# Improving human-machine cooperation through eye tracking

## An agent-based approach

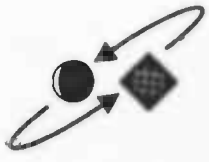
Master thesis
Artificial Intelligence

D.J.H. Everts (s1267507)
January 15, 2007

External supervisor: dr. G.M. te Brake
TNO Defensie en Veiligheid
Kampweg 5
Postbus 23
3769ZG Soesterberg

Internal supervisors: dr. B. Verheij & dr. F. Cnossen
Kunstmatige Intelligentie
Rijksuniversiteit Groningen
Grote Kruisstraat 2/1
9712TS Groningen


TNO Defence, Security and Safety



Artificial Intelligence
Rijks*universiteit* Groningen

## Abstract

In order to improve human-machine interaction, both human and machine need a model which describes (to a certain degree) the state, workings and intentions of the other. Information has to be exchanged in order to keep their models up to date. When for example a computer is busy calculating a request, an hour glass informs the user that the machine is busy and has not crashed. However, humans are less transparent for the machine, when it presents an operator with urgent information it has limited ways of knowing whether the operator is busy thinking or has gone away to fetch coffee. In this research an eye tracker is used to widen the information channel from human to machine. Through the eye tracker the machine knows what the operator is looking at and consequently it can estimate what the operator is doing and is interested in. A multi-agent system was built that improves human-machine interaction through the use of eye tracking. Agent technologies were used, since the computation and data sources are distributed and because flexibility was needed both in the development and deployment phase.

# Acknowledgements

I would like to thank some of the people that have helped me in completing this thesis. First of all my supervisors Bart Verheij and Fokie Cnossen at the Rijks*universiteit* Groningen for their advice and suggestions. Furthermore I would like to thank Guido te Brake at TNO 'Defensie en Veiligheid' for his help, advice and giving me the opportunity to do this research in the inspiring environment of TNO. Last but certainly not least, I would like to thank the following people at TNO, Tjerk de Greef for his support, Kees Houttuin for his knowledge of the ICE and Bert Bierman for his enthusiasm and his help when troubleshooting.

# Table of Contents

## List of tables



## List of figures



## List of abbreviations

ACL = Agent Communication Language
AI = Artificial Intelligence
C2 = Command and Control
ECG = Electro CardioGram
EEG = Electro EncefaloGram
HMC = Human Machine Cooperation
ICE = Integrated Command Environment
JADE = Java Agent DEvelopment Framework
LOA = Level Of Automation
MAS = MultiAgent System
OOP = Object-oriented Programming
SA = Situational Awareness

# 1. Introduction

In this research, techniques were investigated that can improve human-machine cooperation. A multi-agent system (MAS) was built which supports more intelligent ways of giving advice and automating tasks through the use of an eye tracker. A camera based eye tracking system offers an unobtrusive way to deduce what the human operator is doing and what information he is gathering; this knowledge is essential if the machine is to cooperate with the human operator. This research was part of a project conducted at TNO Defence, Security and Safety aimed at developing a better cooperation between human operators onboard a naval ship and the systems they are working with. This thesis reports on the theory and techniques that are used in the MAS, the problems that were encountered and suggests where future research is needed. Our research question can be stated as follows, ***what are the properties of an architecture capable of human-machine cooperation?***

In this thesis we will answer this question by first examining what is necessary to enable human-machine cooperation (HMC). Then we discuss the properties we think a system should posses if it is to support HMC. In this research a system was build which incorporated these properties, in order to test their validity. However a thorough empirical study could not be done within this research due to time constraints, we therefore leave this as future research. However in the overview that is given we do explain how we think the techniques should be used, and give an account our experience with the MAS that used the techniques in this way. This research was exploratory in nature, which was to be expected, to quote Hoc "the necessary development of research on this question of human-machine cooperation can only produce exploratory solutions vis-à-vis the difficult questions of their implementation in real settings." (Hoc, 2001).

## Why is research needed on human-machine cooperation?

Nowadays humans work together with a diversity of machines in their daily lives, for example many routine chores have been automated by machines (e.g. dish washer, bread machine, printer etc.). At present automation can be found almost everywhere, assisting or completely taking over tasks from humans. However there is no cooperation between the two, in the sense that they anticipate on each others needs and help each other when necessary, instead they work side by side as separate entities. Part of this phenomenon stems from the fact that until late in the 1980's, automation was primarily technology driven, if a new technology could automate a task and it

was economically interesting to do so, that task or parts of it were automated. How this affected the operator was only of second interest when designing systems; the focus was on the technology that facilitated the automation. The general mindset was the more automation the better, since machines could perform many tasks much faster, cheaper and more precise.

However there were, and still are numerous tasks that cannot be completely automated, because they require a human level of intelligence which is not yet rivaled by machines. Furthermore it is very difficult, and even impossible for complex domains, to completely specify all the states that system and environment can be in. This means that the system can find itself in situations for which it was not programmed, and thus has no prescribed way of how to function. Humans are much more capable of dealing with abnormal situations, and are therefore needed to monitor automation for abnormalities and take appropriate actions when these occur. In most cases, when possible, subtasks of a complete task were automated, since it was thought to reduce the workload of the operator and increase the safety of the overall system. It seemed logical that by eliminating the human operator as much as possible, we could eliminate human errors as much as possible. But it became clear that this was not the case, in some situations automation even increased the workload of the operator and, even worse, automation related accidents were reported. We think that a significant portion of this problem lies in a poor interaction between man and machine. Instead of working together as a team, man and machine often work together as two completely separate entities. If the machines were replaced by humans this would be explained as a need for cooperation (Hoc, 2001).

## How to enable human-machine cooperation?

In order to let man and machine cooperate with each other, they have to have an understanding of the intentions, state and workings of the other. There is a need for humans and machines to comprehend each others reasoning and behavior (Hollnagel & Woods, 1983). While nowadays human factors are considered when a system is engineered and programmed, these considerations are made beforehand and are static. Only few systems reason dynamically about what the current state and intentions of the human user are. However, in dynamic environments, the state and intentions (of both human and machine) change over time, therefore communication between human and machine has to take place. In well designed systems the programmers have made sure that the system reflects its state and intentions to the operator. The weakest link is however the communication from human to the system. The information the system gets through the normal interaction of the operator with the systems buttons and pointing devices is limited. In most cases an explicit question has to be posed to the operator in order to deduce what he[1] is doing and what his intentions are. This is too obtrusive; it is good engineering practice to develop your system in

[1] Use of the masculine gender in reference to humans has no sexual connotations, but is used only for convenience.

such a way that few physical interactions (button clicks etc.) are needed in order to work with it, and thus most systems are actually designed to receive limited feedback. A camera based eye tracking system offers an unobtrusive way to deduce what the operator is doing and what information he is gathering. In this research an eye tracker is used to widen the information channel between human and machine. A user model is built based upon eye related measures as well as information about tasks the operator has to perform. When the system has constructed a user model it can adapt its performance to the needs of the operator. It can adapt its interface to better suit the needs of the operator, for example it could show only the information that is relevant for the task at hand. Or it can bring information into attention by pointing it out, in cases when the operator fails to notice information the system thinks is important.

Another concept of adapting to the needs of the operator is so called *adaptive* automation, here tasks are dynamically allocated between human and machine in order to optimize overall system performance. The operator takes control over the system in situations were he has enough resources available to do this. This will prevent him from getting bored and it will train his skills in manipulating the system manually. In more demanding situations the system takes over more work from the operator, to prevent him from getting overloaded. The goal is to keep the operator's workload in an ideal range, not too low to prevent boredom and no too high to prevent stress.

## Relation to Artificial Intelligence

Artificial intelligence (AI) is the science that studies intelligent behavior, learning and adaptation in general and tries to incorporate this knowledge into machines. Many of the issues that have to be solved in order to facilitate human-machine cooperation are AI issues. For example, (AI-related subjects are in italics) cooperation between two entities can only exist if they have *knowledge* of the task and task domain at hand, as well as having a good idea of the state, workings and intentions of the other. Besides this knowledge, *intelligent reasoning* is required in order to come to conclusion on what to do with this information. Furthermore cooperation is enhanced when the two entities work together over a period of time and *learn* the preferences and intentions of the other. Furthermore with respect to human-machine cooperation, AI is used to in assist in decision-making and to foresee possible conflicts in the decision-making process of the human and machine. All these abilities a machine should posses in order to cooperate with a human can not be established without the use of AI techniques and concepts.

## Relation to other research

Extensive research has been done on the correct use of automation over the past decades (Bainbridge, 1983; Billings, 1997; Parasuraman & Mouloua, 1996; Parasuraman & Riley, 1997 Rasmussen, 1986; Wickens & Holland, 2000). The concept of adaptive automation was already mentioned by Rouse in 1976. One step beyond adaptive automation is what Hoc calls human-machine cooperation (e.g. Hoc 1995; Hoc 2001). However the technologies for its practical implementation have been studied only recently. A fundamental issue concerns the means by which adaptive automation is invoked (Sheridan & Parasuraman, 2006). In 1992 Parasuraman et al. identified five techniques that can be used as criteria for determining when to adapt automation: critical events, operator performance, physiological measures, modeling and hybrid methods. Scerbo, Freeman et al. give an overview of candidate psychophysiological measures which give information about the workload of the operator (Scerbo, Freeman et al., 2001). However none of these measures can be said to accurately correlate with operator workload for different tasks, settings or even operators.

We propose that besides the five criteria of Parasuraman et al., two other criteria can be helpful, task requirements and deficits in information uptake. When an operator has been assigned multiple demanding tasks, and thus has high task requirements, this can be a cue for automation to take over work or reallocate it to another team member. When we know that the operator has to perform a lot of work, the probability that he requires support is higher. Deficits in information uptake can also be a reason to support the operator. With the aid of an eye tracker we can determine what information the operator is gathering. While it is not per se true that you have processed and consciously know about information that you have seen, the reverse is certainly true; if you have not seen a particular piece of visual information you can not know about it. We can therefore deduce what information the operator is lacking in order to perform the task with success, and let automation take over if this deficiency of information is too high.

Eye movements have been thoroughly studied, however there has been only some research on the on-line assessment of eye movements and how they can be used to change the interaction with the user. Anderson and Gluck used eye gaze as an extra information source to determine students' problem-solving strategies in a tutoring system for algebra (2001). Eye data was used to asses reading performance in a system for automatic reading remediation in Sibert, Gokturk & Lavine (2000). When the user seems to have problems recognising a particular word, indicated by a prolonged fixation, the system aids by pronouncing the word. Starker and Bolt constructed a virtual storyteller which used eye gaze to determine the interest of the user in particular object presented on screen, and used this information to determine which object to talk about (1990). However there has been no research done on how HMC can be enabled or improved through the use of eye tracking information.

## Contribution to science

While answering our research question: *what are the properties of an architecture capable of human-machine cooperation?* We investigate what the requirements are for a system capable of human-machine cooperation (an agent based approach is necessary, see "why use an agent based approach?"), we explain what eye tracking is and how it can be used to adapt automation in such a way that it can enable human-machine cooperation. However, little research has been done on the online use of eye tracking in human-machine cooperation, our research is therefore exploratory in nature. We have investigated and coupled techniques that seem necessary to enable human-machine cooperation, we tested our system with a limited set of test persons and left a thorough empirical validation of our system as future research. We are therefore not able to *define* features in eye movement behaviour, or other psychophysiological measures indicative of the mental or physical state of the human operator. Nevertheless did our research point out directions that seem particularly fruitful when trying to achieve human-machine cooperation.

## Overview of thesis

This thesis is composed of six chapters, the first is this introduction. The following chapter describes automation, adaptive automation, human-machine cooperation and what is needed to enable HMC. Chapter four and five discuss two requirements for a cooperating system. First, the system needs information about the operator, in chapter three we describe how this can be done and we will focus on eye tracking in particular. Second, the system that is to be build will get complex, chapter four describes agent-based programming which is used to harness complexity furthermore an example is given of how a general cooperating multi-agent system could look like. Chapter five reports how the general model was implemented within a practical project. This thesis concludes with a discussion on the results of this research and points out where further research is needed.

# 2. Automation and human-machine cooperation

*"... automation should either be made less intelligent or more so, but the current level is quite inappropriate."* (Norman, 1990)

This chapter discusses automation, human-machine cooperation (HMC) and discusses what is needed to enable HMC. Advances in technology provide an ever increasing application of automation in the everyday life of humans. Machines have taken over an enormous amount of work because they are more accurate, less expensive and always work at their maximum performance. Besides, they never complain, even if they have to perform dirty, monotonous or hazardous work. Furthermore machines can perform tasks that are beyond human limits, such as lifting very heavy loads or calculating complex mathematical problems within a second. In 1951 Fitts published a list of items in which machines were better and things in which humans were better (see table 1).

| **Attribute** | **Machine** | **Human** |
|---|---|---|
| Speed | Much superior | Comparatively slow, measured in sec. |
| Power output | Much superior in level and consistency | Comparatively weak, less than 1500W max, less than 150W during a workday |
| Consistency | Ideal for consistent repetitive action | Unreliable, subject to learning and fatigue |
| Information capacity | Multi-channel. Information transmission in megabit/sec. | Better for principles and strategies, access versatile and innovative |
| Memory | Ideal for literal reproduction, access restricted and formal | Better for principles and strategies, access versatile and innovative |
| Reasoning, computation | Deductive, tedious to program. Fast accurate. Poor error correction | Inductive. Easy to program. Slow, inaccurate. Good error correction |
| Sensing | Specialized, narrow range. Good at quantitive assessment. Poor at pattern recognition | Wide energy ranges, some multi-function |
| Perceiving | Copes poorly with variation in written/spoken material. Susceptible to noise | Copes well with variation in written/spoken material. Susceptible to noise. |

Table 1. The "humans are better at, machines are better at" list (Fitts, 1951)

This list seems to favor the capabilities of the machine, as if the human is mostly a source of limitations and errors. It focuses on the strong points of machines, and leaves out the weaknesses of machines or the strong points of humans. When one would only consult this list, one would think that human operator should be replaced as much as possible by the more powerful machine.

However humans are still needed to perform or control most of the work in this world. There are many situations where complex cognitive reasoning is needed, which is hard or impossible (or at least at this moment) to be performed by machines. Examples are data analysis and decision making in situations where there is ambiguous information. Besides that we cannot (yet) program all the intelligence that is needed to perform some of the more complex tasks, there is another obstacle. It is very difficult, and even impossible for more complex domains to completely specify all the states that system and environment can be in. This means that the system needs to be able to deal with situations which were not anticipated when it was designed.

As a reaction to the Fitts list, there is the "un-fitts" lists (see table 2), which does not concentrate on the shortcomings of humans. The views listed in table 2, fit into design approaches where the human operator is at the center of the design approach, and machines are used to support them (a human-centered versus technology centered design approach).

| **Machines** | |
|---|---|
| **Are constrained in that** | **Need people to** |
| Sensitivity to context is low and is ontology limited | Keep them aligned to the context |
| Sensitivity to change is low and recognition of anomaly is ontology limited | Keep them stable given the variability and change inherent in the world |
| Adaptability to change is low and is ontology limited | Repair their ontologies |
| They are not "aware" of the fact that the model of the world is itself in the world | Keep the model aligned with the world |
| **People** | |
| **Are not limited in that** | **Yet they create machines to** |
| Sensitivity to context is high and is knowledge- and attention-driven | Help them stay informed of ongoing events |
| Sensitivity to change is high and is driven by the recognition of anomaly | Help them align and repair their perceptions because they rely on mediated stimuli |
| Adaptability to change is high and is goal-driven | Affect positive change following situation change |
| They are aware of the fact that the model of the world is itself in the world | Computationally instantiate their models of the world |

Table 2. A version of Woods' "Un-Fitts" list (Hoffman et al., 2002)

## Types of behavior required for performing tasks

There are different types of behavior that can be required to perform a task, some types are easy to automate and others are more difficult to automate. Rasmussen defined three types of operator functioning: skill-based, rule-based and knowledge based behavior (1983). Each consecutive type represents an increasing need for information processing. Skill-based behavior is behavior that follows immediately on an event or intention without the need for conscious control. Take for example an operator of a car, he reacts immediately and subconsciously to curves and bumps in the road. One level higher is rule-based behavior, when the operator sees another car driving towards him on a road coming up on his right side he applies the rules of traffic and lets the other driver go first. Knowledge based behavior requires the most processing of information. When for example our driver hears that there is traffic jam on his route, he will consider possible other routes and then decide how to proceed. Normal operator behavior is of course a mix of these three levels, when 'our' driver contemplates on an alternative route he will of course still react to curves in the road and yield the right-of-way to any vehicle approaching from the right.

Automation can most easily and reliably be applied to tasks that require rule-based behavior, since the rules can be easily transferred in IF ... THEN statements. The workings behind skill-based behavior are often difficult or even impossible to elicit directly from the operator. The operator has become, through extensive experience, such an expert on that behavior that he does not longer know *how* he does it, he just does. With the current level of AI we can also not automate all aspects of knowledge based behavior. For example, theoretically we are capable of automating the take off, mid-flight and landing of an aircraft, it would thus seem that the whole flight can be automated. However this is not being done, because although most flights are routine there can arise situations that are very hard (and even impossible) to predict beforehand. Pilots are still needed onboard to deal with these abnormal situations, because (as said before) humans are (still) much better in dealing with ambiguous information, errors and new unforeseen situations. One could say that at present automation can take care of work that is routine but it fails in abnormal situations. Ironically, it is in these abnormal situations where the workload is the highest; since the operator has to identify the abnormality and reason about what action to take. This is referred to as one of the 'ironies' of automation, in situations where the workload is the highest, automation is of least assistance (Bainbridge, 1983).

## The more automation the better?

At this moment machines are not capable of automating all tasks but at least we should automate as much as possible, right? At first one might think that more automation is always better since that means that the operator has less to do. However in situations where automation cannot act completely autonomous, humans need to monitor the system to make sure it behaves the way it

should and interfere when necessary. This implies that the role of the operator changes from an active participant to a more passive monitor. Unfortunately, humans are not well suited for such a role, e.g. even highly motivated operators cannot maintain full focus on a situation where nothing or little happens for a prolonged period of time. Furthermore when the operator has had extended satisfactory experience with the automation in question, he will stop monitoring the automation thoroughly because it always has behaved the way it is supposed to. This complacency becomes a serious problem in cases of abnormalities. Because of the lack of monitoring, the operator has less situational awareness of the evolving state of the automated system. In order to make (good) decisions, and quickly know what to do in case of an abnormality, one would like to be fully aware of the current situation, that is have good situational awareness. SA can be defined as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future" (Endsley, 1988).

Ironically the problem of complacency will only get worse, not less, when automation is made even more reliable and fewer abnormalities occur. The more reliable automation gets the more complacent the operator will be, because the automation seems infallible. Unfortunately making a completely failure free automated system is impossible. Besides, as automation is getting more and more complex, it does not only mean that automation can handle more complex situations, it also means it becomes more difficult to write the code completely bug-free. Complex automation typically implies that the logic behind the automation is more difficult to comprehend, and that more lines of code are needed to code this logic. Studies on the amount of bugs-per-line yield different results, however it is safe to say that roughly, depending on the language used and development process, every thousand lines of programming code contains a 'bug'. Besides, there are also cases where the hardware crashes or the electric supply to power the hardware fails. In those situations the operator has to take over control again, he has to enter the loop. However he does not have complete knowledge of the situation, until on that moment, he has low situational awareness. He needs time to reorient himself and get familiar with the state of the system, but in time-critical situations this time may not be available. This is summed up in one of Bainbridge's ironies of automation; the higher the reliability of automation the poorer the human response to it will be (1983). We should keep in the back of our minds that humans are not failure free either, however, certainly at this moment, humans have much better error recovery.

Another contributing factor to a diminished SA, is so called being 'out-of-the-loop'. Norman describes being out-of-the-loop as follows: in terms of control theory a system has a desired state, means for getting towards its desired state and a feedback loop which continuously compares the actual state with the desired state and if necessary performs an action which brings the system closer to its desired state. The combination of this control and feedback is called the control loop.

When the human operator is handling the system manually, he is an essential element in the control loop. When automation takes over lower level actions, the role of the human operator changes. He becomes a supervisor instead of a controller: he is out of the loop (Norman, 1990). Because of this he has less knowledge of all the elements in the environment and their behaviors, because they are no longer controlled by him. It is however important to always have a good understanding of the situation you are in, as Endsley points out, in the majority of cases, people do not make bad decisions or execute their actions poorly; they misunderstand the situation they are in (Endsley, 2003)

Besides, when the operator never takes manual control, his operating skills will decay over time, he will become a mere 'button pusher'. As said before, automation usually only fails in abnormal situations, however these situations require the most of the operator's skills. It is therefore a good idea not to always automate as much as possible but to keep the operator in the loop. This will keep the operator aware of the situation, prevent him from getting bored, and it will also keep up his skills.

## Adaptable and adaptive automation

In this research we try to enable HMC, one aspect of HMC is that the machine is capable to adapt his support dynamically to the needs of the operator, one of this types of support is the automation of tasks. One can distinguish two forms of *dynamic* automation: adaptable and adaptive automation. We use the term *adaptable* automation, as automation where the human operator, decides whether the machine or he himself will perform the task. He is therefore also the task manager (See Figure 1). This means however that implicitly the operator has to fulfill an extra task, that of monitoring his own workload.

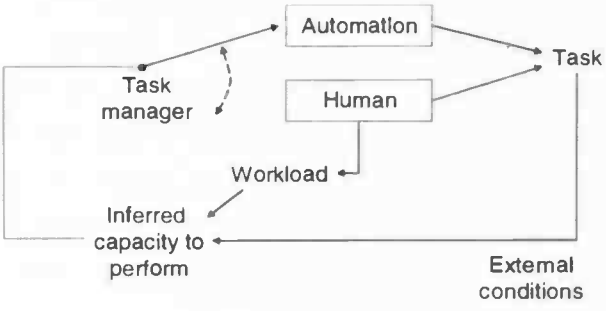



Figure 1. Adaptive automation (Wickens and Holland, 2000)

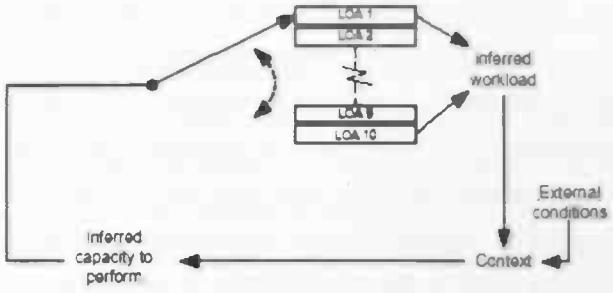



Figure 2. Adaptive automation and LOAs (Kruit, 2003)

However it might well be the case that if the operator had the time to monitor its workload and assess whether it should perform the task, he also would have had the time to perform the complete task himself. In situations were the workload is really high, the operator does not even

have the time to consider which of the tasks have to be automated. Furthermore research has shown that people tend to overestimate their own ability to perform with respect to other humans or automation. The operator might therefore not be the best task manager (Wickens & Holland, 2000). We define *adaptive* automation (AA) as automation where the choice of automation is dependent on the state of the operator *and* system combined. (Now the system fulfills the role of task manager in Figure 5)

Instead of choosing to fully automate or perform a task yourself, a level of automation (LOA) can be introduced (see Figure 6). Each LOA represents a level of autonomy and authority of the system and of the operator. For example Endsley and Kaber (1999) have described ten levels of automation that range from completely manual, via semiautomatic, to fully automatic. As the level of automation increases, the system takes on more authority and autonomy (See appendix table 5). This helps to better support two conflicting needs of the operator. The operator wants to experience a low level of workload, but still have a high level of SA. However when automation takes over (sub)tasks, the SA of the operator diminishes because he is out-of-the-loop for that particular (sub)task. We therefore do not want to completely automate or manually control a task, but choose an appropriate LOA, to reach an optimum between workload and SA.

Workload is related to 'the demand placed upon the operator'. However *experienced* workload can not exclusively be attributed to an external source but is operator specific. Each person has a unique combination of capabilities, motivations, strategies and moods which influence his experienced workload. It is not possible to deduce all these factors online (i.e. while the operator is experiencing them) or even offline for that matter. However an *indication* of workload can be based upon measurable variables.

We therefore choose to use the cognitive task load model of Neerincx, because we think we can measure or deduce the information needed for this model. In Neerincx's model three factors that influence workload are considered (2003). These are: time occupied, task switches which demand attention switches and (level of) information processing which relate to Rasmussen's levels of information processing (skill, rule and knowledge based behavior). The operator should spend as little time as possible in problem areas (see table 3). Therefore the operator should receive support before he 'enters' one of the problem areas. The goal of AA is to better suit the LOA which the operator needs at any given moment. The operator's workload should remain within an optimal range, in this way the operator is neither bored nor overloaded. This makes his work more enjoyable and less error-prone.

| | | Task Performance Period | | |
|---|---|---|---|---|
| | | Short (< 5 min) | Medium (5-20 min) | Long (>20 min) |
| Time occupied | Low | No Problem | Under-load | |
| Info processing | Low | | | |
| Task switches | Low | | | |
| Time occupied | High | No Problem | | Vigilance |
| Info processing | Low | | | |
| Task switches | Low | | | |
| Time occupied | High | Cognitive lock-up | | |
| Info processing | All | | | |
| Task switches | High | | | |
| Time occupied | High | Overload | | |
| Info processing | High | | | |
| Task switches | High | | | |

Table 3. Neerincxs factors that influence workload

## What is human machine cooperation?

The goal of our research is to develop a system that will cooperate with the operator. By this we do not mean that the system should merely be compliant, it should do more than that, the system should work together with the operator to reach a common goal. This can be achieved when the system anticipates on the actions and needs of the operator and helps when necessary. A real life example would be a surgeon assistant already having the right instruments in his hands even before the surgeon has requested them. Through years of study and experience the assistant knows what the surgeon is doing at this moment, what is going to do next *and* what instruments he will need. However we are aware that, at least in this research, we are not capable of implementing a cooperation between man and machine as well as that can exist between humans. When humans cooperate they communicate implicitly with each other and anticipate on the needs of the other. Besides intelligence and knowledge about the other, they need extensive domain knowledge in order to be able to do that. Research on training pilots to cooperate in the cockpit has shown that domain knowledge is a prerequisite to the development of the ability to cooperate. Studies have shown a correlation between domain expertise and cooperation (for example, Orasanu and Salas, 1993). For most applications where humans work together with machines, this domain knowledge and the ability to reason on it, is hard or impossible to code completely in a machine because of the sheer complexity of this knowledge.

# 3. Gathering information about the operator

Cooperation between two entities can only occur, when both anticipate on the needs of the other. And in order to anticipate on the needs of the operator, the machine needs to have a model about the state, workings and intentions of the human operator. In other words the machine needs to know in what state the operator is currently, how he (in abstract terms) gets from one state to the other, and what state he wants to achieve. The machine needs this knowledge to detect where, when and how he can support the operator. Task descriptions and the current situation in which these tasks have to be performed give an indication of the state of the operator, i.e. what he is doing at this moment and what workload the operator experiences. If the operator has already performed a certain task and has given feedback about his perceived level of workload this can be an indicator the next time the task presents itself. However tasks are sometimes more demanding than on other occasions and it is not always clear how different tasks interfere with each other. The performances on the current tasks can also be used as an indicator of the workload of the operator as well. However it only gives an indication of workload since performance is influenced by workload, but does not necessarily reflect the real workload. There are situations were a high performance is maintained, but at the costs of a high workload because the operator puts in a lot of effort. Physiological measures can also be used to estimate the workload of the operator. Many physiological measures such as heart rate variability, EEG recordings, blink rate and pupil dilation seem to correlate with workload levels. However there are tasks for which these measures dissociate from each other. It is therefore necessary to use multiple indications of workload, and use domain knowledge to favor one of the indicators when experience has shown that a particular indicator is better in determining the workload in that situation.

In this research we use information that we can obtain from the eye tracker and all input devices available to us, namely the keyboard, touch screen and mouse input. The eye tracker was not used in this research as an input device, the operator could not explicitly manipulate the system with his eyes (as is done in experiments where the focus of the eye would act e.g. as an extra mouse pointer). However the information we received from the eye tracker, and how we could use that information to improve HMC, has had much attention in our research. Besides giving psychophysiological measures such as blink rate and pupil dilation, the eye tracker also gives information about where the operator is looking.

When a machine presents information to its user via its displays, the machine knows at *what* the operator is looking when it knows *where* on its displays he is looking. The system can now deduce what information the operator has gathered from its displays, this knowledge can be used to form an estimate about the intentions of the operator. Using this estimate the system could adapt the interface so that it better suits the intentions of the operator, and it can be used to better interact with the operator. For example imagine that there are three displays with textboxes on each of them, but there is just one keyboard to control the complete system. When the operator now looks at the textbox on the first screen and starts typing, the system can deduce that this textual input is 'meant' for this textbox. It thus tries to deduce the intentions of the user and adapt its behavior accordingly. Again the focus of the operator is not meant as an explicit manipulator of the system, instead the system tries to deduce the intentions of the operator using this information and can then decide to adapt its behavior to better suit the needs of the operator. Using the information about where and at what the operator has looked, the system can also detect deficits in the information uptake of the operator, when for example a parameter on one of its displays needs the attention of the operator, but the operator has not looked at it, it can choose to let that display blink or, when more immediate actions are required, sound an alarm.

## Eye tracking

The remainder of this chapter explains what eye tracking is and what information we can distil from using it. Eye tracking is a technique which enables researchers to determine what a subject is looking at. The theory behind eye tracking is simple: by following the eye movements of the operator the system can determine where he or she is looking. When this data is combined with information about what is present at that position, you know *what* the operator is looking at. The practical side of eye tracking is not that simple though. In order to collect this data, high-precision instruments are needed and complex mathematical calculations have to be made.

In current eye movement research, there are 3 different methods in use for tracking the eye movements. Eye tracking can be done electronically by placing skin electrodes around the eyes. When the eyeballs move an electric signal is generated that is picked up and used to calculate their position. Another method uses (non-slipping) contact lenses and a magnetic field, a small spool in these lenses generates an electromagnetic signal when moving through the surrounding magnetic field, which can be detected. The third method uses cameras to track features of the eye and the head to determine the position of the eyes and head. There are systems where these cameras are mounted to a construction that is placed on the head of the subject. This ensures that the eye of the subject is fixated on the same position in the camera. While this increases the accuracy of the system, it prevents the operator from moving about in a natural way. Other

systems mount the camera on, for example, the table on which the operator is working, in this situation the operator probably does not even notice that his eye movements are being tracked.

In our research we used such a camera eye tracking system from the Swedish company Smart Eye (a screenshot of the eye tracking software can be seen in Figure 14 in the appendix). Four cameras where needed for the ICO project, because of the large visual field the operator has to monitor. Figuring out the most optimal position of the four cameras required a great deal of time, especially since normal setups of this system only use 2 cameras. As said before eye tracking systems use high-precision instruments and calculations, because of this it requires a lot of 'hands-on' practice to fully understand what works best in a particular experimental setup. Things like lighting influence the pupil dilation and can also influences the eye tracking measurements if it radiates a lot of infrared. Most eye tracking systems use infrared spotlights and infrared pass-through filters which help to differentiate the subject from the background (see figure 3). In our setup, typical "cool white" fluorescent lamps were uses as lighting which radiate only a small amount of infrared. But one also finds that simple things like using different chairs (with different heights and inclinations), different facial expressions (e.g. the operator is smiling) influence the measurements of such a delicate system. Furthermore beards and/or glasses also reduce the accuracy of the readings enormously.

When these issues have been found and resolved, the following steps are needed in order to let the eye tracking system work correctly. First the monitors, at which the operator is going to look, have to be defined in a coordinate system. Then the cameras have to be 'placed' in the coordinate system, this can be done semi-automatically. The cameras have to be calibrated, this is done with the use of a checkers board. The corners of the checkers are used to link together the images from all the four cameras. Then a profile of the operator has to be constructed, snapshots are taken from selected poses of the head. These poses should be spread as much as possible over the total visual field of the cameras. Now particular features of the head have to be marked, for example the location of the eyebrows, mouth corners, ears and eye corners of the operator.

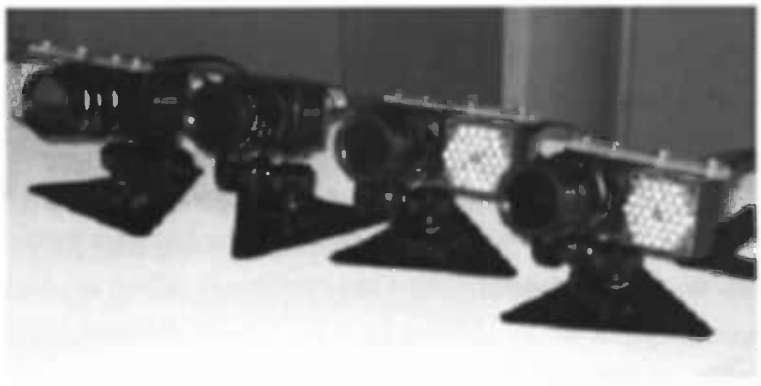


Figure 3. The cameras of the eye tracking system. On the sides of the cameras you can see the infrared emitting diodes

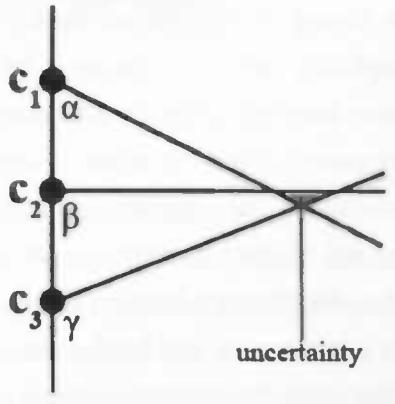



Figure 4. Triangulation.

The position of the markers in the coordinate system is determined using a geometrical method called triangulation (See Figure 4). Since the positions of the cameras are defined in the coordinate system, the angle that the markers make in each of the cameras can be used to calculate the position of the markers in the coordinate system. Now the position of the head and eye can be tracked over time.

## Eye movements

The human eye can only distinguish details when the scene of interest is projected at the fovea of the eye. Therefore the eyes move continuously in order to keep the objects of interest in focus. When the eyes have to move to another region of the visual scene this is done through a so called saccade. This is a very rapid motion (actually it is the most rapid motion the human body can perform) and only takes 30-120 ms to complete. This movement is ballistic in nature, which means that the movement is completely determined before it has begun. No feedback can be used during a saccade, because the feedback of the eyes is lagging too much (the feedback would arrive after the movement was made). A saccade is followed by a 'fixation' which last 200-600ms where the eyes are focussed at a certain part of the visual scene. However the eyes are not completely still, the eyes drift around the point of fixation and are corrected by micro-saccades when the point loses focus (Yarbus, 1967). This jitter of the eyes is filtered by our visual system and we are not aware of it. This is important to note since it means that even with a perfectly calibrated eye tracking system, the output signal will be different from what the subject thinks he sees. Therefore filtering has to be applied on the eye tracking signal as well, since instead of getting an accurate picture of what the eyes are pointing at we want information about what the operator thinks he is looking at. (See figure 5, for a record of the eyes of a subject scanning a particular scene)

The visual system is very important in our information gathering, normally the bulk of our information uptake is acquired through our visual system. As said before, the eyes have to focus an object of interest on its fovea in order to be able to see details. Therefore the eye gaze implicitly indicates the areas of the subject's attention. However objects in the periphery can enter into the subject's attention, but this is usually followed by a fixation on that point in the periphery. But even when it seems that the eyes of the subject are focussed at an object, they sometimes later can not recall that object. However if the information presented at the screen is needed by the operator and he performs so called "task-relevant" looking, one can assume that what the operator focuses on is an indication of what information the operator is taking up. Now we can measure and book-keep the following values (in addition to many others), fixation length (how long the operator focuses on a point of interest), inter-dwell times (the time it takes before he focuses again on the point of interest) and the total fixation time.

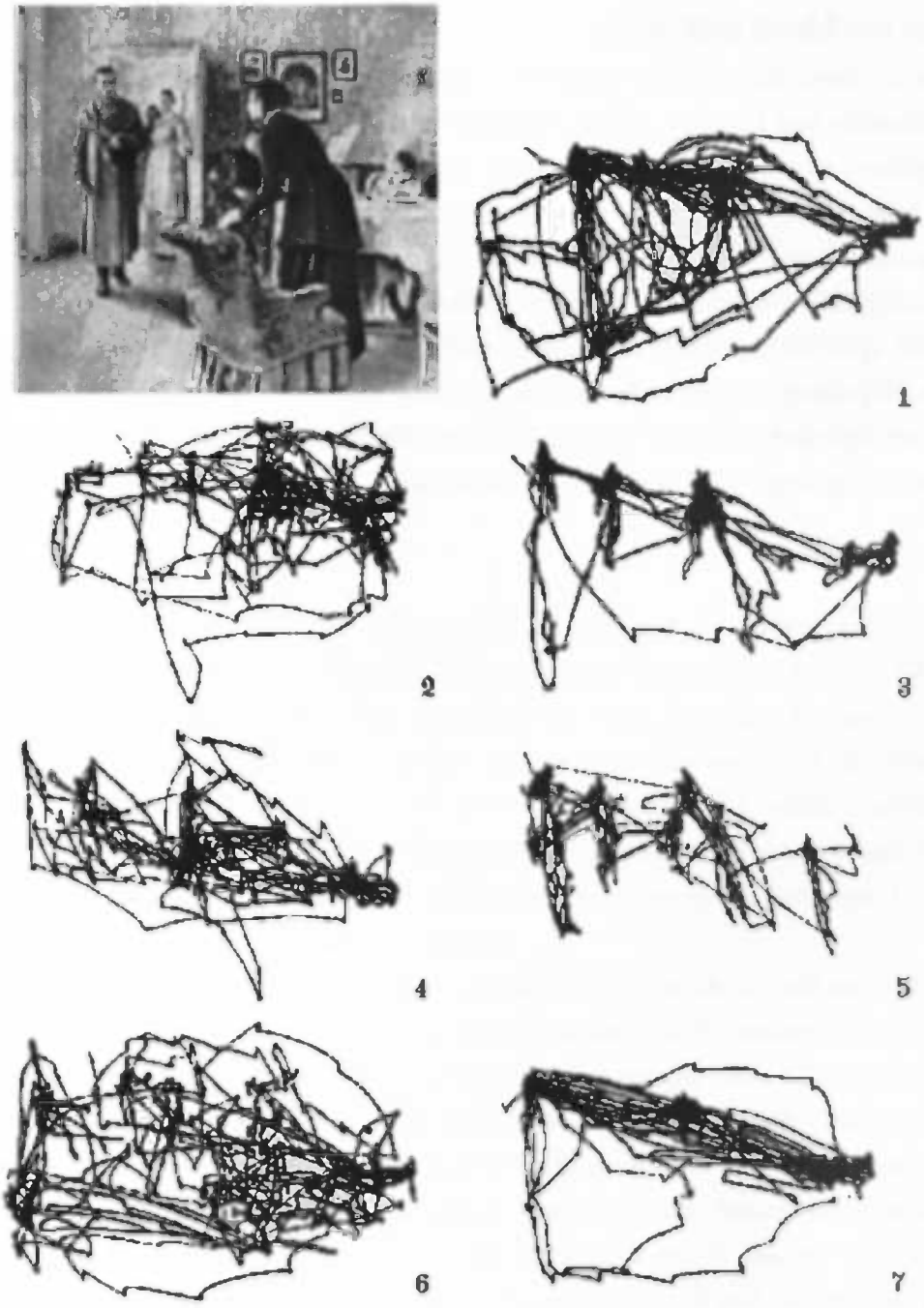



Figure 5. Seven records of eye movements by the same subject. Each record lasted 3 minutes. 1) Free examination. Before subsequent recordings, the subject was asked to: 2) estimate the material circumstances of the family; 3) give the ages of the people; 4) surmise what the family had been doing before the arrival of the "unexpected visitor;" 5) remember the clothes worn by the people; 6) remember the position of the people and objects in the room; 7) estimate how long the "unexpected visitor" had been away from the family (Yarbus, 1967).

## The eye as workload indicators

Several studies have showed that the pupil diameter is an indication of the workload of an operator (Wickens and Holland, 2000). However pupil dilation is dependent on many factors, such as lighting, eye lid closure, stress, age and the testosterone level besides information processing load. Besides that, pupil dilation was not possible to derive in this setup, since this requires equipment that can detect deviations in pupil diameter of 0.2 mm.

Eye tracking information alone is not an accurate indicator of the operator's workload; it is used to build up evidence about the state of the operator. For a better indication of workload more psychophysiological measures such as heart rate variability can be used and combined in order to build up more evidence. Subjective measures such as questionnaires and debriefings could be used to calibrate these indicators of workload.

# 4. Harnessing the complexity of a cooperating system

A system that is capable of cooperating with a human operator is inherently complex. An agent-based approach was therefore used to reduce the complexity by modeling the different functionalities which such a system should possess. In this chapter we explain *what* agents are, and *why* we have used an agent-based approach. We then discuss how a general setup of a cooperating MAS could look like.

## What is an object?

Almost a decade ago agent[2] became a buzzword in literature on computer science and an explosion of agent related articles followed. The agent hype has certainly led to a lot of fruitful research as well as to an ever growing diversion on what is meant by agents and agent technology. Agent-based programming can be seen as a further evolution of object-oriented programming (OOP), because of this they share some characteristics which contribute to the confusion between object and agent. We therefore start by explaining what an object is and work out the differences with respect to agents. OOP is a not a programming language, it is a way of thinking about software design; a so called paradigm.

When the first computers were constructed, programmers wrote their code sequentially as list of instructions, much like the recipes that you find in cookbooks. For example first initialize an integer x with value 3, *then* calculate x times 29 *then* read an integer from a file and add this to x. Then programming languages were constructed that allowed pieces of similar code to be grouped in to procedures, these procedures could be called at every location within the program. After this OOP emerged; in OOP parts of the programs that share the same functionality, which can be procedures and data structures, are grouped in individual units called objects. This functionality can now be accessed in every point in the program by an instance of that object. This means that code needed to solve the problem has to be written only once. For one thing this helps to reduce the complexity and to keep things organized. Also objects can be debugged in isolation, and multiple instances of an object can be created.

In OOP it is good programming practice to map a software object to a physical object in the real world, this makes it easier to conceptualize what its functionalities should be. Take for

[2] The word agent stems from the Latin '*agere*' which can be roughly translated as 'to do'.

example a data object called a stack, this is a last-in-first-out data structure. This object has a neat real world equivalent in one of those dinner plate wells you see in buffets, and behaves in much the same way. If you want to store a new item, you lay it on top and push down the rest. If you want to access the third item you first have to pop of the first and second item. An object is defined by its methods and its internal state, in this case it would have a method for the pop and push action and an internal state describing what and how many elements there are on the stack. Mapping an object to a real world equivalent has the benefit of being able to talk about their functionality on a high level of abstraction. Agent-based programming uses this same concept, but on an even higher level.

## What is an agent?

In agent-based programming software components are not defined terms of methods and state, instead components (agents) are defined by their behavior. Agents are defined on a higher level of abstraction with respect to objects. An agent behaves in a certain way in order to reach its goals, this autonomy is also what sets it apart from the slave-like objects.

In computer science, an agent is a distinct module which acts on behalf of itself or on behalf of another software, human or robotic agent. What exactly defines this module as being an agent is a matter of debate in the scientific community. In the literature on agents, many different definitions are used and defined, some very open, classifying almost everything as an agent, and some more strict. The reason for this diversity lies in the wide scope of applications in which agent technologies are used (Franklin & Graesser, 1996). For example agent technologies are used to simulate ant behavior but are also used in the gaming industry to simulate virtual opponents. Each of these applications has its own characteristics which call for specific features that are needed by those agents.

While there is no consensus on the definition of what constitutes an agent, there is a general understanding on the essence of an agent. In their paper "Is it an agent, or just a program" Franklin and Graesser (1996) have distillated a set of minimum requirements out of numerous definitions. They suggest that an agent is (at least):

- Persistent: agents run continuously instead of being executed on demand.
- Reactive: agents perceive their environment and react to changes.
- Autonomous: agents operate without direct intervention from other agents.
- Goal-oriented: agents take the initiative to accomplish their goals.

For example consider an agent which *continuously* monitors the stock market. While its 'owner' is at work it *autonomously* monitors the stock market. When a stock drops below a certain threshold or there is strong negative trend, it *reacts* to it. Its goal is to maximize its owners profit, it will therefore try to warn its owner and advise him to sell the stock in question. When it can not

get in touch with its owner (or it has been given full authority) it takes the *initiative* and sells the stock on behalf of him.

In this research we use the definition of Franklin and Greaser to define our agents. This still leaves room for interpretation, for example is an agent persistent (enough) if it stops running after the user has shut down its PC (or is it just being cooperative)? Or more even trickier what does it exactly mean to take the initiative to accomplish your goal? Would the stock agent in our example show enough initiative if it was not capable of selling stock in question because the first broker it asked rejected? In the American heritage dictionary to "take the initiative" is described as "the power to originate[3] something". According to such a weak definition even a simple house thermostat shows enough initiative, if it is too cold it starts up the boiler in order to heat up the room. Instead of drifting away in definitions, we choose to view agents in more open way. We use the concept of agents to structure our software in a clear way, but some of our agents are less persistent, reactive, autonomous and goal directed than others. All of them are agents in the strict sense, but some of them may not be regarded as what is 'meant' by agents by others. Russell and Norvig have put it this way: "The notion of an agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents." (1995).

## What is a multi-agent system?

Simply put when multiple agents interact they form a multi-agent system (MAS), however they start to become really interesting when the agents together are capable of reaching goals that are difficult or impossible to achieve by an individual agent. An agent which does not have methods or data available to achieve his goals has to communicate and collaborate with other agents.

When for example the stock market agent in the previous example wants to get it touch with its owner it can not do this on its own. Instead it asks the cooperation of a communication agent which does have this ability. The communication agent can then send a text message to the cell phone of the owner, but maybe it will only do this after it has asked the agenda agent to check whether its owner is not in an important meeting.

The agents in this example all have their own goals, but these goals are sub goals of a common (implicit) goal namely "support the owner". The agents will therefore always cooperate with each other in order to give the best support possible to the owner. One can distinguish two types of MAS based on the degree of cooperation between the individual agents. In cooperative MAS the agents have a common goal, and thus all act for the greater good of the system. In self-interested MAS however, agents have their own goals and are only interested in increasing their own performance even at the cost of other agents. When one of the agents interacts with an agent owned by someone else its goal is not to maximize the other agent's performance. The stock

[3] To come into being; start.

agent in our example has to be self-interested. It has to negotiate an optimal deal with other stock agents and not feel sorry for causing bad performance of its fellow agents.

## Looking beyond the hype of using agents

When technologies (or anything else for that matter) are hyped, people tend to use it just because everyone else is using it. It is therefore good to think about why and how agent technologies can help in a particular project. Agent technology provides a new way of developing software, but new ways are not necessarily always better. One can imagine software problems were the overhead of using agents is too much. When using agents in a sensible way, information and capabilities are distributed, which inherently means that communication has to take place for agents to reach a goal that depends on more than one of these distributed 'goods'. Furthermore some problems can be solved more efficiently when the information and computation is centralized because it allows for detecting larger patterns within the problem space. Think of it in this way; suppose you divide all the pieces of a 1000 piece puzzle over 10 separate agents, and let them work together to find out what the solution to the puzzle is. Now a lot of communication is needed to synchronize the constraints and possibilities of each of the 10 sub problems. Other problems can be more easily divided into smaller sub problems, suppose you want to find all the puzzle pieces that are blue(ish). In this case dividing the 1000 pieces over 10 agents, that are distributed over 10 machines, results in a tenfold increase in speed. In other words some problems lend themselves better to be 'agent-ified'.

The most important thing to realize however, is that agent-based programming is only a methodology. Expecting a solution to your problem simply by using agents will lead to sure disappointment. The 'intelligent' capabilities of the agents are not inherent properties, they still have to be developed and coded. In the end the actual agents are software programs, therefore basic software engineering practice should not be ignored when developing agents. These and other issues to consider before using agent technologies can be found in "Pitfalls of Agent-Oriented Development" (Wooldridge and Jennings, 1998)

## Why use an agent-based approach?

The agent concept is most useful when analyzing systems that are so complex that they are best described and understood on a high level of abstraction which leaves out the less important details. For complex systems it can be more natural to describe components in terms of being responsible of performing a certain task than in terms of classes and methods. Giving appropriate support to an operator means that you have to gather information about his state, deduce what his state is, reason about what support could be given best and when it should be given. This means a

divers, complex and large system is needed in order to give appropriate support, therefore a high level of abstraction is needed in order to keep an overview of the complete system.

Another benefit of the agent-based approach for this research lies in the fact that the different agents that provide support and deduce the workload of the operator, are likely to be implemented by different research groups (where one group is e.g. focused on human-machine communication, and another on operator workload). By using an agent-based approach there is already a clear separation of the functional elements from the start. Each group can define their own criteria as when to provide support, and define how they want to execute this support.

Wooldridge lists four characteristics of problems which are suitable for an agent-based approach (Wooldridge, 2002). An agent-based approach is suitable when *"the environment is open, or at least highly dynamic, uncertain, or complex"*. A system that is capable of deducing the dynamic workload of the operator based upon multiple measures, and uses this to adaptively automate task is inherently complex. Agents are used to get a grip on the complexity. Agents are suitable in cases where *"agents are a natural metaphor"*. Within the system certain behaviors can be pointed out, such as 'provide pro-active support when needed' or 'estimating the workload of the operator'. Agents can be used as a natural metaphor of entities that perform these behaviors. Abstraction of the components that work together is again helpful to get a grip on complexity.

Environments that have *"distribution of data, control or expertise"*. In order to estimate accurately the state of the operator many data sources (such as the eye tracker and task related measures) have to be combined. A task can also be dynamically automated or reallocated to another operator, when the operator is overloaded or does not have the required expertise to cope with the situation at hand.

In case of the need to use *"legacy systems"*. It is often not feasible or desirable to completely redesign all the software that the multi agent system has to work with. Therefore, certainly at the research stage, the system will have so called 'legacy' components. Components that use a different and perhaps outdated type of technology. Furthermore, as it often is the case in research, the components that make up the system are not fixed. When research shows that extra support is needed in a particular area, an extra agent can be programmed an added to the system. This flexibility of the agent-based approach is a great advantage. Also, when multiple agents work together to solve (parts of) the problem there can be an increase in (Wooldridge, 2002):

- Confidence: independently derived solutions can be cross-checked for errors.
- Completeness: agents can share their local view to achieve a better global view.
- Precision: agents can share results to increase precision.
- Timeliness: results can be derived more quickly when agents work in parallel.

Wooldridge further argues that agents in a MAS should besides being persistent, reactive, autonomous and goal-oriented be social in order to achieve its goal in cooperation with other agents. This is even his definition of an agent in general (thus not in a MAS per se), probably with the assumption that agents are most useful in MAS settings.

## Agent communication

When agents need to work together in a MAS, they need to have a way to communicate with each other. The interaction between agents is fundamentally different from the interaction between objects. Objects interact with each other by invoking their (public) methods. The object of the function being called has no control over the execution of this function, an object is a slave to its caller. Agents however are autonomous, they have full control over their inner workings. Instead of calling the function of another agent, an agent communicates by means of a message encoded in an agent communication language (ACL) they both understand. Through this communication an agent can ask another agent "Would you vacuum clean room 34?". The agent who has received this request first parses this message to determine the intent of the calling agent. Then it can decide depending on its state, goals and other plans whether it will fulfill this request.

## Speech acts

Speech act theory states that language interactions have some of the same properties as physical actions, since they too can change the state of the world. Therefore in speech there is a form of acting, for example "I now pronounce you man and wife" or "I declare war to..." changes the state of the world (Austin, 1962). Agents use speech acts in the same way as other actions to help them reach their goals. Speech acts theory started with the (posthumously published) work of John Austin in 1962 "How to do things with words". Austin distinguished 3 types of actions within a speech act the locutionary, illocutionary and perlocutionary act. Suppose you ask a friend "Would you hand me that book?", the locutionary act would be the act of uttering that sentence. The illocutionary act refers to the effect that the sender wants to achieve by uttering this sentence, in this case this would be that your friend brings you that book. The perlocutionary act refers to the actual effect that the utterance has, this might be your friend bringing you the book but also the speech act "Get it yourself" is a perlocution of the speech act.

## Agent communication languages

The most well known agent communication languages (ACLs) KQML and FIPA ACL are both based upon the speech acts theory. (The agents in this research were implemented using the JADE framework, which supports the FIPA ACL). Messages in these languages provide means to describe the beliefs, desires and intentions of the sender, which are used by the receiver as an

extra means to decode the message successfully. KQML and FIPA ACL are meta languages which mean they provide a common format for a message but not for the content. They define what could be called an envelope in which content can be placed. This envelope has slots where information regarding the message is stored, for example: the sender, receiver, language, ontology and the 'performative' that is used. The performative relates to the illocutionary act of the message, it defines the intent of the sender for the message. The FIPA ACL was constructed in order to alleviate some of the shortcomings of KQML, one of it being its rather long list of 41, seemingly ad hoc defined, list of performatives. The FIPA ACL has cut this down 20 performatives, however using the performatives can still be tricky. Careful study of what each of the performatives mean is needed, but when they are used consistently this can further improve the flexibility since the syntax of the messages is now defined. When a new agent enters the MAS, he can correctly parse the message that is send to him. In order to understand what is said to him, the semantics of the message, an ontology is needed.

### Ontologies

An ontology is a description of the concepts and relations that exists in an agent community. In order to enable knowledge sharing between different types of agents, agents from different programmers and companies, written in different programming languages, using different communication protocols and running on different hardware platform a common ground is needed. An ontology provides this through an explicit specification of the concepts that are used to describe the knowledge. For example an 'music shop' ontology could describe that a CD track has a title represented as a string, duration (time), performing artist (string) and that a CD is composed of tracks.

## A multi-agent system cooperating with the operator

Now that we have defined what an agent is, and how they can communicate with each other, we use this knowledge to create a general framework of agents that perform behaviors that have a common goal in cooperating with the human operator. This model is not meant to serve as a template for other MASs cooperating with human operators, it is meant as a general setup in which we discuss the possible behaviors and interactions between the agents. Real implementations of MASs are limited by constraints in time, complexity and equipment, in this general framework we do not have these constraints. This framework will however be used in the implementation that is discussed in the next chapter.

## Defining the behaviors of the agents

The first step in setting up a multi-agent system is to identify the overall behavior the MAS should possess, we then break up this behavior in smaller sub-behaviors until the complexity of a behavior is such that it can be carried out by a single agent. Ideally an agent should be concerned with one or more clearly defined tasks that can be seen as being a functional unit.

The overall goal of the MAS is to cooperate with the operator, the system has to work together with the operator to reach a common goal. This can be achieved when the system anticipates on the actions and needs of the operator and helps when necessary. In order to be able to do that the system has to have knowledge about the operator, the tasks at hand, and how it can best support the operator. It can build up knowledge about the state of the operator by processing data from the input devices, an eye tracker (which gives information about blink rate, pupil dilation etc.) and other psychophysiological input (such as heart rate variability, amount of transpiration etc.). Furthermore it can build up knowledge about the focus of the operator by processing the eye ball position, heading, pitch and yaw data it receives from the eye tracker agent. In the model below you can see an *eye tracker agent,* this agents is responsible for obtaining and filtering the data from the eye tracker. This agent then sends this information to a so called *operator focus agent* who processes this data in order to get to a good estimate about the current and past focus of the operator.

At this point one could ask why these two agents are not merged into a single agent. The first reason not to do this is that these two activities, acquiring and filtering of data and processing of data are functionally different. It can, for example, be very useful that the filtered data is also available to other agents in the MAS. When the two agents are not merged, the *operator focus agent* does not 'own' the filtered data and does not need to concern about distributing it to others, it is only interested in processing the filtered data. Another advantage is that we can test the workings of the eye tracker agent in separation of the other agents, or feed the operator focus agent with eye tracking data from another agent (which produces random eye movement data or has saved previous live recordings of the eye tracker agent).

When we sum up the tasks that have to exist in this framework of agents, we see that first of all there is the task of acquiring the psychophysiological measures and the data from the eye tracker. Next this information needs to be processed to determine the workload of the operator and *what* information the operator has accumulated. Also the performances on the tasks can be monitored, as this also gives an indication of workload. And then there are the behaviors that decide when and how to support or cooperate with the operator. The procedure of defining these tasks as suitable agent behaviors is a process of splitting, merging and renaming until a final, fitting, configuration of behaviors is found. Determining exactly how many behaviors are needed is not possible, since the number of behaviors and their particular responsibilities depends on the

balance between generality, efficiency, flexibility and to some extent the 'taste' of the designer. While designing a possible set of (agents performing) behaviors a diagram can give better insight into how these behaviors should work together. Figure 1 depicts a diagram of a set of agents performing the behaviors that were described here above.

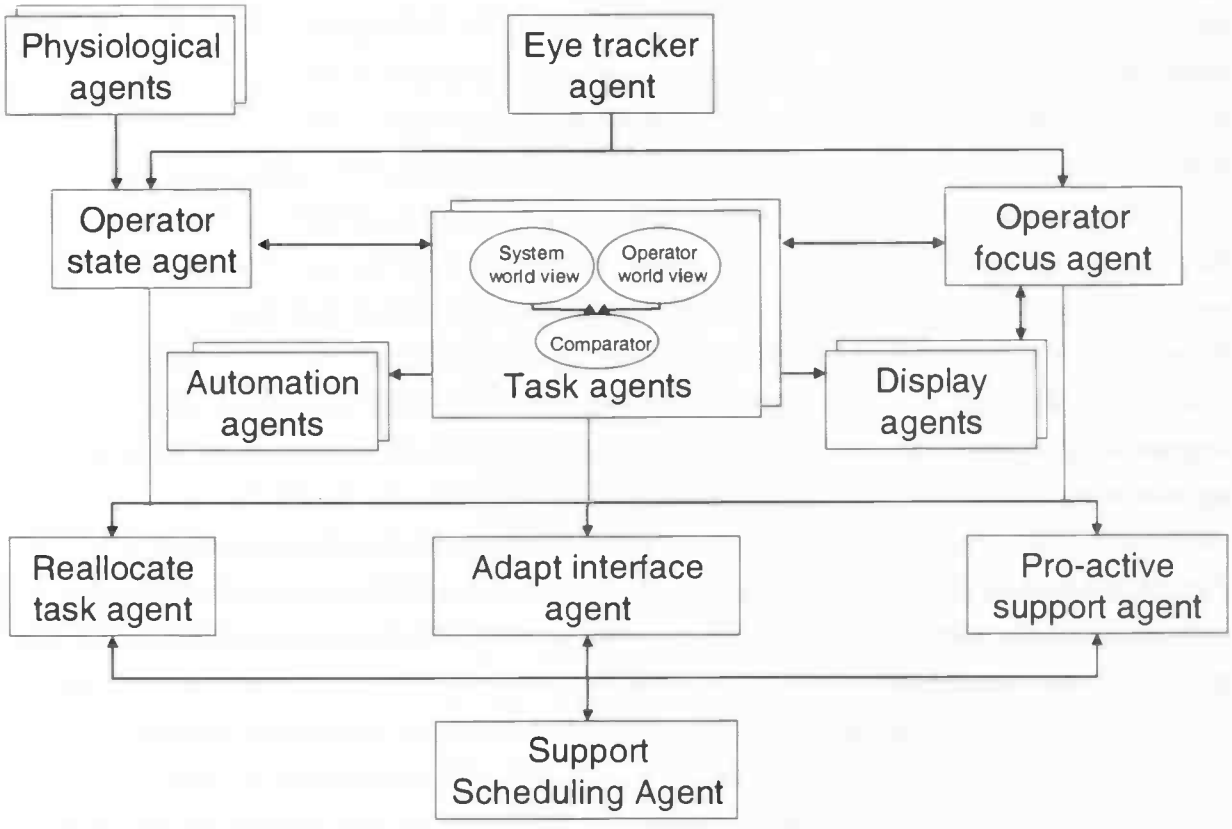



Figure 6. General framework of a MAS cooperating with a human operator

## Explanation of the agents in the model

Each of the physiological measures that are used are acquired and filtered by individual agents. The *operator state agent* [4] estimates the workload of the operator. It uses information about the physiological state of the operator, the performance on the tasks and the state of the tasks. The *task agent* monitors the task of which it is responsible. Besides monitoring the performance of the operator, it has a model of what it thinks the state of the task is, and a model of what it thinks the operator thinks that the state of the task is. The comparator detects discrepancies between the operator and system world view. These discrepancies can be a sign for the support agents to take action. When for example the operator has not noticed a parameter that is outside the desired range, the *adapt interface agent* can choose to let that display blink. Furthermore the *task agents*

[4] Names of agents are written in italics

give estimates about how demanding their tasks are at this moment. A *task agent* can adjust this estimate when it notices that its estimate is always different from the estimate the *operator state agent* has on the real workload of the operator. The *operator focus agent* determines where the operator is looking, through the use of information from the eye tracker and past information it has stored about the eye movements (which can be used to filter out noise). It then asks the *display agents* what is presented at location (x,y), when the display agent answers with a particular object, the focus agent can check whether this object is relevant to one of the tasks. This is useful because even when entities are clearly within view, if they are not central to the task at hand, they frequently remain unseen (Rensink, 2000). In order words, when an object is relevant to a task, the a priori chance that the operator has seen it, is higher. When the *operator focus agent* has determined the current focus of the operator, the corresponding task agent updates its operator world view model. The *display agents* can show information about tasks to the operator or to other team members.

In this model there are three types of support, but there could be more or other types of support, depending on the domain in which such a system would be used. The *reallocate task agent* can reallocate tasks to another team member or to an automation agent. Each of the *automation agents* can take control of a task when this is required by one of the support agents. The *adapt interface agent* can adapt the interface in order to better suit the needs of the operator or to bring certain parts of it to the attention of the operator. The *pro-active support agent* gives support when it has a good estimate about what the operator will do next and if it can assist the operator in doing that. The support agents report to a single agent, the *support scheduling agent*, how well they think they can improve the operator support. This agent then decides which of the agents can go ahead and give orders to execute the support. This setup reduces the autonomy of the support agents, however a scheduling agent may be needed to efficiently coordinate the actions of the agents. Another approach would be to let the agents work out a schedule as a group effort, however this would lead to a lot of extra communication and the agents themselves would need knowledge about scheduling and prioritizing different forms of support. It is therefore more efficient to choose for this option, and in this MAS we want the scheduling to be rather rigid and fast. We therefore make up the rules for scheduling a priori, and they are of such nature that they need no further processing. For example, if the *reallocate task agent* thinks it is necessary to reallocate a task, it will always get precedence over the other support agents, simply because you do not want to adapt the interface or give proactive support for a task that is going to be reallocated anyway.

# 5. Implementation of a cooperating system

An implementation of the general model that was described in the previous chapter was build within an existing research program at TNO Defence, Security and Safety (TNO DSS). The Royal Netherlands Navy has requested TNO DSS to investigate and develop technologies which can further support operators of command and control (C2) centers onboard naval ships. New C2 supporting techniques are needed in the navy as well in other C2 domains, for example air controllers have to deal with a large increase in air traffic and operators of video surveillance centers get more and more video *and* audio feeds which they have to monitor. But specifically C2 operators in the navy have witnessed a large increase in workload. In order to stay on top of affairs, today's navy has to deal with a large number of different missions, a growing volume *and* complexity of available data, an increasing time pressure and an increase in joint/combined missions with changing partners.

Despite this increase in workload there is also a push towards staff reduction to cut down costs. The costs of command and control systems have always been an important constraint on their design. In most settings personnel costs make up 40% of the total costs (Maas, 1999). The focus of cost reduction is therefore on decreasing the number of personnel. New concepts are needed to deal with these developments. Automation is going to play an even greater role than before, since it is needed to successfully merge the trend of increasing demands with a reduction of staffing.

Application of information technology has until now not resulted in a significant cut back of the number of employees in command information centers. This is partly because reduction of the number of manning also results in a reduction of the total human decision making capability (Maas, 1999). Thus if we can develop ways in which we can support humans in their decision making, there is a lot of efficiency to be gained by reducing the number of manning. In other words the goal is to let the operators work smarter, not harder.

## The Integrated Command Environment

TNO has setup the Integrated Command Environment program to research concepts that support operators to such an extent that smaller teams are still as capable as normal sized teams in performing C2 tasks. Traditionally personnel on navy ships have been specialized; each member onboard had its specific task that he or she fulfilled. In the current situation there are three loci of control onboard a naval ship. The ships bridge is concerned with navigation, the technical centre

with technical issues such as propulsion and the command centre is in charge of tactical decisions. Tasks and workload are divided among multiple operators which have their own specialized software and procedures and are focused on different parts of the mission at hand.

In the ICE these three centers are integrated in to one C2 environment, where all of these tasks are addressed. Bringing the operators together benefits the communication between them, now explicit as well as implicit communication (through body language) is possible. Operator tasks that were once separated are now merged into one task(set), in this process dependencies between tasks are exploited. Consider a *simple* example where a navigational task and an engine monitoring task are combined, now the operator knows about the state of the engine and can incorporate this knowledge immediately in making navigational decisions, instead of having to ask a crew member of the technical centre first. In the ICE the operators handle multiple tasks instead of one specialized task. The merging and automation of tasks allow for a reduction of manning. However further research is needed to find out if operators can remember how to perform many different tasks, and if intelligent software can adequately support them in doing so. There is also an increase in flexibility since each of the operators can take over (sub)tasks of other operators. If one of the operators gets overloaded with the tasks he has been assigned, tasks can be reallocated to either the system or another operator.

As said before, in the ICE setup, the operator has to control and monitor multiple tasks. In order to present information about all these tasks ergonomically a single 19 inch monitor will simply not suffice. Therefore the operator workstation consists of 4 screens that can be operated by touch (see figure 7).

Figure 7. Prototype of the integrated command environment.

On three large screens in front of the operators, information is projected that is relevant to all the team members. This helps to reduce unnecessary communication. In figure 14 (in the appendix) you can see how the eye tracking cameras are placed within this setup.

## Intelligent automation

One of the research goals of the ICE is to develop strategies for flexible task allocation. During operation, (sub)tasks can be delegated to other human members of the team or to virtual agents. In this setting, both human and machine work *together* to improve their overall performance. Salas et al. defined a team as "a distinguishable set of two or more people who interact dynamically, interdependently, and adaptively toward a common and valued goal/objective/mission, who have each been assigned specific roles or functions to perform, and who have a limited life span of membership" (1992). When we extend this definition to include machines, this is exactly what we want to accomplish, i.e. a mixed human-machine team. In this sense we could also interpret the "integrated" in ICE as the integration between human and machine on a functional level.

Besides team work two other themes were identified which seem crucial to improve operator efficiency and performance: support of decision making and situational awareness (SA). Decision-making is a cognitively demanding process where people have to gather information from sources relevant to the problem. In the ICE this can range from human (e.g. 'intel' reports) to electronic observers (e.g. radar). One way to support the operator is through taking over subtasks of the decision making process. When for example the operator has to decide what route it should steer the ship past hostile territory, the system could aid by presenting the best possible routes and their specific advantages (e.g. low radar reflection, time of being in hostile territory). Now the operator can save time and directly focus his attention on evaluating higher order characteristics such as strategic advantages.

As said before, in order to make (good) decisions, one would like to be fully aware of the current situation, that is have good SA. SA can be defined as "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future" (Endsley, 1988). In cases where there are many complex elements in the environment that have to be assessed, people have to consume large quantities of information in order to build up SA.

In time critical conditions this can easily lead to information overload: given the information there is too little time to find the key elements and their interrelations. In situations where this happens, humans tend to neglect or throw out data until a manageable amount is reached (Taylor, 2005). This strategy is rational from the operator point of view, but it is certainly not optimal from the ICE point of view. Operators with poor SA may find it difficult to reorient themselves to

system functioning in times of system failure or unpredicted events. The operators' performance may be compromised in situations where the automated system fails (Wickens & Holland, 2000). Therefore we would like to know when an operator is overloaded and give appropriate support.

If for example the operator has not looked at his radar display for quite some time, one can assume that his information gathering behavior is not optimal. If the system has deduced that the operator is neglecting radar data, and an interesting 'track' has appeared on radar some time ago, the system acts in an intelligent, pro-active way. Depending on the level of attention needed for this new activity, the system tries to shift the attention of the user by visual or audio cues. On the other hand, if the user is very busy and the required action of the subject is routine, such as a simple course correction to stay out of shallow waters, the system could take over this task completely, or perhaps after it has notified the operator. However the system is of course reluctant in taking more invasive actions such as weapon deployment. When an operator focuses a lot of attention to a particular object, an agent can act pro-actively and give contextual information about the object. For example when an airplane track is inspected for a period of time the agent could search for additional information in available database sources (locally, or via the internet) such as the time of take off or the type cargo it is carrying. This information could be presented like a tool tip text or be presented on a different screen according to its level of relevance. When the adapt interface agent notes that the operator is neglecting data of importance it can adapt the interface. If for example the operator is focused on the left upper corner of the radar screen and an important track appears in the right lower corner. The agent could draw extra attention to this track by letting the track 'blink', in more extreme situations it could show a 'follow-me' arrow from the upper left corner to the track of interest or even give an audio warning like: "Incoming hostile at 140 degrees".

## The implementation

The implementation that was build is based upon the framework of agents that was discussed in chapter three. The agents were constructed using the Java Agent Development Environment. JADE was designed to simplify the implementation of multi-agent systems, by providing a framework of facilities needed in most MASs. For example the communication between the agents is facilitated by JADE as well as white and yellow page services, through which agents can find each other by name or by service respectively. JADE's middleware is FIPA compliant; "FIPA specifications represent a collection of standards which are intended to promote the interoperation of heterogeneous agents and the services that they can represent" (quote from fipa.org). JADE is completely written in Java, however we programmed the agents in C# using Visual Studio 2005 because most of the software in the ICE project was programmed in C#. We could therefore benefit from previous experience with C#, but it also made it easier to reuse code

that was already available within the ICE project. There had already been done research on a (C#) program which tried to detect whether an operator neglected a track, the system thought was important (but not within an agent paradigm). There are a number of different agent frameworks, however JADE is the most used and therefore probably the best documented framework. While the implementation was based upon the general model, some of the agents were merged in order to further cut down the complexity of the system. This does not mean that the general model is no longer of importance, it was still used as an overview of the overall behavior of the system. However when implementing such a complex system, a cut down in complexity is most welcome. Of course a certain level of generality should still remain in the actual implementation, in order for it to support future additions of agents etc. A model of the agents that were implemented is depicted in Figure 3. Here you can see that the eye tracker agent is merged with the operator focus agent. The task agents are also responsible for their own automation (which was relatively easy since the tasks were simple), and their own visualization. In the general model the support agents report to a single agent how well they think they can improve the operator support and this 'scheduling' agent would decide which agent could go ahead and give the appropriate support. In this implementation, we constructed the support that each of the agents provided in such a way that there were no situations were their support could interfere negatively with each other. Because of this a scheduling agent was not needed and therefore not implemented.

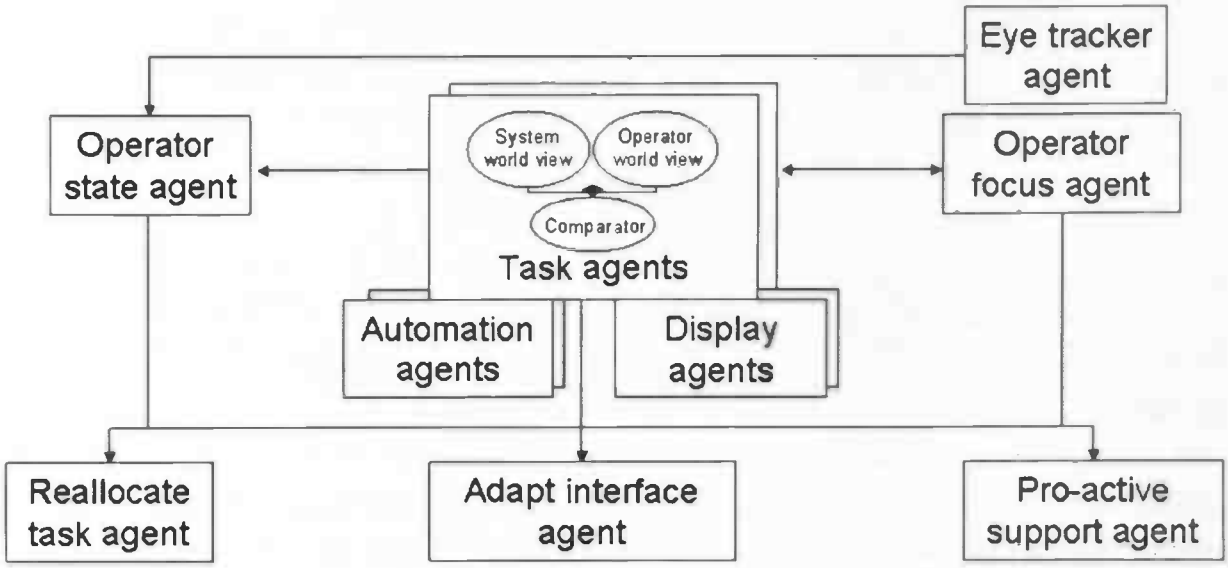



Figure 8. The implementation that was used in the ICE project

## How the agents interact

The *eye tracker agent* establishes a TCP connection with the eye tracker system, it receives information about the position of the eyes 60 times per second. This information is directly available to *the operator focus agent* because it is merged with the *eye tracker agent* into one single agent, which reduces the communication over the agent network. The *operator focus agent* filters the eye position information using a simple averaging filter over the past 200ms. It then 'asks' the *display agents* what is presented at that particular position. The JADE behavior that is defined to perform these actions is described in appendix 2, as well as the function that calculates the current eye fixation of the human operator.

The display agents (each merged into their corresponding task agent in this implementation) respond with the object of focus or a message indicating that nothing was found. This information is also passed to the *task agent* who was presenting this information to update its operator world view model.

The MAS was setup in such as way that the *operator state agent* received all information related to the workload of the operator. This included the blink frequency, the (touch screen) click frequency, error measures on the tasks, descriptions on how difficult the current tasks are and information about the velocity at which the operator moves his head from screen to screen (which can also be an indicator of workload). However it was not possible to come up with an algorithm that combines all of these measures into an accurate assessment of the workload of the operator. This because such an assessment is very hard to make in general, but it is also dependent on the tasks and the environment that operator has to deal with. Therefore an extensive empirical study on the workload of operator with respect to these tasks would be needed. There was no time to do such additional research nor was that the goal of our research.

Therefore a crude measure of workload was used based upon the three dimensions that were given by Neerincx (see chapter two). The first measure 'time occupied' is deduced by tracking how long the operator focuses on each of the screens. But since each of these tasks has a monitoring element, there is no sure way of telling when the operator has time to spare. However the time occupied measure only has a real impact on the workload when it reaches 70-80% of the total time available (Beevis, 1992). We therefore only sought to predict when this limit was reached and not to give an accurate prediction on the exact percentage of total time that the operator required. We used a simple rule: when the operator focused for more than 2 seconds on a screen that did not require attention this was seen as time that the operator had to spare. (the algorithm is described in full in appendix three). The information processing level of each of the tasks in each of the automation states was determined beforehand and used to determine the current level of information processing of each of the states. In table 4 the information processing levels are described. These values have no direct connection with a measurable quantity of

performing these task, they are used to signify that one task is more difficult that another task, and give an estimate about how much more difficult that task is.

| | Engine task | Identification task | Circle tracking task |
|---|---|---|---|
| Manual mode | 3 | 4 | 2 |
| Concur mode | 2 | 2 | 1 |
| Automatic mode | 0 | 0 | 0 |

Table 4. Levels of information processing on a scale of 0 to 10

Three levels of automation were used for tasks, manual, concur and automatic mode. When a task is performed in manual mode, the operator receives no decision support what so ever. When in concur mode the system gives advice to the operator about what to do.. In automatic mode the automation agent takes over the complete task. The task switches could be determined using the eye tracker, since each of the tasks is presented at an individual screen. Whenever the operator focuses on a different screen it is assumed that he also he switched he attention to this task, therefore a task switch can be determined. The number of task switches over 5 seconds is counted.

**The task agents**

In order to test the cooperating skills of the system, tasks had to be built to simulate a working environment in which a future operator of the ICE could work. However the tasks were made simple enough to be understood without much or any training, so that people testing this system could work with limited instructions. More importantly simple tasks are not difficult to automate, learning effects disappear more rapidly and is easier to determine when and how to automate the task. We have created simplified versions of tasks an ICE operator should perform, they are therefore not identical to tasks a 'real' operator has to perform, but since only concepts of adaptive automation were tested, this is will not negatively influence our research. On the following pages the different tasks are explained and screenshots of the tasks are provided.

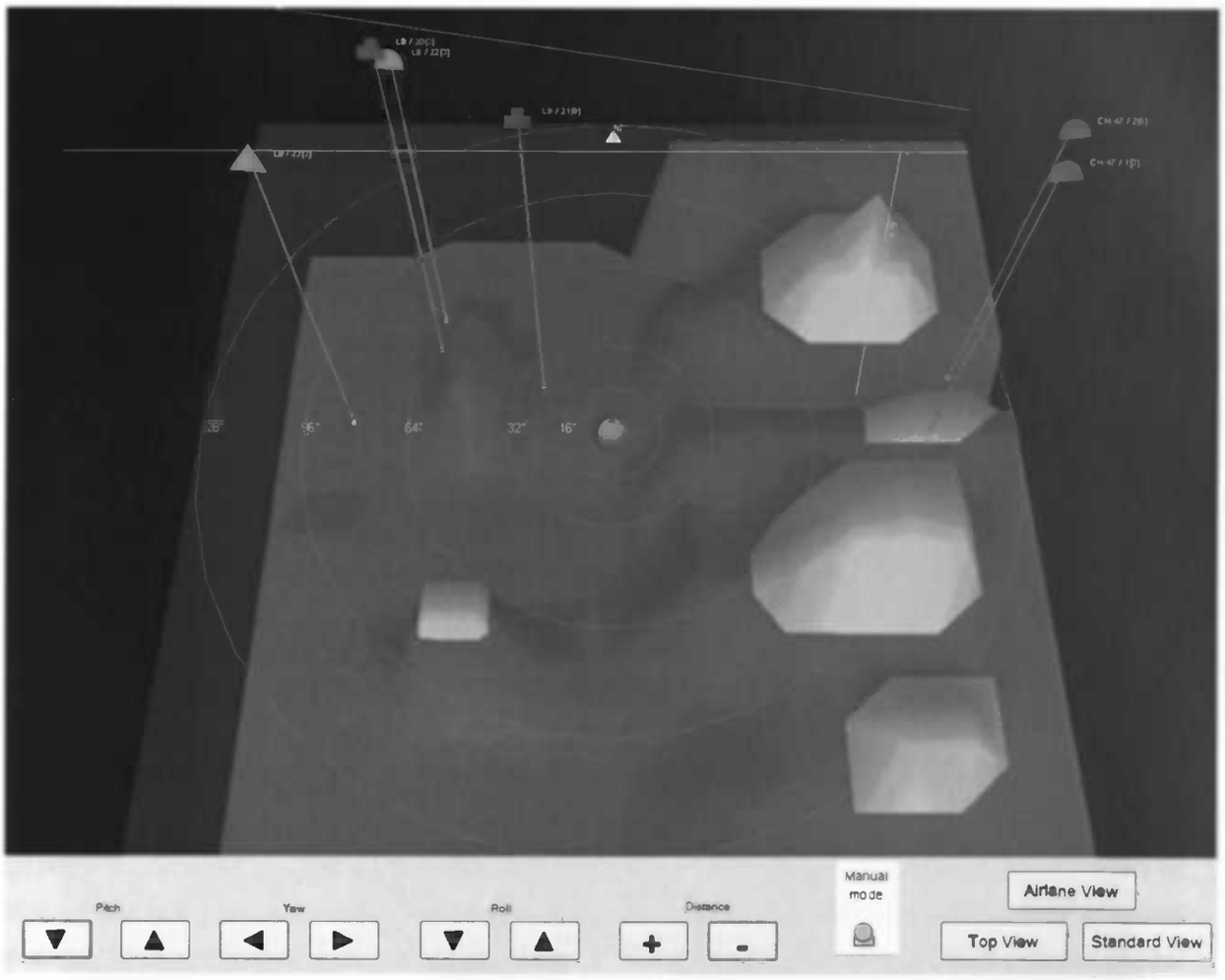



Figure 9. Screenshot of the upper screen of the identification task

The main objective of the identification task is that of identifying tracks that fly in the airspace around the ship (see figure 9). The buttons on the screen can be used to pitch, yaw and roll the 3D radar representation. The operator could also manually choose a level of automation (by clicking on the current mode, a different mode could be chosen). Additionally three common viewing points in the 3D world were easy to access through a click on the corresponding button (i.e. air lane view, top view and standard view which is the active view in figure 9). The operator has to give the tracks the correct identification. Furthermore the operator has to monitor if the tracks follow the air lane that is defined for this air space. The air lane is depicted by the two white lines in the top of the screenshot. If a track would suddenly leave the air lane and head straight for the ship (the blue dot in the center of the screen), this would make that track (at least) suspect. You can see that the track that has left the air lane has been given the suspect identification by the operator (the types of identification are shown in figure 10).

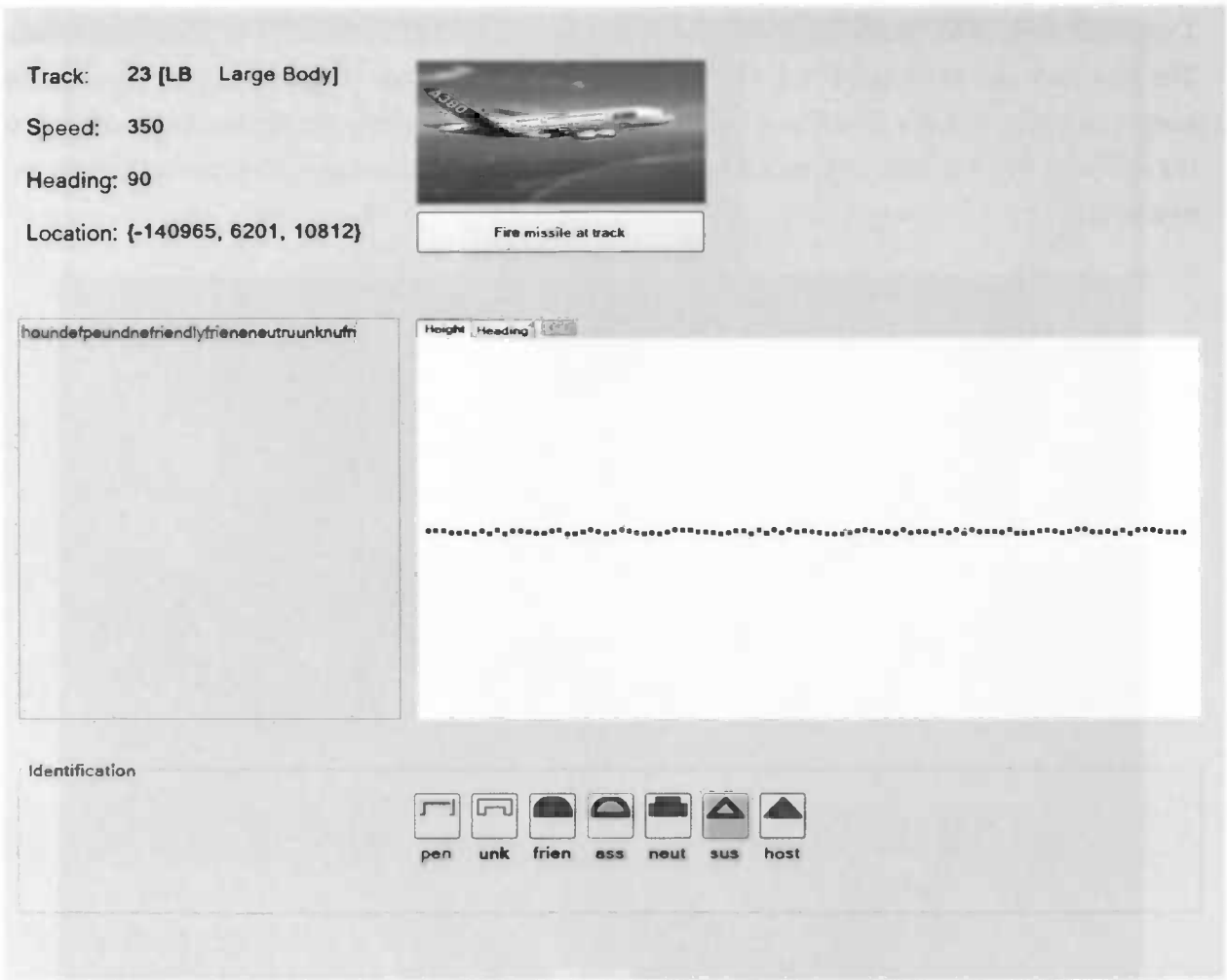



Figure 10. Screenshot of the lower screen of the identification task

This screenshot shows the second screen that is needed by the identification task. On this screen speed, heading and location of the track were presented. The identification of the track could be chosen by clicking on the corresponding button. The information about the correct identification of the track is 'hidden' in the textbox in the middle left. In this case the correct identification would be friendly (and not suspect, the identification the operator has chosen in this case). We have chosen for this approach since the full set of rules which govern identification of real tracks in a naval setting are far more complex and depend on many conditions. Our approach does not require the operator to know this rules, we therefore do not have to instruct the operator and train him in applying these rules.

Two other tasks were based upon the multi attribute task battery (Comstock & Arnegard, 1992). The first task is a skill-based task (as defined by Rasmussen, see chapter two), which requires keeping a circle within a predefined square. When the circle is within the square, the operator has the option to fire a missile at a track of choice. This task has an analogy with keeping 'locked on' to a target.

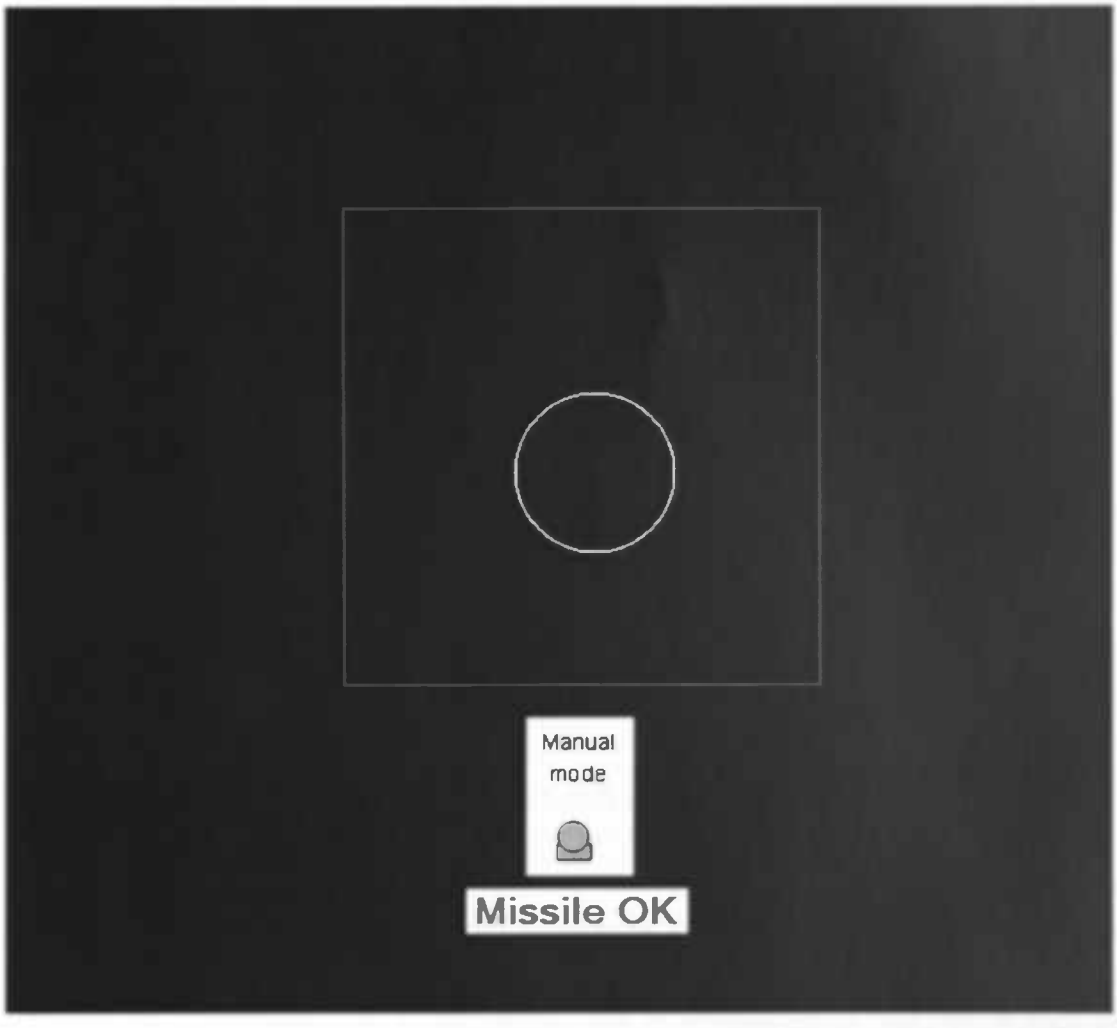



Figure 11. Screenshot of the tracking task.

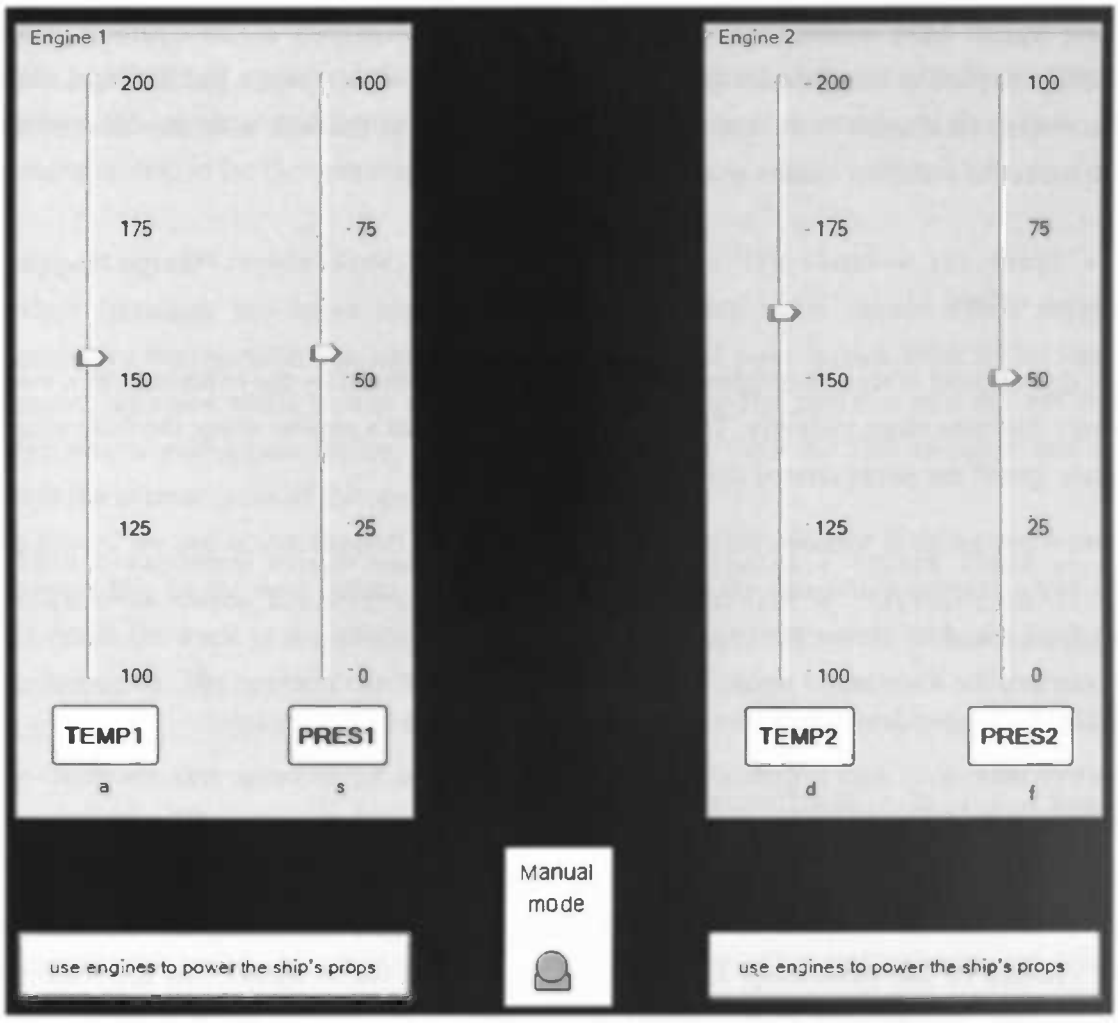



Figure 12. Screenshot of the engine monitoring task.

The second task is a rule-based monitoring task. Here the operator has to monitor the temperature and pressure of two engines. If the temperature gets too high or the pressure too low he has to intervene and give an order to fix the engines (by pushing on the corresponding button). The engines can be used for tactical maneuvers such as staying out of striking range of possible enemies. The operator has to monitor the temperature and pressure of both engines. The temperature has to remain under 175, and the pressure above 25.

The *task agents* have information about which tasks are assigned to the operator and how demanding they are at this moment (expressed as an integer value within a [0,10] range, see table 4). The engine task agent correlates the expected workload of the task with the function that is used to move the scrollbar sliders around:

```
Pos += Sin(0.3T) + Sin(1.2T) + (d_workload)*Sin(d_workload*0.6T)
(T_interval= 600ms)
```

Thus if d_workload is set to an higher level (this can be influenced by the experimenter), the scrollbars fluctuate more violently. The *trackingTaskAgent* has a similar setup, the following functions 'push' the circle around the screen:

```
Pos.X += Sin(0.0162T) + 2Sin(0.0486T) + (d_workload)*Sin(d_workload*0.0162T)
Pos.Y += Sin(0.0324T) + 2Sin(0.0810T) + (d_workload)*Sin(d_workload*0.0162T)
(T_interval= 300ms)
```

The *radarTaskAgent* determines its expected workload by the following rule:

```
Workload = 4 + (2 * numOfUnidentifiedTracks)
```

The tracks are simulated by the *scenarioAgent*. This agent is not shown in the agent diagram (figure 7) because the information this agent simulates will in a real setting be provided by the sensor systems of the naval ship. It is thus not part of the set of agents that are required for human-machine cooperation. Every second the *scenarioAgent* updates the position of the tracks and broadcasts this information to all agents that are subscribed to this information. The scenarioAgent generates a new track when a timer has elapsed a predetermined time:

```
d_triggerAirlaneTrack += 50 + d_random.Next(100);
d_triggerSpecialTrack += 50 + d_random.Next(300);
```

Which means that every 50 seconds ±100 seconds there appears a new regular track (which will have a set course within the approved airlane), and every 50 seconds ±300 seconds there will appear a track that will exhibit hostile behavior (and thus requires extra attention from the operator).

The task agents have an operator world view model, in which information is stored about what information the operator has gathered about this task and how long he has focused on that information (for example, has the operator seen track X, and how long has he looked at it). The system world view consists of the information that the system has about the world, this includes

information about tracks, engines etc. Tracks are data structures that contain information, for example heading and speed, about objects of tactical interest such as jetfighters, ships and submarines. The system world view can also give an indication of workload, since more tracks mean more workload for the operator.

**The support agents**

The adapt interface, pro-active support and reallocate task agent decide which support is appropriate for the operator. The adapt interface agent would make a track blink if that track did not receive attention while it was not yet identified. When the operator still did not look, a message box would appear telling the operator to look at track X. This message box would appear at the current focus of the operator.

The role of the pro-active support agent is to deduce what the operator is doing and whether it can support him in the near future. In this implementation the pro-active support agent would already select the track in the information screen when the operator would look at a track that is not yet identified. The operator can now directly give identification to the track without having to click on it.

The reallocate task agent is the agent in control of reallocating a task to another (virtual or human) operator. When for example the operator is too busy, tired or making too many mistakes, a task can be reallocated to another operator who is less busy or tired and has the right competences. In situations where the task in question can be automated (which is the case for all the implemented tasks), an agent can also take over (part of) a task, thereby relieving the operator. The reallocate task agent would set the level of automation of each of the tasks according to the current workload of the operator. The algorithm to decide *what* and *when* a task should be automated, and to what degree, is as follows (in pseudo code):

```
//Increase automation
If(TimeOccupied > 7, InfoProcessing > 7)
      If(EngineTaskAgent.automationlevel < 3)
            EngineTaskAgent.automationlevel++;
      Else if(TrackingTaskAgent.automationlevel < 3)
            TrackingTaskAgent.automationlevel++;
      Else if(radarAgent.automationlevel < 3)
            radarAgent.automationlevel++;
      Else
            Show message "I will ask another teammember to assist"
Else If(TaskSwitches > 7)
       If(focusOnEngineTask)
              If(TrackingTaskAgent.automationlevel < 3)
                  TrackingTaskAgent.automationlevel++;
```

```
            Else If(EngineTaskAgent.automationlevel < 3)
                EngineTaskAgent.automationlevel++;
            Else
                //operator has only one task left, and
                //still shows many task switches
                Inform ICE supervisor
        Else If(focusOnTrackingTask)
            If(EngineTaskAgent.automationlevel < 3)
                EngineTaskAgent.automationlevel++;
            Else If(TrackingTaskAgent.automationlevel < 3)
                TrackingTaskAgent.automationlevel++;
            Else
                //operator has only one task left, and
                //still shows many task switches
                Inform ICE supervisor
//Decrease automation
If(TimeOccupied < 3, InfoProcessing < 3)
        If(EngineTaskAgent.automationlevel > 1)
            EngineTaskAgent.automationlevel--;
        Else if(TrackingTaskAgent.automationlevel > 1)
            TrackingTaskAgent.automationlevel--;
        Else if(radarAgent.automationlevel > 1)
            radarAgent.automationlevel--;
        Else
            //operator still experiences underload
            Inform ICE supervisor
Else If(TaskSwitches < 3)
        If(focusOnTrackingTask)
            If(TrackingTaskAgent.automationlevel > 1)
                TrackingTaskAgent.automationlevel--;
            Else If(EngineTaskAgent.automationlevel > 1)
                EngineTaskAgent.automationlevel--;
            Else
                //operator still experiences underload
                Inform ICE supervisor
        Else If(focusOnEngineTask)
            If(EngineTaskAgent.automationlevel > 1)
                EngineTaskAgent.automationlevel--;
            Else If(TrackingTaskAgent.automationlevel > 1)
                TrackingTaskAgent.automationlevel--;
            Else
                //operator still experiences underload
                Inform ICE supervisor
```

In this algorithm the 'Neerincx' workload measures (see chapter 2) determine when to increase or decrease the level of automation.. If (only) the task switches are high, but the information processing and time occupied are not, this is a indication that the operator is busy but he is not overloaded. Therefore one of the simple tasks is automated, which means that the operator will have to perform less task switches. Extra information from the eye tracker is used to determine which task should be chosen for a change in automation level. If the operator is looking at the tracking task, we can decrease the automation level, because this change will be noticed immediately by the operator since he is looking at that particular screen. But we do not want to increase the automation level of a task presented at a screen, on which the operator is focussing. Since this would mean that operator suddenly has to quit the task which he was performing, and then switch to a new task. Instead we want to automate tasks which do not have the current attention of the operator, this would mean that automation would take over tasks, in cases of operator overload, which the operator would neglect or give less attention to.

The *reallocate task agent* would also monitor if the *eye tracker agent* was able to determine the position of the operator's head. Since if the eye tracker agent was not able to do so, this could mean that the eye tracking system was not able to find the operator while he was there or that the operator was no longer present behind 'the controls'. If besides this no keyboard or touch screen activity is measured we want the system to respond, in our setup the agent would, after 50 seconds, show a message box stating that the systems has difficulties detecting his presence and the operator is asked to move his head closer to the monitors. If after another 50 seconds the operator still was not detected, the *reallocate task agent* would set the automation level of all tasks to automatic mode and inform the ICE supervisor about its actions.

## Interaction with the system

Much of the behavior of the system can be deduced when reading about the behaviors of the agents in the previous paragraph. While a thorough empirical validation of the system was left as future research, the system was tested by a small group of fellow researchers. These trials showed that the eye tracker provides valuable information for building a model about the information uptake of the user. Users of the system found it interesting to see that a track started to blink whenever they were not paying attention to it. Or, that the system would warn the operator that it could not detect his presence, which would happen repeatedly in cases were the operator would talk about his findings with other researchers and neglect the task for more than 50 seconds. This behavior made the users feel that the system was intelligent, and knew what they were doing.

The use of our measure for operator workload was as expected crude, in some cases it increased the level of automation while this was not needed. Furthermore noise from the eye tracking system would sometimes give inaccurate measures too, which would then lead to an

erroneous change in the automation level. It proved very hard to get a stable measure which we could use to direct the cooperation with the human operator. When we would fake the workload measurements by estimating the workload of the operator ourselves and then feed this information to the MAS, the system would give the appropriate support, and it gave the operator the sense that it was cooperating with him.

# 6. Discussion

In this research a MAS was built that was capable of human-machine cooperation. Implementing this MAS and the infrastructure needed by the MAS required a large amount of work and time. We therefore had to leave a thorough empirical analysis of our system as future research. However the system was tested by a small group of fellow researchers and these trials showed that the eye tracker provides valuable information for building a model about the information uptake of the user. In the following paragraphs we discuss the findings of our research.

## Using psychophysiological measures is difficult

Besides giving information about the information uptake of the operator, the eye tracker can also provide extra clues for estimating the workload of the operator. However none of these measures, neither in this research nor in any research done so far, are accurate enough to give an estimate of the workload on its own. As other research has suggested (e.g. Scerbo, Freeman et al., 2001), data from different psychophysiological measurements have to be integrated to reach a more accurate estimate, however exactly which measures and in what way this should be done needs *much* further research. During this research it became clear that trying to define a measure, or a combination of measures that will accurately estimate the workload is extremely hard. And it may well be impossible to find a measure that is also task independent, and even person independent. We found that using only psychophysiological measures made the system very sensitive to the task at hand, the characteristics of the operator, noise produced by the recording devices or by the experiment surroundings. We therefore think that future developments should focus on a more robust way of trying to deduce workload. This can be done by incorporating more information about the factors that can contribute to the expected workload instead of trying to measure the experienced workload through the perceived workload, see Figure 13.

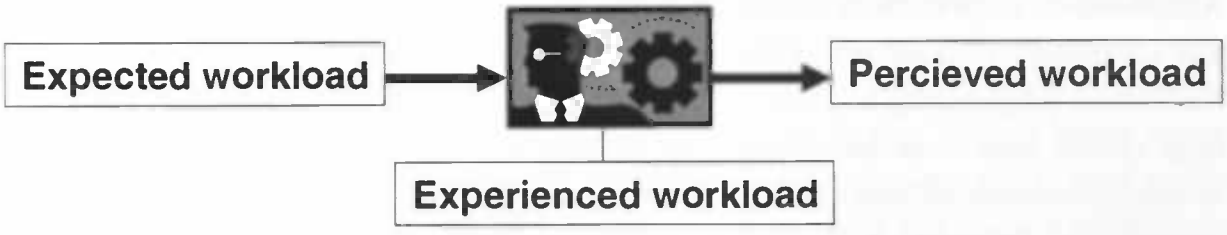



Figure 13. 'Real', experienced and perceived workload of the operator

The expected workload is determined by all the information we have about the tasks (e.g. the number of tasks and their difficulty), the environment the operator is in, and the state of the operator himself. However this expected workload estimate will always deviate from the workload the operator really experiences (the experienced workload). The perceived workload is the estimate of the experienced workload we get from observing the human operator. The perceived workload has an inherent delay, which means that you can only indicate that the operator is overloaded at the moment that he is already overloaded. Through training most operators become flexible with respect to variations in workload, and can therefore cope with e.g. a higher workload for a certain period of time, without showing direct signs of higher workload (the perceived workload remains the same). But it is of course not desirable that automation only implements additional support when the operator finally 'buckles' under a higher workload and starts to show signs of overload.

## Eye tracking gives valuable information, but better resolution is needed

Eye tracking technology has improved enormously, only a few years ago multiple feature tracking and position calculation at 60Hz was certainly not possible. Now that it is possible it has improved the spatial resolution of the eye tracker. With the Smart Eye Pro system we could achieve an accuracy of about 3-4 degrees in best case scenarios. However better operator support can be given when the resolution would be even higher. Most display 'items' are smaller than 3-4 cm (which will occupy 3-4 degrees at a distance of 60cm), but in most cases this resolution could not be achieved, which would mean that it was not possible to determine at what object the operator was focusing, but rather at what part of the screen the operator was looking. This meant that there were cases where the system could not determine whether the operator was paying attention to a particular object, while the operator did in fact do so. This would lead to unnecessary alarming the operator that a track or task needed attention. But for other measures the eye tracker provides valuable information, for example the time occupied per task and the number of task switches (used by the Neerincx estimate of workload) could be estimated using the information from the eye tracker.

## Agent-based approach is useful

The underlying problems and challenges of human-machine cooperation are very complex. Furthermore programming in general can be difficult, and easily becomes very difficult when large systems have to be built. When also the data *and* computation is distributed, it easily becomes enormously difficult to develop a system where the software components cooperate in a coherent way. Debugging also becomes hard because multi-threading tends to introduce non-deterministic behavior in the system. A certain error can occur one time, but does not need to

occur the next time you repeat the *same* routine. This is because the different threads are not always started at the same time, and the time scheduled to each of the threads also depends on other processes running on the computer. But it is specifically in these situations were agent-based programming shows its worth. It helps the programmer to keep an overview of all the different pieces of software that work together. One can debug the system on a larger level of abstraction, e.g. agent X behaves in such a way and this could interfere with the behavior of agent Y. Summarizing, an agent-based approach helps to decompose the overall 'problem' into smaller sub problems, and it abstracts away unimportant details when one needs an overview of the complete system. And this overview is needed again, to setup the organization and communication of the agents in the MAS. We must stress that this reduction of complexity is immensely important when constructing complex systems.

## Future research

In order to be able to make strong conclusions with respect to the effectiveness of the MAS build in this research, a thorough empirical validation has to take place. Essentially most ingredients to perform such a research are already constructed in this research. A future empirical analysis should test the system in three modes, without any computer support, with adaptive automation, and with human-machine cooperation. The operator will then be presented with a scenario where the workload will steadily increase until he can no longer cope with the demand placed upon him. We want to find out at what point the operator 'buckles' under the workload, in the situation without computer support, with AA and HMC. We can then see what the added benefit of computer support is, and whether our approach to HMC is better than AA.

Furthermore, how the operator will learn to cooperate with a support providing MAS will be interesting to research. Will the algorithm described in chapter six cause the operator to become complacent and think "Well if things get really hectic and I get overloaded I just wait for the automation to clean up the mess", if so, is this we want? Thus besides that research is required on the technical issues of human-machine cooperation, there is also research needed on how automation can be incorporated within an existing (in this case strict hierarchical) organisation. For example, how much authority do we grant the automation? How will the operator interact with a machine that may have more authority then he does? Automation will behave differently from how a human would behave, this is exactly the reason why we automate certain tasks, however human operators will expect (to a certain degree) human-like interaction from the automation, certainly when the automation behaves in an intelligent way. To what degree should we adapt the automation to give human-like feedback, show emotion etc. and to what degree should we instruct the operator how the automation works in detail and where it differs from what you would expect from a human team member?

## Conclusion

Our research question was stated as follows: *What are the properties of an architecture capable of human-machine cooperation?* In this thesis we have researched this question by building a system that was capable of a (crude form) of human-machine cooperation. In order to let man and machine cooperate with each other, they have to have an understanding of the intentions, state and workings of the other. This means that the interface of the system must be so that the operator can easily discern in what state the system is, and he needs training in how the system works. The information a system normally gets through the interaction of the operator with the systems buttons and pointing devices is limited. The system needs information of a more continuous form if it is to deduce the state of the operator. Psychophysiological measures seem like a candidate however there is not one single measure that is accurate enough and a method which joins multiple psychophysiological measures together to increase accuracy is not available yet. We have proposed that besides the five criteria of Parasuraman et al., two other criteria can be helpful to determine the state of the operator, task requirements and deficits in information uptake. Task requirements give a more stable measurement of the state of the operator and can perhaps successfully be incorporated with psychophysiological measures. Deficits in information uptake gave valuable information for the system to decide if it should warn the operator, or even completely take over a task. Besides knowledge about the operator and intelligence needed to reason on this knowledge, the system needs extensive domain knowledge in order to be able to know when and how to support the operator. We thus need a complex system that has to fulfill a collection of different tasks, when building such a system a reduction in complexity is needed. In this research an agent-based approach was very helpful in decomposing and abstracting the system.

Concluding, the system that was built incorporated knowledge from many different domains. However many of those domains are not 'mature' yet and need further research. For example much research is needed on the subject of deducing operator workload. Also more research is needed on the subject of adaptive automation. It seems that AI is going to play a big role in achieving human-machine cooperation, because a system that is capable of cooperating has to have AI capabilities such as learning, reasoning and planning. But again, also more research is needed on those areas. We therefore have to conclude is that human-machine cooperation still needs a lot of research, but HMC is essential if we want machines to take over more work.

# References


Anderson, J., & Gluck, K. (2001). What role do cognitive architectures play in intelligent tutoring systems? In K. S. Carver (Ed.), *Cognition & Instruction: Twenty-five years of progress* (pp. 227-262): Lawrence Erlbaum Associates Publishers.

Austin, J. (1962). *How to do things with words*. Cambridge: Harvard University Press.

Bainbridge, L. (1983). Ironies of automation. In *Automatica* (Vol. 19, pp. 775-779).

Bellifemine, F., Caire, G., Trucco, T., & Rimassa, G. (2006). JADE programmers's guide [Electronic Version]. Retrieved 2-9-2006 from http://jade.tilab.com/doc/programmersguide.pdf.

Bellifemine, F., Caire, G., Trucco, T., Rimassa, G., & Mungenast, R. (2006). JADE administrator's guide [Electronic Version]. Retrieved 2-9-2006 from http://jade.tilab.com/doc/administratorsguide.pdf.

Billings, C. (1997). *Aviation automation : the search for a human-centered approach.* Mahwah: Lawrence Erlbaum Associates Publishers.

Comstock, J., & Arnegard, R. (1992). Multi-attribute task battery *NASA Technical Memorandum 104174.*

Degani, A. (2003). *Taming Hal: Designing interfaces beyond 2001.* New York: Palgrave MacMillan.

Endsley, M. (1988). Design and evaluation for situation awareness enhancement. *Proceedings of the Human Factors Society 32nd Annual Meeting*, 97-101.

Endsley, M., & Kaber, D. (1999). Level of automation effects on performance, situation awareness and workload in a dynamic control task. *Ergonomics, 42*, 462-492.

Fitts, P. (1951). *Human engineering for an effective air navigation and traffic control system.* Columbus.

Gruber, T. (1995). Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human-Computer Studies, 43*, 907-928.

Hoc, J. (2001). Towards a Cognitive Approach to Human-Machine Cooperation in Dynamic Situations. *International Journal of Human-Computer Studies, 54*(4), 509-540.

Hoc, J., Cacciabue, P., & Hollnagel, E. (1995). *Expertise and technology: Cognition & human-computer cooperation.* Hillsdale: Lawrence Erlbaum Associates.

Hoc, J., Mars, F., Milleville-Pennel, I., Jolly, E., Netto, M., & Blosseville, J. (2006). Evaluation of human-machine cooperation modes in car driving for safe lateral control in bends: function delegation and mutual control modes. *Le Travail Humain 69*, 153-182.

Hoffman, R., Feltovich, P., Ford, K., Woods, D., Klein, G., & Feltovich, A. (2002). A Rose by Any Other Name...Would Probably Be Given an Acronym. *IEEE Intelligent Systems 17*(4).

Hollnagel, E., & Woods, D. (1983). Cognitive Systems engineering: new wine in new bottles. *International Journal of Human-Computer Studies, 18*, 583-600.

Kaber, D., & Endsley, M. (2004). The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theoretical Issues in Ergonomics Science, 5*, 113-153.

Kaber, D., Riley, J., Tan, K., & Endsley, M. (2001). On the design of adaptive automation for complex systems. *International journal of cognitive ergonomics, 5*, 37-57.

Kruit, V. (2003). *Adaptive autonomy in Command and Control decision-making.*

Maas, H., & Keus, H. (1999). A Methodological Approach to the Design of Advanced Maritime Command & Control Concepts. *Proceedings of the 1999 Command and Control Research Technology Symposium.*

Neerincx, M. (2004). Cognitive task load analysis: allocating tasks and designing support. In E. Hollnagel (Ed.), *Handbook of Cognitive Task Design.*

Norman, D. (1990). The "problem" of automation: Inappropriate feedback and interaction, not "over-automation". In D. Broadbent, A. Baddeley & R. J. (Eds.), *Human factors in hazardous situations* (pp. 585-593). Oxford: Oxford University Press.

Nwana, H., & Ndumu, D. (1999). A Perspective on Software Agents Research. *The Knowledge Engineering Review, 14*, 1-18.

Orasanu, J., & Salas, E. (1993). Team decision making in complex environments. In A. Klein, J. Orasanu, R. Calderwood & C. Zsambok (Eds.), *Decision making in action: Models and methods* (pp. 327-345). Norwood: Ablex Publishers.

Parasuraman, R. (1993). Effects of adaptive function allocation on human performance. In D. Garland & J. Wise (Eds.), *Human factors and advanced aviation technologies* (pp. 147-157). Daytona Beach: Embry-Riddle Aeronautical University Press.

Parasuraman, R. (2000). Designing automation for human use: Empirical studies and quantitative models. *Ergonomics, 43*, 931-951.

Parasuraman, R., & Miller, C. (2004). Trust and etiquette in high-criticality automated systems. *Communications of the Association for Computing Machinery, 47*(4), 51-55.

Parasuraman, R., & Mouloua, M. (1996). *Automation and human performance: Theory and applications*. Mahwah: Lawrence Erlbaum Associates Publishers.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse and abuse. *Human Factors, 39*, 230-253.

Rasmussen, J. (1983). Skills, rules, knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics, 13*, 257-266.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. New York: Elsevier Science Inc.

Rensink, R. (2000). Seeing, sensing and scrutinizing. *Vision Research, 40*, 1469-1487.

Rouse, W. (1976). Adaptive allocation of decision making responsibility between supervisor and computer. In T. Sheridan & G. Johannsen (Eds.), *Monitoring behavior and supervisory control* (pp. 295-306). New York: Plenum Publishing Corporation.

Rouse, W. (1988). Adaptive aiding for human/computer control. *Human Factors, 30*(4), 431-443.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Upper Saddle River: Prentice Hall.

Salas, E., Dickinson, T., Converse, S., & Tannenbaum, S. (1992). Toward an Understanding of Team Performance and Training. In R. Swezey & E. Salas (Eds.), *Teams: Their Training and Performance* (pp. 3-29). Norwood: Albex Publishers.

Scerbo, M. (2001). Adaptive automation. In W. Karwowski (Ed.), *International encyclopedia of ergonomics and human factors* (pp. 1077-1079). London: Taylor & Francis.

Scerbo, M., Freeman, F., Mikulka, P., Parasuraman, R., Di Nocera, F., & Prinzel, L. (2001). The efficacy of psychophysiological measures for implementing adaptive technology. *Technical Paper NASA/TP-2001-211018*.

Sheridan, T. (2000). Function allocation: Algorithm, alchemy, or apostasy? *International Journal of Human-Computer Studies, 52*, 203-216.

Sheridan, T., & Parasuraman, R. (2006). Human-automation interaction. *Reviews of Human Factors and Ergonomics, 1*, 89-129.

Sibert, J., Gokturk, M., & Lavine, R. (2000). *The reading assistant: eye gaze triggered auditory prompting for reading remediation*. Paper presented at the 13th annual ACM symposium on User interface software and technology

Taylor, S. (2005). *How Much Information is Enough?* Paper presented at the 10th International Command and Control Research and Technology Symposium., McLean.

Wickens, C., & Holland, J. (2000). *Engineering psychology and human performance (3rd edition)*. New York: Harper Collins.

Wilson, G., & Russell, C. (2003). Real-time assessment of mental workload using psychophysiological measures and artificial neural networks. *Human Factors, 45*, 635-643.

Wooldridge, M. (2002). *An introduction to MultiAgent Systems*. Chichester: John Wiley and Sons Ltd.

Wooldridge, M., & Jennings, R. (1995). Agent Theories, Architectures, and Languages: a Survey. In M. Wooldridge & R. Jennings (Eds.), *Intelligent Agents* (pp. 1-22). Berlin: Springer-Verlag.

Wooldridge, M., & Jennings, R. (1998). Pitfalls of agent-oriented development. In K. Sycara & M. Wooldridge (Eds.), *Proceedings of the Second International Conference on Autonomous Agents* (pp. 385-390). Minneapolis: ACM Press.

Yarbus, A. (1967). *Eye movements and vision*. New York: Plenum Press.

# Appendix

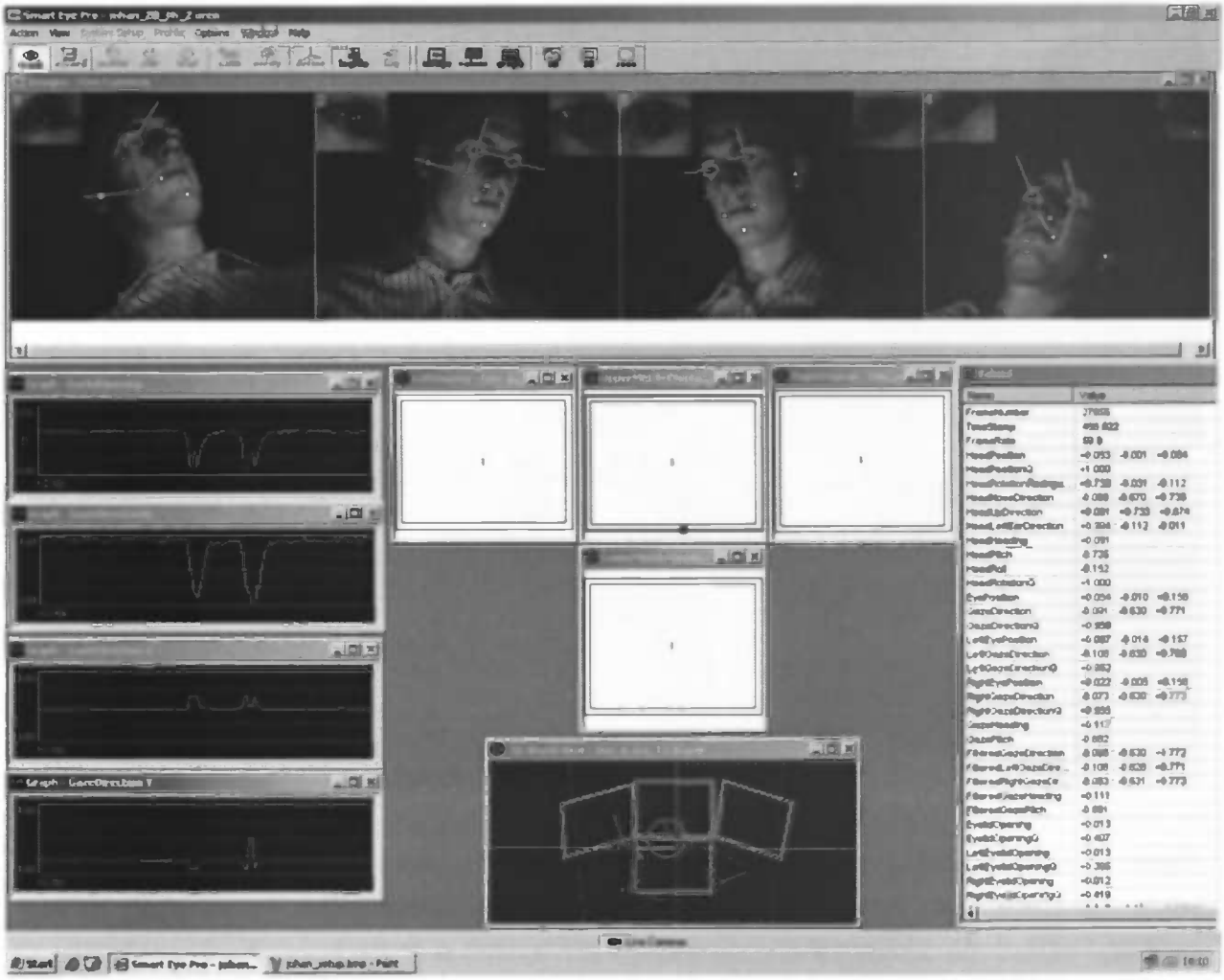


Figure 14. Screenshot of the eye tracking software. Here you can see the video feed from all 4 the cameras, and the estimation of the system about the orientation of the head and eyes. On the left the graphs represent the values of the eye lid opening, quality of gaze direction and the gaze direction in the x and y dimension. One can immediately see that the quality of the estimation of gaze direction goes down when the eyes are closed, as is to be expected. In the middle the four screen of the ICE setup are depicted, and the gaze of the operator is indicated. The table on the right show additional parameters that are calculated by the eye tracking software.

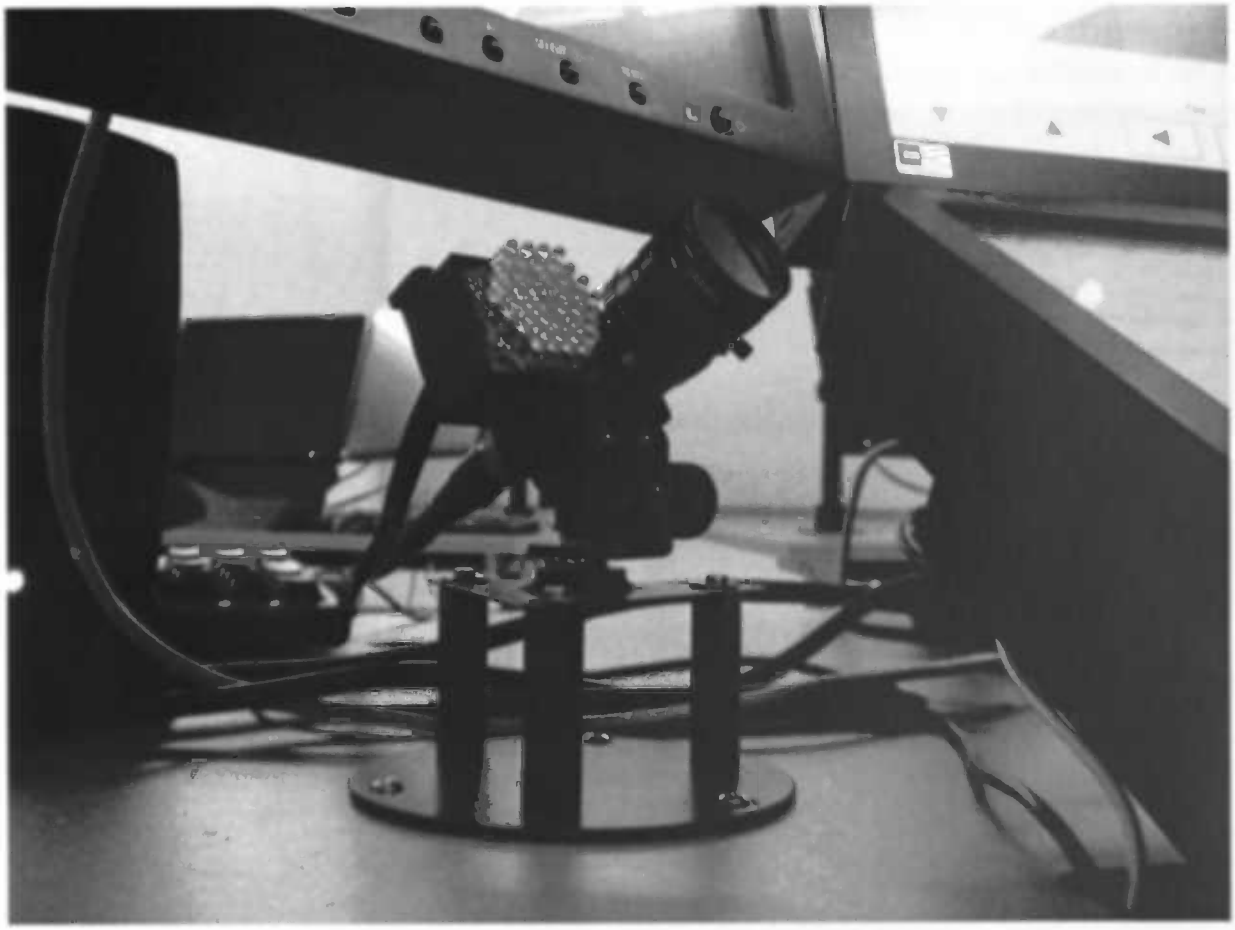


Figure 15. Photo on which you can see how the eye tracking cameras are placed within the ICE setup. The cameras are positioned in such a way that the operator will not accidentally hit his hand against them (under normal operation of the ICE system of course).

1) Manual control (MC) – the human performs all tasks including monitoring the state of the system, generating performance options, selecting the option to perform (decision making) and physically implementing it.
2) Action support – at this level, the system assists the operator with performance of the selected action, although some human control actions are required.
3) Batch processing – although the human generates and selects the options to be performed, they are then turned over to the system to be carried out automatically.
4) Shared control – both the human and computer generate possible decision options. The human still retain full control over the selection of which option to implement; however, carrying out the actions is shared between the human and system
5) Decision support – the computer generates a list of decision options that the human can select from or the operator may generate his or her own options. Once the human has selected an option, it is turned over to the computer to implement.
6) Blended decision making – at this level the computer generates a list of decision options that it selects from and carries out if the human consents. The human may approve of the computer's selected option or select one from among those generated by the computer or operator.
7) Rigid system – this level is representative of a system that presents only a limited set of actions to the operator. The operator's role is to select from among this set. He or she may not generate any other options.
8) Automated decision making – at this level, the system selects the best option to implement and carry out that action, based upon a list of alternatives it generates (augmented by alternatives suggested by the human operator
9) Supervisory control – at this level the system generates options, selects the option to implement and carries out that action. The human mainly monitors the system and intervenes if necessary.
10) Full automation – at this level, the system carries out all actions. The human is completely out of the 'loop' and cannot intervene.

Table 5. Different levels of human-machine control.
Adopted from Endsley & Kaber (1999)

**Appendix 1** Source code engineTaskAgent

Here you can see the source code of the agent class of the *engine task agent,* which is one of the more simple agents. In the setup function you can see that it starts up a form on which the visualization of the task is handled. As said before the visualization and the automation of the task agent were merged into one *engine task agent.* The automation is taking care of on the same thread on which the form runs, this because it makes it less difficult to access the variables that are used in the engine simulation; it prevents cross thread issues. After the agent has started up a new thread with the visualization form, it adds a behavior which checks for new messages. This behavior is defined at the end, it states that it is a cyclic behavior (and this will run over and over again). In the action section of the behavior you can see that it checks its inbox for new messages. The ontology parameter of a message lets you define which ontology should be used when parsing this message. After we had redesigned our ontology many times over and over, we decided to misuse the ontology parameter (which is string variable) and write the name of the entity to which the message was related in the ontology parameter (until the ontology was more matured). So for example the message with the ontology parameter set to automationLevel would have as it content the updated automation level of the engine task. The block statement at the end of the behavior does not explicitly stop the execution of the behavior, it defines that the next execution of the behavior is scheduled to occur after 15 ms.

```
using System.Threading;
using jade.core;
using jade.core.behaviours;
using jade.lang.acl;

namespace engineTaskAgent
{
    public class engineTaskAgent:Agent
    {
        /// The operator has to monitor slide bars indicating
        /// the temperature and pressure of two engines.
        /// (Temp < 75 && Press > 25)
        Thread d_formThread;
        public engineTaskAgentForm d_form;

        public override void setup()
        {
            d_formThread = new Thread(new ThreadStart(startForm));
            d_formThread.Start();
            addBehaviour(new recieveMessage(this));
        }
```

```
        public override void takeDown()
        {
            try
            {
                d_formThread.Abort();
                d_form.Dispose();
            }
            catch { }
        }
        public void startForm()
        {
            d_form = new engineTaskAgentForm(this);
            System.Windows.Forms.Application.Run(d_form);
        }
    }
    public class recieveMessage : CyclicBehaviour
    {
        public engineTaskAgent d_agent;
        public recieveMessage(engineTaskAgent owner)
        {
            d_agent = owner;
        }
        public override void action()
        {
            ACLMessage msg = d_agent.receive();
            if (msg != null && msg.getContent().Length > 0)
            {
                if (msg.getOntology() == "automationLevel")
                {
                     d_form.automationLevel = int.Parse(msg.getContent());
                }
                if (msg.getOntology() == "keyPressAtScreen1")
                {
                    d_agent.d_form.keyPressed(msg.getContent());
                }
                if (msg.getOntology() == "workload")
                {
                    d_agent.d_form.workload = int.Parse(msg.getContent());
                }
            }
            block(15);
        }
    }
}
```

**Appendix 2** Source code readEyeTrackerData

This behaviour inherits from CyclicBehaviour, this behaviour repeats over and over again, at the end of the behaviour it waits d_agent.cycleTime milliseconds before it repeats the behaviour. This behaviour lets the agent connect to the Smart Eye client, if it has a connection it will request data from the client, and use this information to calculate the current fixation of the operator. It then

```
public class readEyeTrackerData : CyclicBehaviour
{
    private operatorFocusAgent d_agent;
    private Byte d_state;

    public readEyeTrackerData(operatorFocusAgent agentPerformingBehaviour)
    {
        d_agent = agentPerformingBehaviour;
        d_state = 0;
    }

    public override void action()
    {
        switch (d_state)
        {
            case 0:
                d_agent.connectToSmartEye("134.221.42.169", "3001", false);
                d_state++;
                break;
            case 1:
                if (d_agent.isConnectedToSmartEye())
                    d_state++;
                break;
            case 2:
                if (d_agent.getDataFromSmartEyeClient())
                {
                    d_agent.calcFixation(); // see next page
                    d_agent.requestRelatedTrack();
                    d_agent.distributeInformation();
                }
                break;
        }
        block(d_agent.d_cycleTime);
    }
}


protected void calcFixation()
{
    if (d_frameList.Count > 0)
    {
        System.Drawing.Point point = new System.Drawing.Point();
//Use the first frame to determine to which screen the operator is looking
//this is not ideal but there are 60 samples/second, so this effect is small

        d_screennumber = d_frameList[0].screenNumber;
```

```
        foreach (smartEyeData frame in d_frameList)
        {
            if (frame.screenNumber == d_screennumber)
            {
                point.X += frame.eyeLocation.X;
                point.Y += frame.eyeLocation.Y;
            }
        }
        point.X /= d_frameList.Count;
        point.Y /= d_frameList.Count;
        // transpose the Smart Eye coordinates to TFT monitor coordinates
        transposeCoordinates(ref point);
        d_fixation = point;
    }
}
```

**Appendix 3** Source code algorithm to determine operator "freeTime"

```
//In milliseconds
long freeTime = 0;
for (int i = 0; i < d_smartEyeData.Count-1; i++)
{
  //Operator is not focussing at one of the screens
  if (d_smartEyeData[i].screen == 0)
    freeTime += d_smartEyeData[i].time - d_smartEyeData[i + 1].time;

  //Operator looks for 2s. at the engine task while there is no engine error
  if (d_smartEyeData[i].screen == 1 && !engineError)
  {     long l = (d_smartEyeData[i].time - d_smartEyeData[i + 1].time);
        if (l > 2000) freeTime += l - 2000;
  }

  //Operator looks for 2s. at the middle (radar) screen, while there
  //are no tracks that need to be identified
  if (d_smartEyeData[i].screen == 2 && d_numOfUnidentifiedTracks == 0)
  {     long l = (d_smartEyeData[i].time - d_smartEyeData[i + 1].time);
        if (l > 2000) freeTime += l - 2000;
  }

  //Operator looks for 2s. at the tracking task && no tracking error
  if (d_smartEyeData[i].screen == 3 && !trackingError)
  {     long l = (d_smartEyeData[i].time - d_smartEyeData[i + 1].time);
        if (l > 2000) freeTime += l - 2000;
  }

  //Operator is looking at the lower (track information) screen, while there
  //are no tracks that need to be identified
  if(d_smartEyeData[i].screen == 4 && d_numOfUnidentifiedTracks == 0)
    freeTime += d_smartEyeData[i].time - d_smartEyeData[i + 1].time;
}
```