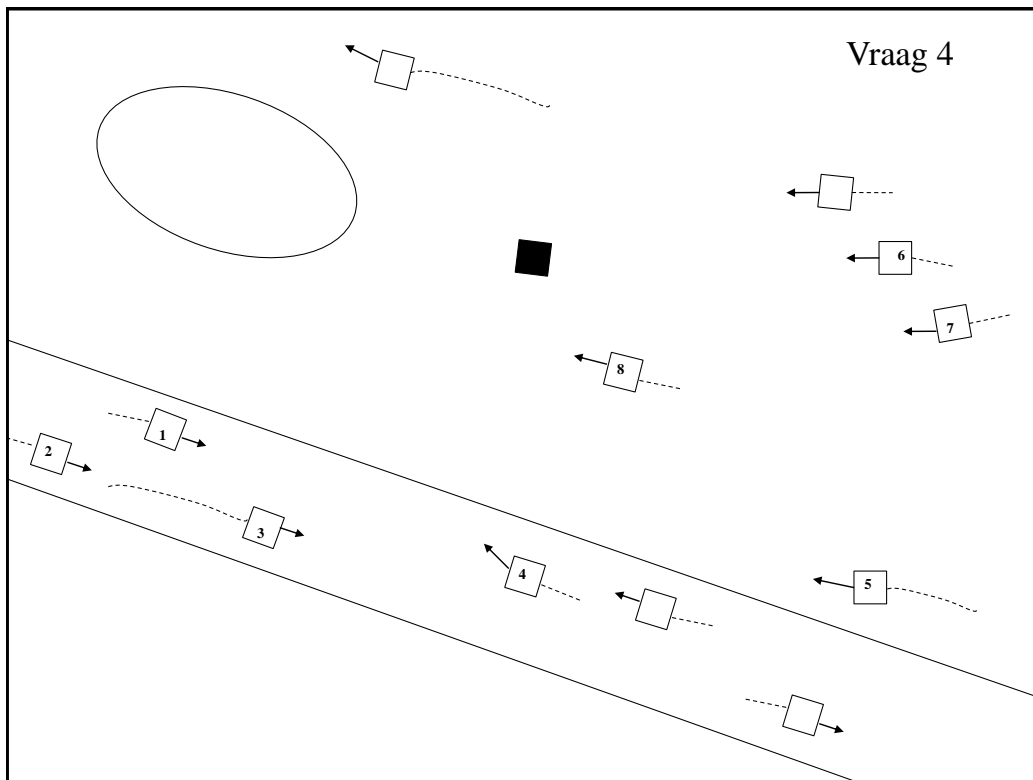
- A. Contact 1
- B. Contact 2
- C. Contact 3
- D. Contact 4
- E. Contact 5

Vraag 3



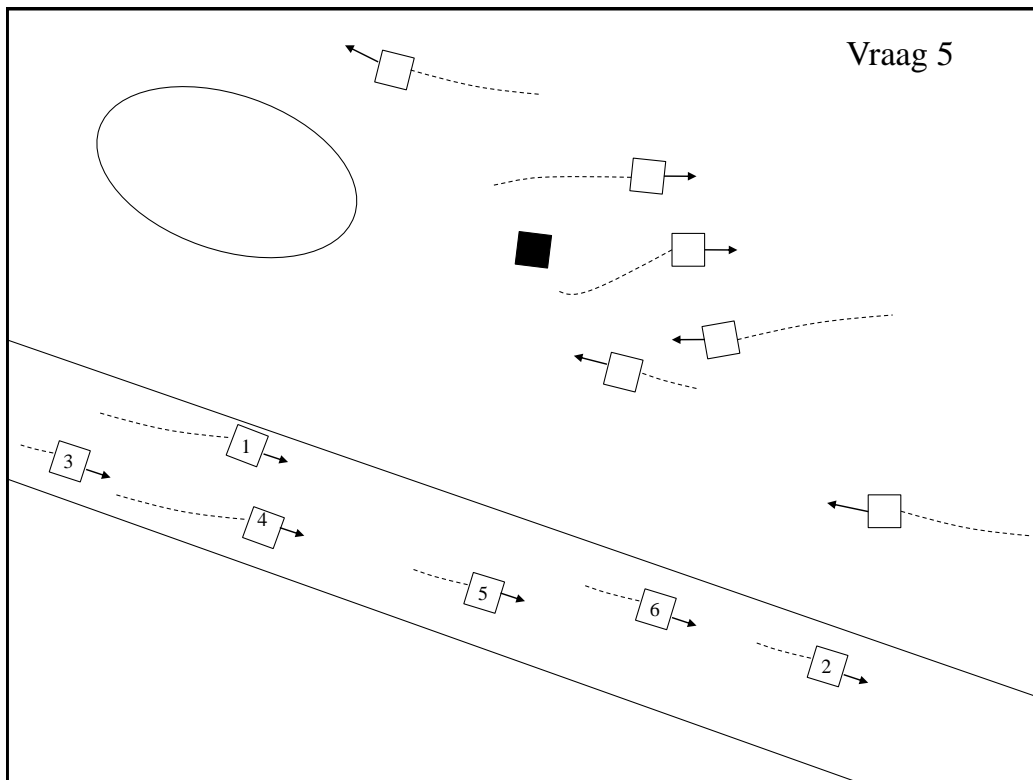
3. Welk contact is het meest bedreigend?

- A. Contact 1
- B. Contact 2
- C. Contact 3
- D. Contact 4
- E. Contact 5



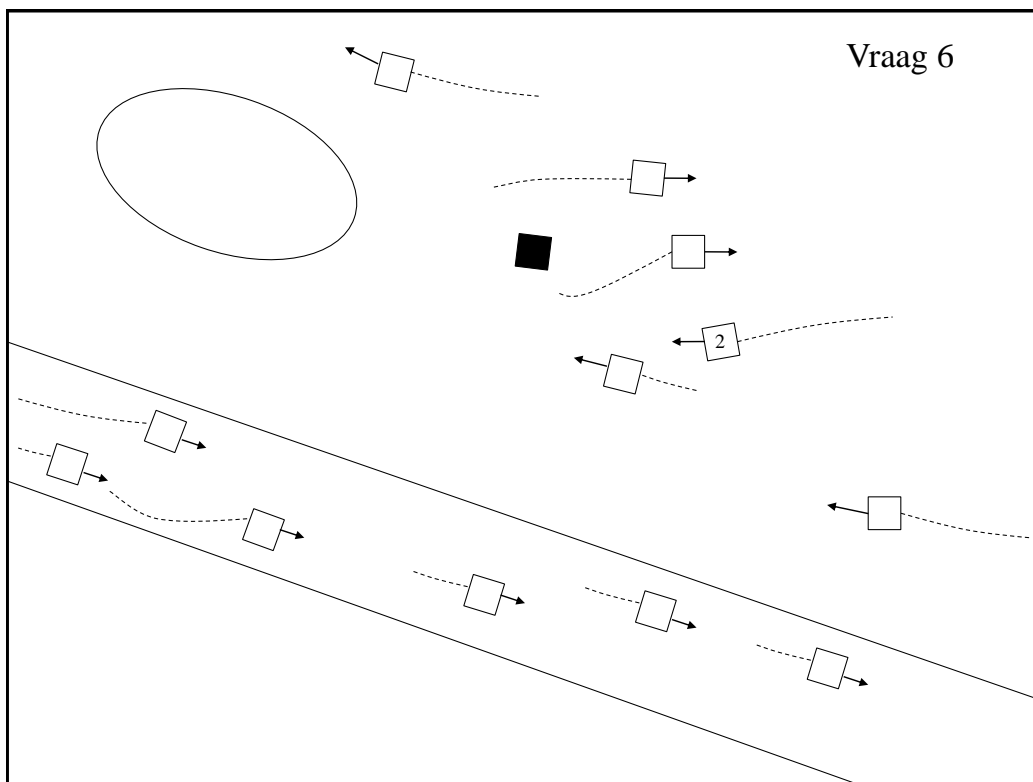
4. Contacten 1 t/m 8 varen langzaam. Welk van deze contacten is het meest bedreigend?

- A. Contact 5
- B. Contact 6
- C. Contact 7
- D. Contact 8
- E. Geen van bovenstaande contacten

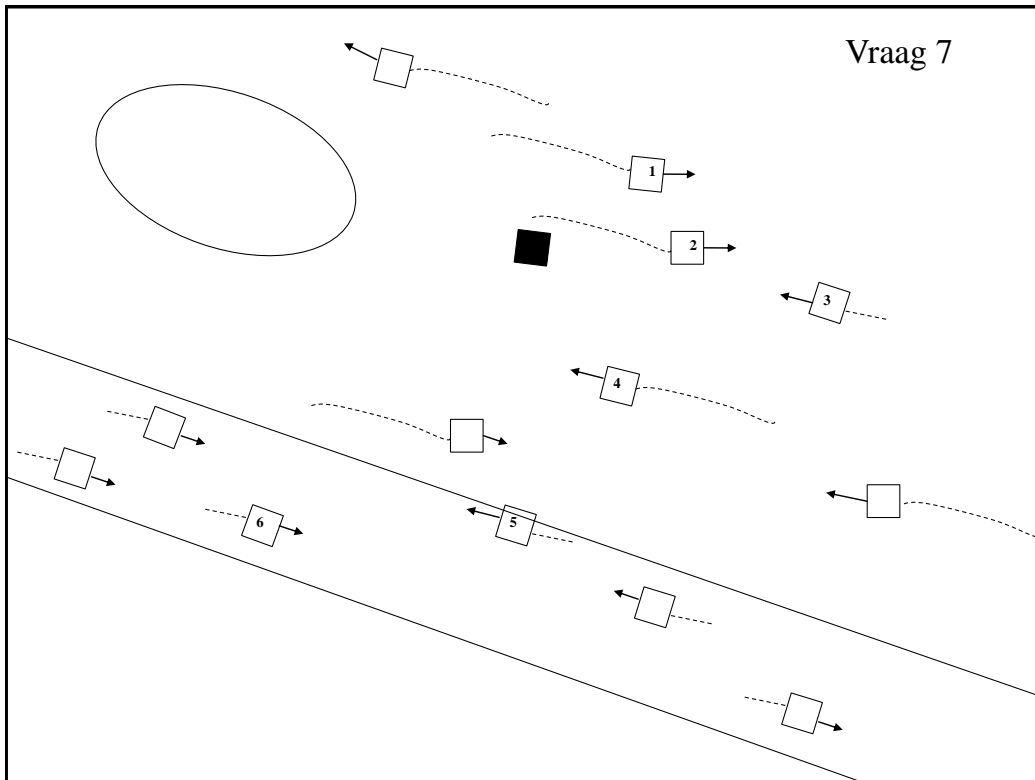


5. Waarom is contact 1 het meest bedreigend van de contacten die binnen de vaarroute varen?

- A. Meest bedreigend door snelheid en koers
- B. Meest bedreigend door afstand tot eigen schip en koers
- C. Meest bedreigend door snelheid, koers en afstand tot eigen schip.
- D. Meest bedreigend door snelheid en afstand tot eigen schip

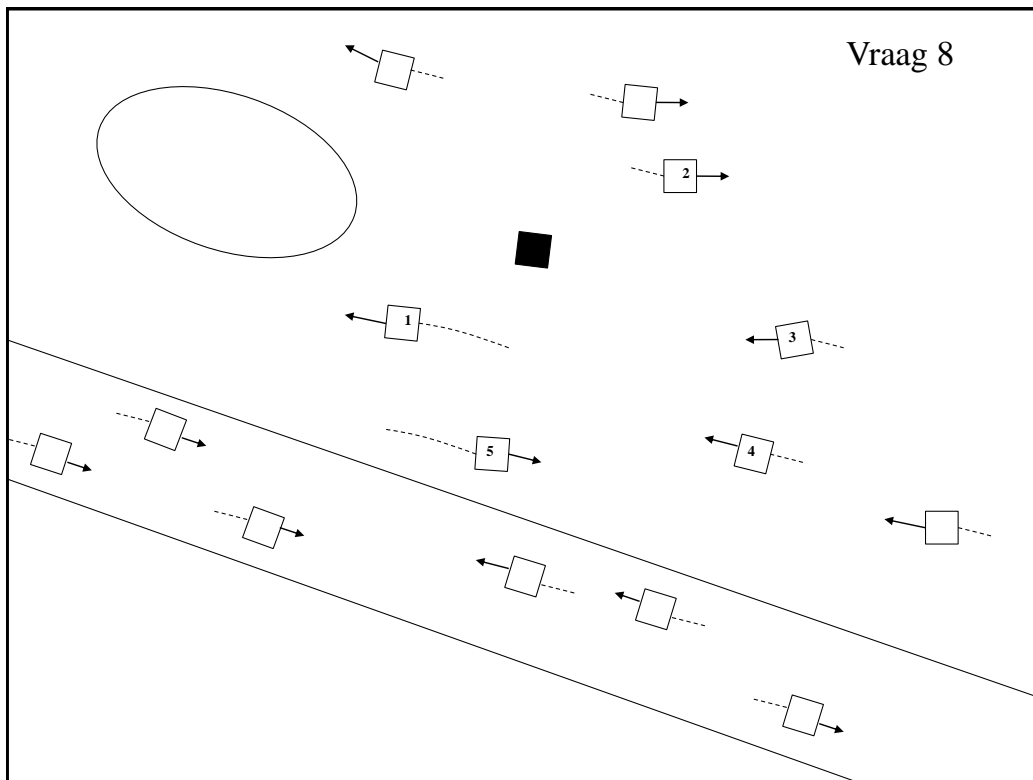


6. Waarom is contact 2 het meest bedreigend ten opzichte van de contacten die *buiten* de vaarroute varen?
- A. Vanwege afstand tot eigen schip en de koers
 - B. Vanwege de hoge snelheid en de afstand tot eigen schip
 - C. Vanwege de koers en de hoge snelheid
 - D. Vaart buiten de vaarroute en met hoge snelheid



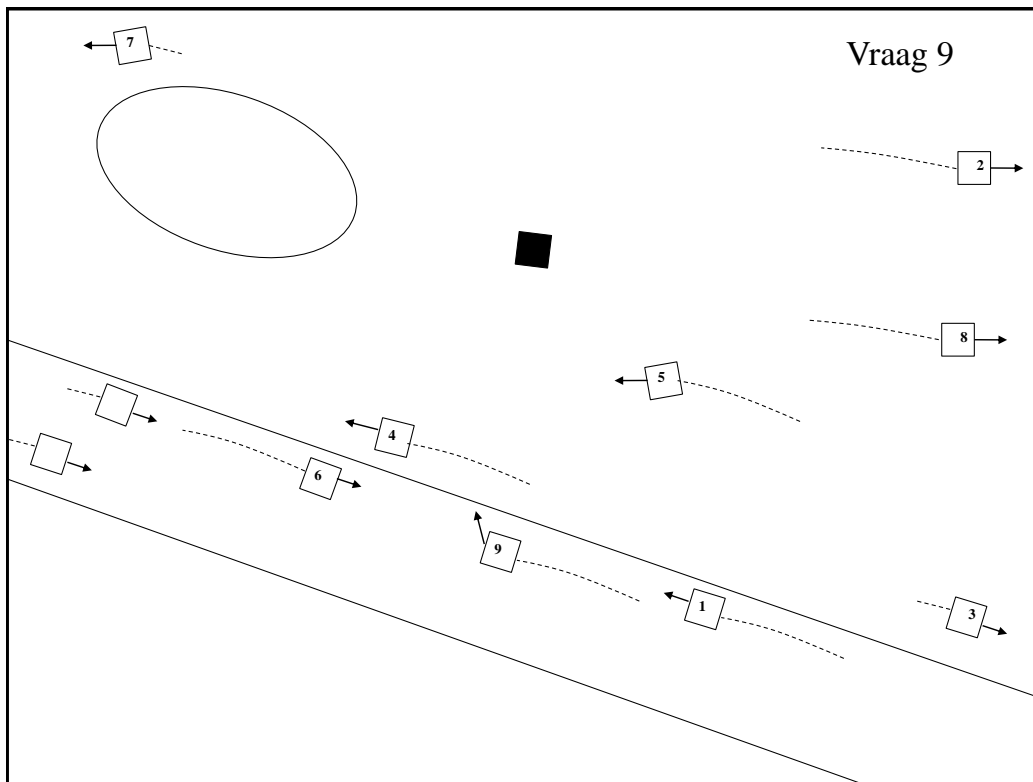
7. Welk contact is bedreigender? 3 of 4

- A. Contact 3
- B. Contact 4
- C. Even bedreigend
- D. Allebei niet bedreigend



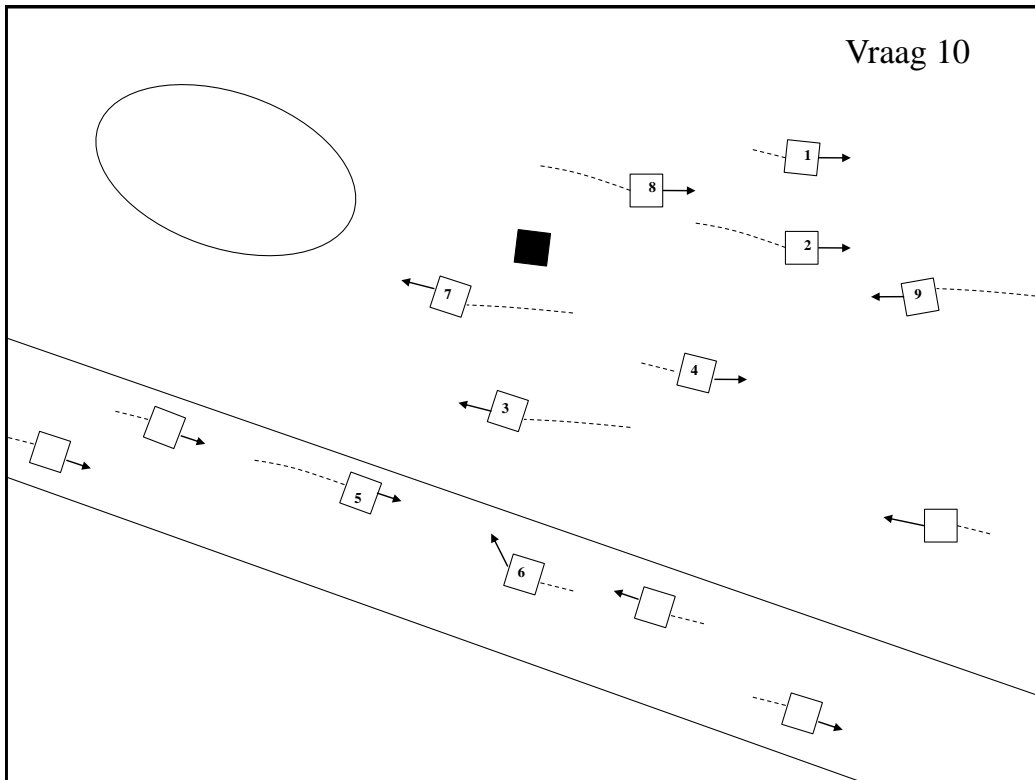
8. Welk contact is het meest bedreigend?

- A. contact 1
- B. contact 2
- C. contact 3
- D. contact 4
- E. contact 5



9. Wat zijn de 5 meest bedreigende contacten?

- A. 2, 4, 5, 7, 8
- B. 1, 4, 5, 6, 9
- C. 1, 4, 5, 8, 9
- D. 2, 4, 5, 8, 9



10. Wat zijn de 5 meest bedreigende contacten?

- A. 3, 4, 7, 8, 9
- B. 2, 3, 7, 8, 9
- C. 2, 3, 6, 7, 8
- D. 2, 4, 6, 7, 8

Oefenen met de ondersteuning

Highlights

U ziet een aantal situaties met verschillende contacten.

Probeer de vragen te beantwoorden over de ondersteuning die gegeven worden, aan de hand van de uitleg. Als er iets onduidelijk is kunt u het altijd aan de proefleider vragen.

Aan het einde van deze test zullen de antwoorden met u worden besproken.

Betekenis van de symbolen



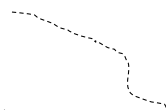
Eigen schip



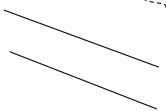
Andere contacten



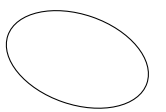
Pijl geeft vaarrichting aan



Spoor achter een contact, geeft aan waar schip heeft gevaren. Lang spoor betekent hoge snelheid en kort spoor een lage snelheid



vaarroute



Land

ondersteuning



Contact door u gemarkeerd als **neutraal** (niet bedreigend)

Niet dikgedrukt (advies is niet veranderen)



Contact door u gemarkeerd als **neutraal** (niet bedreigend)

Dikgedrukt, advies is veranderen (markeren als bedreigend)



Contact door u gemarkeerd als **bedreigend**

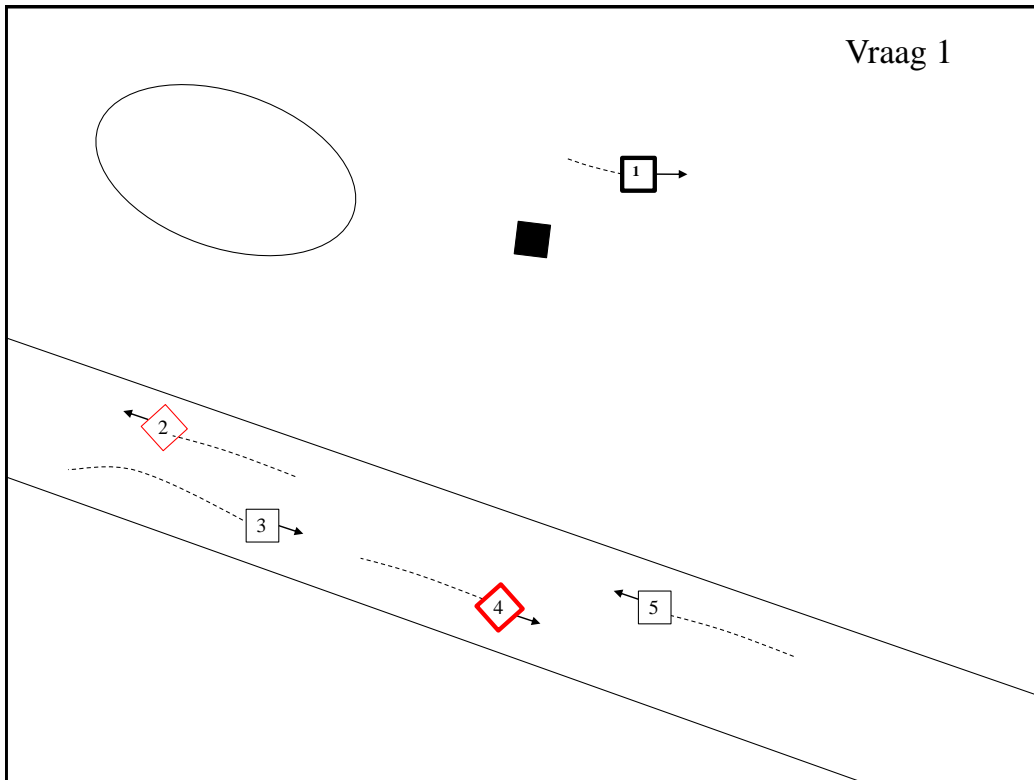
Niet dikgedrukt (advies is niet veranderen)



Contact door u gemarkeerd als **bedreigend**

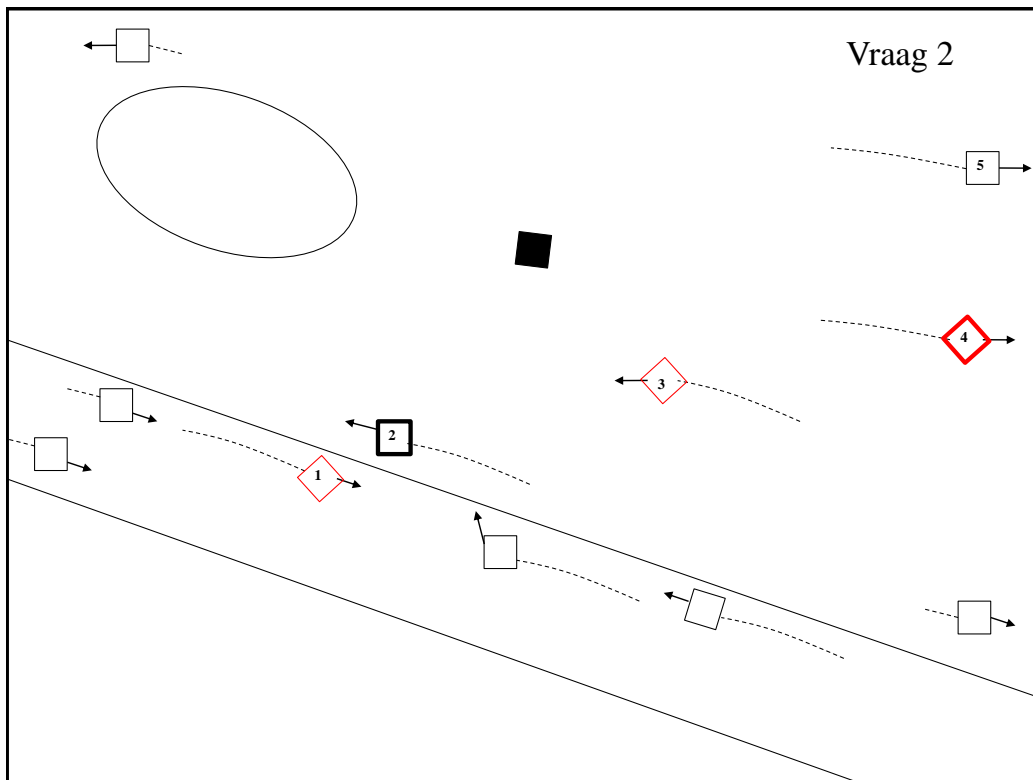
Dikgedrukt, advies is veranderen (markeren als neutraal)

Vraag 1



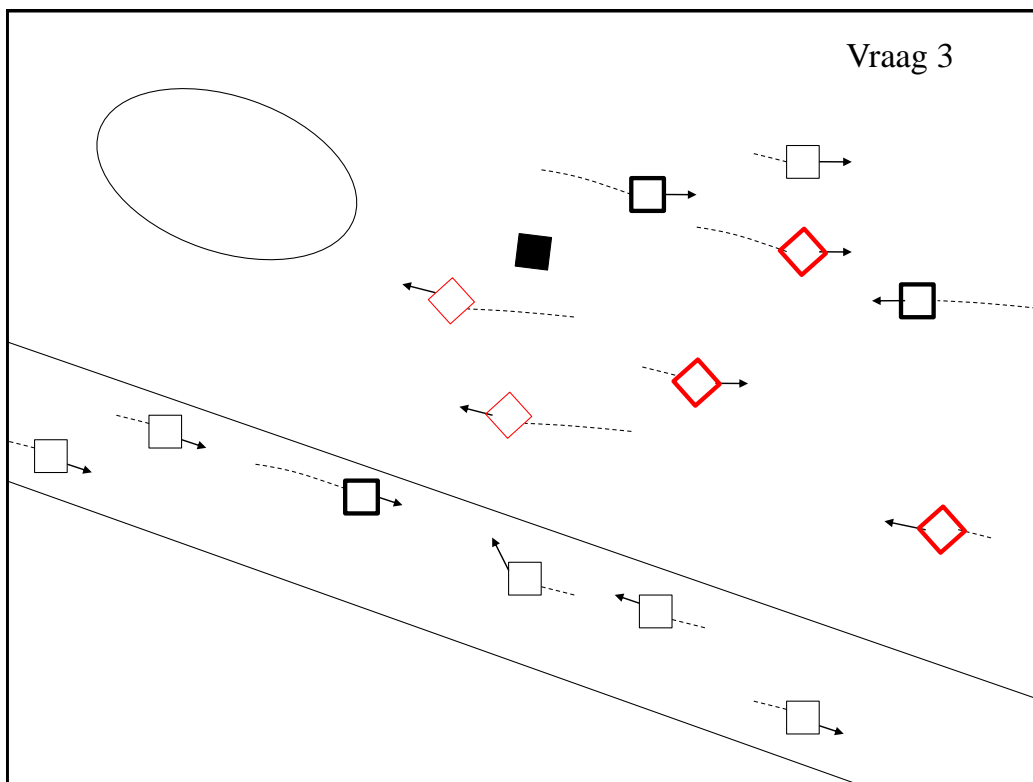
1. Als je het systeem opvolgt, welke contacten moeten er worden aangeklikt (veranderd)

- A. Contacten 2 en 4
- B. Contacten 1 en 4
- C. Contacten 3 en 5
- D. Geen contacten



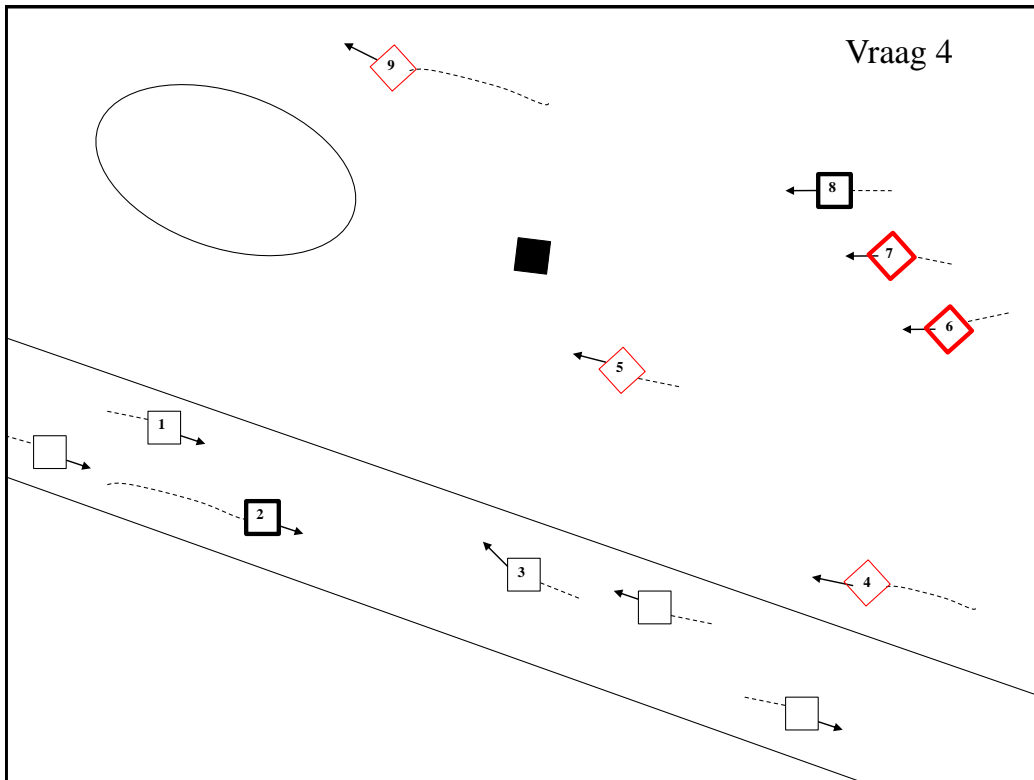
2. Hoe zou het systeem de volgende contacten 1 t/m 5 classificeren?

- A. Dreiging: 1,3,4 Geen dreiging: 2,5
- B. Dreiging: 2,4 Geen dreiging :1,3,5
- C. Dreiging: 1,2,3 Geen dreiging : 4,5
- D. Dreiging: 1,3 Geen dreiging : 2,4,5



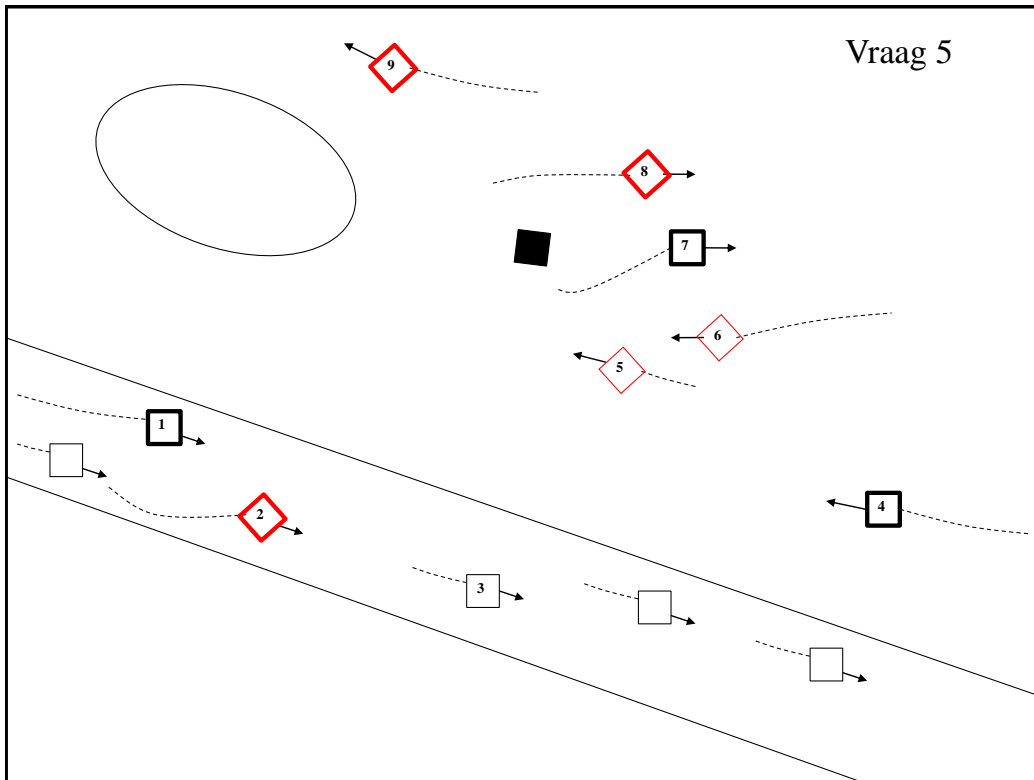
3. Hoeveel contacten zou het systeem veranderen (van bedreigend naar neutraal of andersom) in zijn classificatie?

- A. 3
- B. 5
- C. 6
- D. 8



4. Wat zijn volgens het systeem de 5 meest bedreigende contacten?

- A. Contacten 2,4,5,8,9
- B. Contacten 2,5,6,7,8
- C. Contacten 4,5,6,7,9
- D. Contacten 1,4,5,8,9



5. Wat zijn volgens het systeem de 5 meest bedreigende contacten?

- A. Contacten 1,2,4,7,8
- B. Contacten 2,5,6,8,9
- C. Contacten 4,5,6,7,8
- D. Contacten 1,4,5,6,7

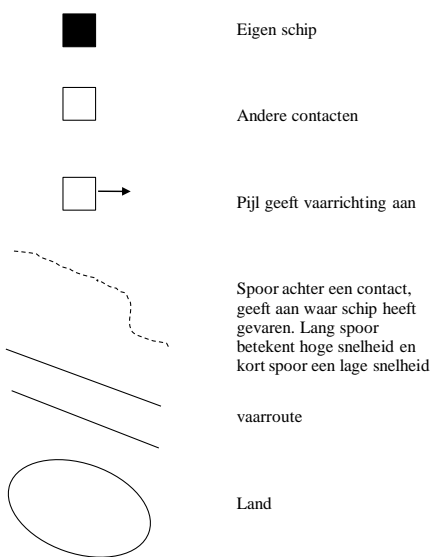
Oefenen met de ondersteuning

Argumentatie

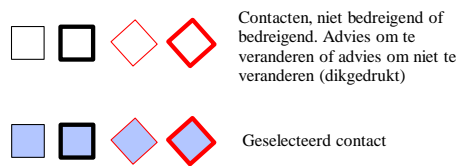
U ziet een aantal situaties met verschillende contacten. Probeer de vragen te beantwoorden over de argumenten die gegeven worden, aan de hand van de uitleg. Als er iets onduidelijk is kunt u het altijd aan de proefleider vragen.

Aan het einde van deze test zullen de antwoorden met u worden besproken.

Betekenis van de symbolen

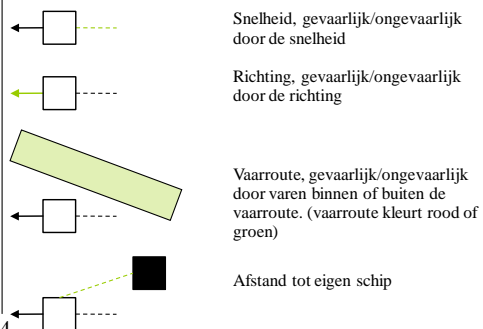


Ondersteuning

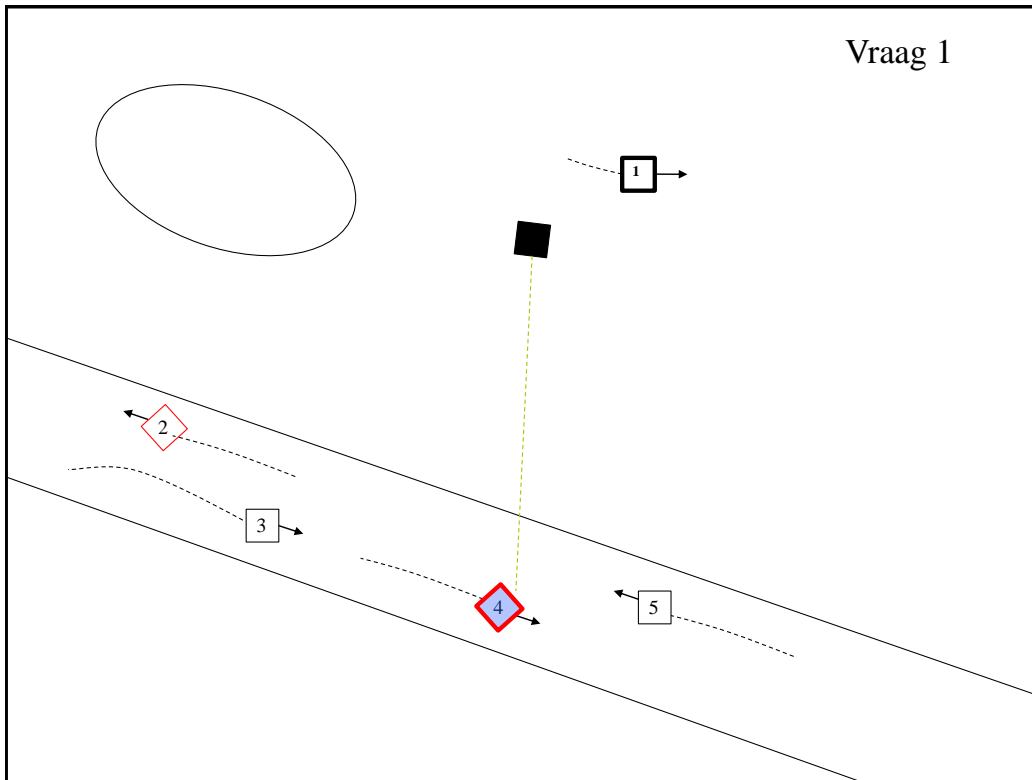


Argumenten

Groen geeft argumenten die het advies zeker maken. Rood geeft aan wat het advies onzeker maakt



Vraag 1



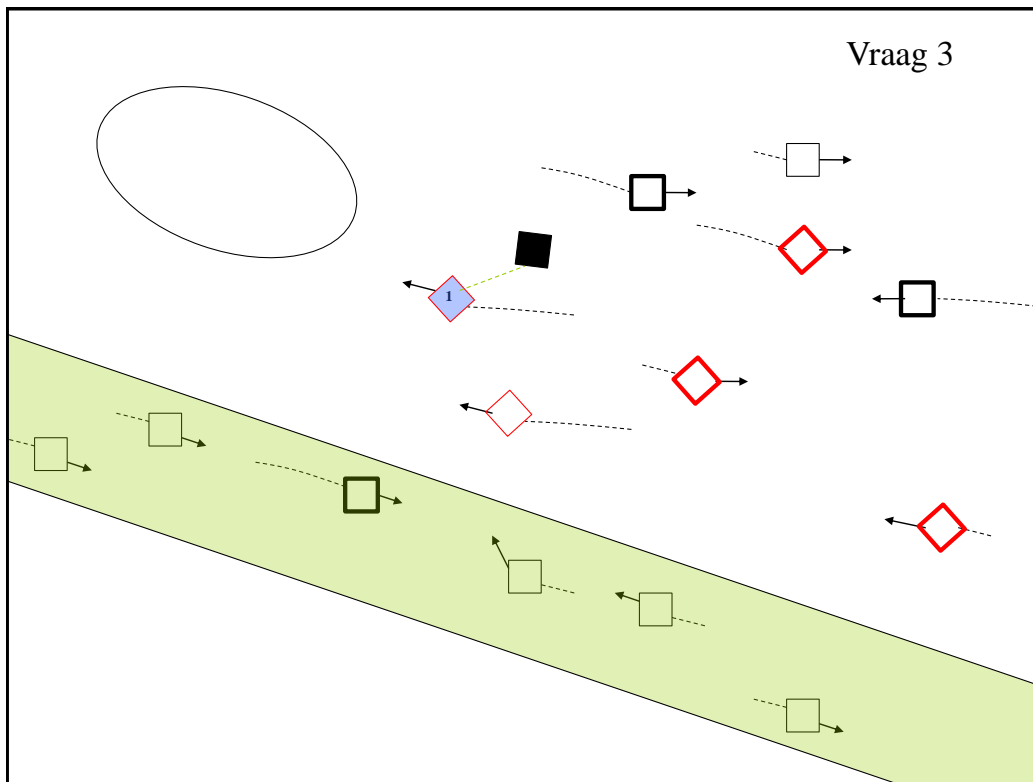
1. Welk argument geeft het systeem hier?

Het systeem is zeker dat contact 4:

- A. gevaarlijk is omdat het dichtbij het eigen schip vaart.
- B. ongevaarlijk is omdat het ver weg is van het eigen schip.

Het systeem is onzeker dat contact 4:

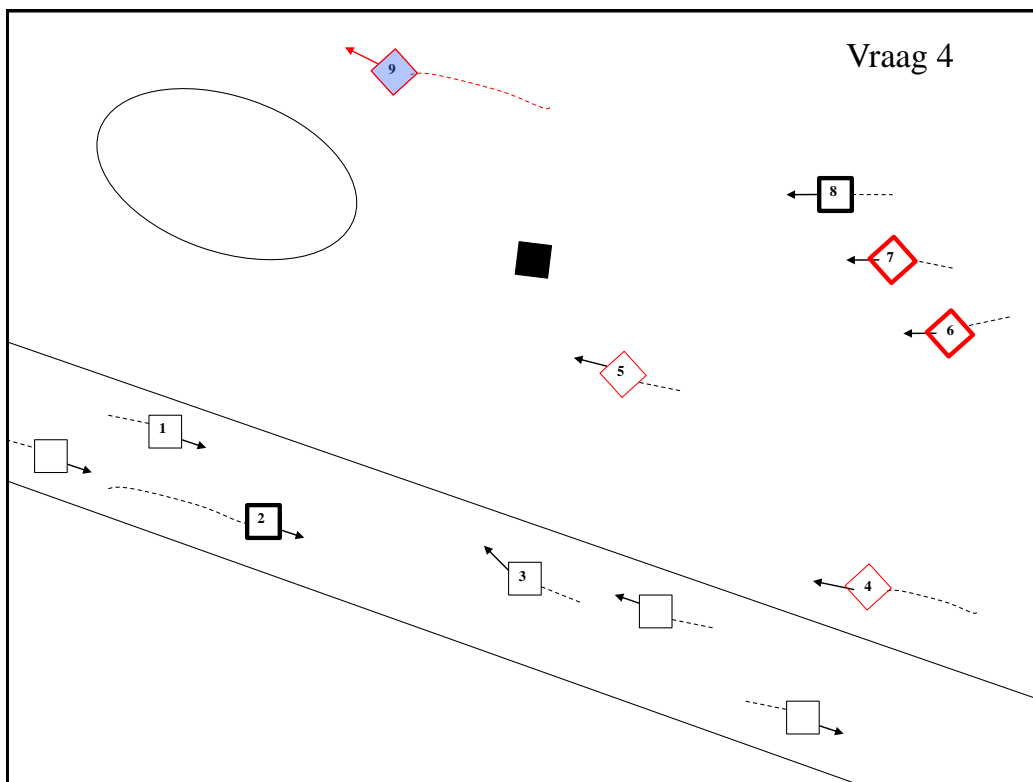
- C. gevaarlijk is omdat het ver weg is van het eigen schip.
- D. ongevaarlijk is omdat het dichtbij het eigen schip vaart.



3. Welk argument geeft het systeem hier?

Contact 1 is:

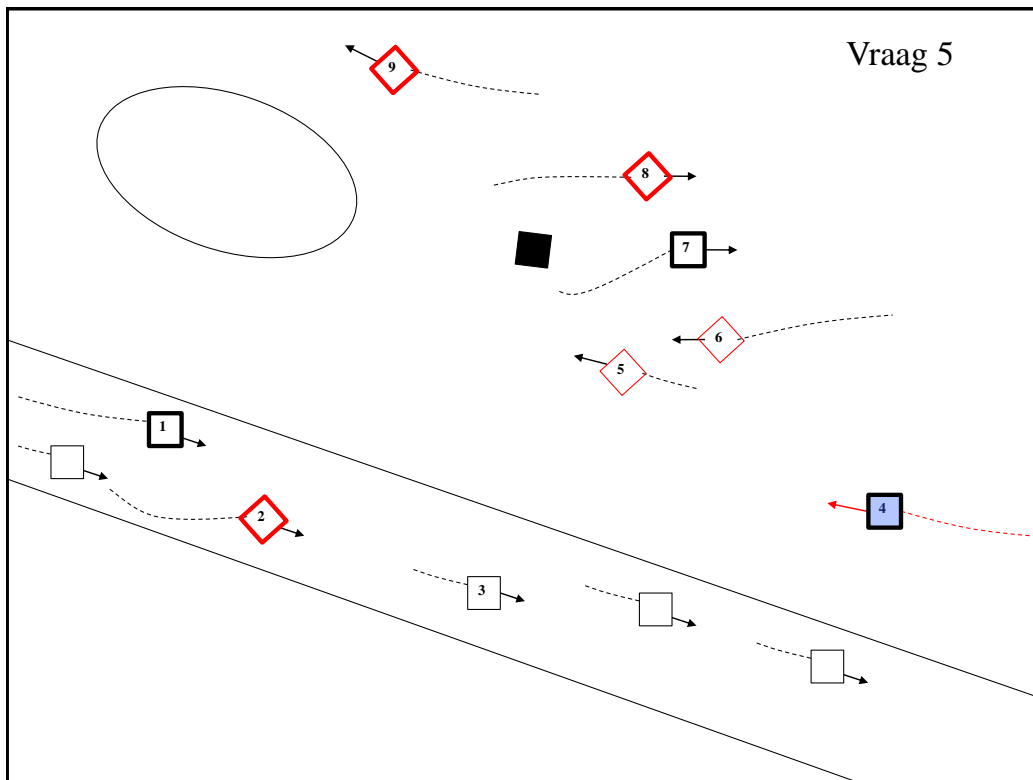
- A. Zeker gevaarlijk omdat het buiten de vaarroute vaart en dichtbij het eigen schip.
- B. bedreigend, maar de afstand en vaarroute maken het advies onzeker.
- C. Zeker niet bedreigend, maar de afstand en vaarroute maken het advies onzeker.



4. Welk argument geeft het systeem hier?

Contact 9 is:

- A. Zeker gevaarlijk vanwege snelheid en richting
- B. Gevaarlijk, maar het advies zou wel eens fout kunnen zijn vanwege de snelheid en richting
- C. Zeker niet gevaarlijk vanwege snelheid en richting.



5. Welk argument geeft het systeem hier?

Advies, contact 4 is bedreigend

- A. Zeker door snelheid en richting.
- B. Onzeker wegens snelheid en richting.

Advies, contact 4 is niet bedreigend

- C. Zeker door snelheid en richting.
- D. Onzeker wegens snelheid en richting.