MASTER

SOWAT : ship operator workload assessment tool
an explorative implementation of a workload assessment and visualization tool for ship operators

van Weert, J.C.M.

Award date:
2006

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
SOWAT
Ship Operator Workload Assessment Tool

An explorative implementation
of a workload assessment
and visualization tool
for ship operators

by
J.C.M. van Weert

Primary Supervisor TU Delft/TNO:
Ir. M. Grootjen (marc@grootjen.nl)

Secondary Supervisor TU Delft/TNO:
Prof. Dr. M.A. Neerincx (neerincx@tm.tno.nl)

Supervisor TU Eindhoven:
Dr. A.I. Cristea (a.i.cristea@tue.nl)

Additional Board member TU Eindhoven:
Prof. Dr. Ir. J.J. van Wijk (vanwijk@win.tue.nl)

Eindhoven, August 2006
Preface

This research was conducted at the Technical University of Delft, for TNO and the Royal Dutch Navy and for my graduation at the Technical University of Eindhoven.

In the first place, I would like to thank Eindhoven University of Technology, Delft University of Technology, TNO and the Royal Navy for giving me the opportunity to graduate in such a broad and interesting domain. Several people deserve personal thanks: Marc Grootjen for his supervision, pleasant cooperation and feedback, Alexandra Cristea for her supervision and useful feedback, Marx Neerincx for his input and Jack van Wijk for his feedback on the visualizations. Last but not least, I would like to thank Marjolein, my family and my friends for their support and encouragements.

Jochum van Weert
Eindhoven
August 2006
Table of Contents

Preface ..........................................................................................................................3
Table of Contents .............................................................................................................4
Summary .........................................................................................................................6
Abbreviations ..................................................................................................................7
1. Introduction ...............................................................................................................8
  1.1 Background ..........................................................................................................8
  1.2 Research question ...............................................................................................8
  1.3 Scope ....................................................................................................................8
  1.4 Method .................................................................................................................9
  1.5 Structure of the report .......................................................................................9
2. Adaptive support .........................................................................................................10
  2.1 High level framework for adaptive support .......................................................10
  2.2 Taskload .............................................................................................................11
  2.3 Measuring effort and performance ......................................................................16
    2.3.1 Physiological measures .............................................................................17
    2.3.2 Subjective effort measures .........................................................................22
    2.3.3 Performance measures ...............................................................................24
    2.3.4 Overview ....................................................................................................26
3. Design and implementation of the software ............................................................27
  3.1 High level design ...............................................................................................27
  3.2 Input sources .......................................................................................................28
    3.2.1 The Observer ...............................................................................................29
    3.2.2 The ADIOS software ..................................................................................30
    3.2.3 Workload watch .........................................................................................32
    3.2.4 Task model ..................................................................................................32
    3.2.5 User model ..................................................................................................34
    3.2.6 Physiological measures .............................................................................35
    3.2.7 Expert ratings .............................................................................................36
  3.3 Features ..............................................................................................................37
  3.4 SOWAT tool usage .............................................................................................43
  3.5 Implementation details .......................................................................................44
    3.5.1 Operator state log ......................................................................................44
    3.5.2 Calculating Taskload data ..........................................................................45
    3.5.3 Visualizations .............................................................................................46
4. Experiments and analysis ............................................................................................47
  4.1 ADIOS 1 .............................................................................................................47
    4.1.1 Experiment design ......................................................................................47
    4.1.2 Measured data ............................................................................................48
  4.2 OLA 2 ................................................................................................................49
    4.2.1 Experiment design ......................................................................................49
    4.2.2 Measured data ............................................................................................49
  4.3 ADIOS 2 .............................................................................................................50
    4.3.1 Experiment design ......................................................................................50
    4.3.2 Measured data ............................................................................................50
  4.4 Example analysis ..................................................................................................50
5. Conclusions and future work .....................................................................................57
  5.1 Conclusion and discussion ..................................................................................57
5.2 Future research and improvements
5.2.1 Adaptive Hypermedia and the DTA framework
5.2.2 Real-time SOWAT
5.2.3 Determining CTL interval
5.2.4 Visualizations
5.2.5 Interpretation of physiology
References
Appendix A: SOWAT User Manual
Summary

With the introduction of more and more technology into navy ships, the role of the operators of these ships has shifted from lower level monitoring and control, to higher level supervision. Information processing demands for the operators have increased substantially. To deal with this new situation, an operator centered approach needs to be taken and operators should be provided with real-time support. The allocation of tasks between the operators and the automated system should be based on the actual cognitive task load of the operators. The central question of this project is how this cognitive task load can be detected automatically and how this information can be used to support the operator.

During this project a software tool was developed to answer this question. First we performed research into the characteristics of task load and the different measurements that can be used to determine task load. Once we established this theoretical background, we performed experiments in order to collect data that served as an indicator of operator taskload. Based on this collected data, we designed and implemented the SOWAT software tool. This tool is able to process data from several of these data sources into an expression of operator taskload over time. With this tool, analysis can be done on the experimental data. This analysis gives insight into the taskload flow of the operator during these experiments and it can determine critical regions in this taskload.

With the implementation of a software tool that can calculate operator task load from actual measurement data, it has implicitly been shown that taskload can be detected automatically. Operator support strategies can be based on thresholds of this taskload. The SOWAT tool provides functionality that can determine these thresholds. Future research should expand the possibilities to determine these thresholds and real-time task load assessment should be implemented, so different task allocation strategies based on task load can be constructed and tested.
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCF</td>
<td>Air Defense Command Frigate</td>
</tr>
<tr>
<td>CTL</td>
<td>Cognitive Task Load</td>
</tr>
<tr>
<td>DTA</td>
<td>Dynamic Task Allocation</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>ECG</td>
<td>Electro Cardiogram</td>
</tr>
<tr>
<td>EEG</td>
<td>Electro Encephalogram</td>
</tr>
<tr>
<td>EOG</td>
<td>Electro Oculogram</td>
</tr>
<tr>
<td>LIP</td>
<td>Level of Information Processing</td>
</tr>
<tr>
<td>RSME</td>
<td>Rating Scale Mental Effort</td>
</tr>
<tr>
<td>TA</td>
<td>Task Allocation</td>
</tr>
<tr>
<td>TO</td>
<td>Time Occupied</td>
</tr>
<tr>
<td>TSS</td>
<td>Task Set Switches</td>
</tr>
<tr>
<td>SCC</td>
<td>Ship Control Center</td>
</tr>
<tr>
<td>SOWAT</td>
<td>Ship Operator Workload Assessment Tool</td>
</tr>
<tr>
<td>WLW</td>
<td>Workload Watch</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Background

The operation of complex systems, like navy ships and power plants, can put a high load on the operators of these systems. With the introduction of more and more technology, fewer personnel have to perform more high level tasks and process more information. Optimal functioning of human-machine systems asks for an operator-centered design approach and thus for an operator assessment method. Neerinxc (2003) designed such a method. This method assesses the operator using taskload. For example, over- or under loaded operators are more likely to make errors or miss important information, and thus compromise optimal operation of a system. However, this method was used for design purposes only. Altering the system (e.g. adapting to the operator) after implementation is not possible.

In order to keep these increasingly complex systems running smoothly at all times, it is important to design specifically personalized support which can differ in time (Neerinxc et al., 2005). Changes in operator taskload should be detected early on and supported in the right way at the right time. Support can be given by reallocation of tasks between persons (Grootjen et al., 2006), or by changing the level of automation (Kruit, 2004).

1.2 Research question

This research covers the design of a taskload assessment tool for ship operators of the Royal Dutch Navy. The research question is formulated as follows:

*Can operator taskload be detected, quantified and visualized and can this information be used to optimize task allocation?*

1.3 Scope

In order to support the operator at the right time with the right support, a system that can assess taskload real-time is needed. This information can be used to allocate tasks among ship operators or change the level of automation, in order to optimize operator performance and effort. Therefore, the scope of the current research will be the design of a taskload assessment tool, which can be used to indicate variations in taskload over time. This will be done off-line, after the required information has been gathered. Although real-time taskload assessment and actual task allocation is outside the scope of this project, suggestions will be made on how such a system can be achieved in the future.
1.4 Method

First a study into the characteristics of taskload and adaptive support was conducted. This way a better idea of how taskload can be measured and how it can be used in the design of the tool was obtained. After this study, experiments were performed in order to verify these characteristics, to establish domain knowledge and to gather actual taskload data for analysis. Based on those results, the software tool was designed and implemented. Finally, another experiment was conducted and improvements were made, based on this experiment.

1.5 Structure of the report

The structure of this report is as follows: Chapter 2 introduces a high level framework for adaptive support and it discusses the concept of taskload and different ways in which taskload can be measured. Chapter 3 covers the design of the SOWAT software tool. Chapter 4 describes the experiments, carried out for data collection and it shows some example analyses with the SOWAT tool. Finally, chapter 5 presents the conclusions and discussion.
2. Adaptive support

In this chapter we present a general high level framework for adaptive support. Once this framework has been introduced, we will go into more details of how we can theoretically fill in some of the components of the framework.

2.1 High level framework for adaptive support

Grootjen and Neerincx (2005) present a framework for dynamic task allocation on navy ships (DTA framework). The framework is based on the idea that for the successful implementation of adaptive support for ship operators, a lot of different data has to be gathered. This data is categorized into different models and an allocator is suggested that can derive an adaptation work plan from the data in these different models. The framework suggests the usage of 4 models that together cover the required information necessary for the derivation of a work plan for task allocation and support. A schematic view of the framework is depicted in figure 2.1.

![Framework for dynamic task allocation (DTA framework)](image)

**Figure 2.1: Framework for dynamic task allocation (DTA framework)**

**Modeling components:**
The *Modeling components* collect information from the environment and store it in the appropriate models.

**Operator model:**
An essential source of information for the allocator is the operator model. In order to adapt the system to the characteristics of the operator, the operator model can contain a large variety of information:
• Taskload measures (see allocator).
• Profile parameters; routine on the job (Benyon and Murray, 1988), spatial abilities, memory span (Neerincx et al., 2005), education & characteristic knowledge, preferences, generic role in environment.
• Physiological measures.
• Performance; reaction time, time pressure, correct handling.
• Subjective effort;
• Behavioral variables; eye tracker, gaze tracker.

Task model:
The task model should contain information about the possible tasks for an operator. It should be noted that this information represents task demands that affect human operator performance and effort (i.e. it is not a definition of the operator cognitive state). The task model should be seen as some kind of static representation, expressing all possible tasks that can be executed (Neerincx, 2003).

System model:
In order to support a user, a system needs to have information about itself available as well. Consequently, the system model contains technical information about the different system components (e.g. layout, software applications and dependencies). Special cases of system models are for example application models and dialogue (interaction) models (Brown et al., 1990).

Context model:
The context model (or domain model) contains relatively static, high level information of the operational unit and its environment (e.g. the readiness state of a combat unit).

Allocator:
The allocator is the mechanism that can combine the information from the different models to generate a work plan. This work plan is based on the derived operator load. To express this load, a model is needed. We present this model in the next section.

2.2 Taskload

Generally, the term workload is used to describe the human’s ability to deal with situations that require varying degrees of attention and (cognitive) resources. Human workload is an abstract concept that is hard to quantify and therefore hard to measure. Fink et al. (2005) define workload as the ratio of resources required to perform a set of tasks, to the resources available to a person at a given time.

\[
\text{Workload} = \frac{\text{resources}_{\text{required}}}{\text{resources}_{\text{available}}}
\]

While this is an intuitive definition, it doesn’t give any indication as to how this can be quantified. How does one determine the exact amount of resources required for a certain task, or how does one determine what amount of resources a person is using for a task, or even what amount of resources a person has available?
At TNO, Neerincx and Besouw (2001) have developed a model specifically aimed at representing the load of naval operators. This model relates the load of the operator to the tasks he has to perform. We call this taskload. This Cognitive Task Load Model (or CTL-model) is a 3-dimensional load space that can be used to quantify the load of a (navy) operator and to discern certain critical regions in this load. The model expresses operator taskload in 3 dimensions:

1. **Time Occupied (TO)**

At some times the ship operator is busy performing tasks, while at other times there is nothing for the operator to do and he is idle. To capture this concept in the model, the TO variable is defined as the amount of time that the operator is busy doing something (useful), divided by the total time. If we assume an array of measurements \((i)\) and we define \((st, et)\) as an arbitrary period between start time \((st)\) and end time \((et)\), then TO over the period \((st, et)\) can be formally defined as given in table 2.1. For a visual representation of this definition, refer to table 2.4

Time occupied is the classical quantifier of taskload. Such assessments are often based on the notion that people should not be occupied more than 70 to 80 percent of the total time available (Beevis, 1992). To express operator taskload, it used to suffice to determine the TO of the operator. However with the introduction of more automation, the nature of the tasks that the operator has to perform has changed and the TO factor by itself no longer suffices to express operator taskload. Therefore the CTL model contains 2 other factors that, combined with the TO factor, give a better quantification of taskload.

\[
TO(st, et) = \left( \sum_{i,time \in (st, et)} occupied(i) \cdot dist(i, i + 1) \right) + begin - end \\
\]

\[
occupied(x) = \begin{cases} 
0 & \text{if } x\text{.task} = \text{”idle”} \\
1 & \text{otherwise} 
\end{cases}
\]

\[
begin = occupied(i_{min} - 1) \cdot dist(st, i_{min})
\]

\[
end = occupied(i_{max}) \cdot dist(et, i_{max} + 1)
\]

\[
i_{min} = \min(i\text{.time} \in (st, et))
\]

\[
i_{max} = \max(i\text{.time} \in (st, et))
\]

\[
dist(x, y) = y\text{.timestamp} - x\text{.timestamp}
\]

**Table 2.1: Definition of Time Occupied**
2. Task Set Switches (TSS)

Due to automation, the responsibilities of the operator have increased. Therefore he may have to deal with several problems at the same time. This causes the necessity for the operator to shift his attention between different problems. These attention shifts affect the load of the operator and are therefore expressed in the CTL model, using the Task Set Switches (TSS) variable.

Specifically, in the operation of a navy ship, the operators are presented with certain so-called alarms. An alarm means that there is a situation the operator has to deal with, e.g. a fire. For each alarm there is a predefined set of tasks that has to be carried out in order to address the alarm. New alarms can occur before the current alarm has been handled, so several alarms can be “active” at the same time. In this case, the operator may have to switch between the handling of the different alarms, thus switching between different task sets. The TSS variable is defined as the amount of switches between different task sets over a certain period of time, e.g. a minute.

If we assume an array of measurements \((i)\) and we define \((st, et)\) as an arbitrary period between start time \((st)\) and end time \((et)\), TSS per minute over the period \((st, et)\) can be formally defined as given in table 2.2.

\[
TSS(st, et) = \frac{\sum_{i.time \in (st, et)} switch(i - 1, i)}{et - st} \cdot 1 \text{ minute}
\]

\[
switch(x, y) = \begin{cases} 
0 & \text{if } x \text{.taskset} = y \text{.taskset} \\
1 & \text{otherwise}
\end{cases}
\]

Table 2.2 Definition of Task set switches

3. Level of Information Processing (LIP)

Every task set contains subtasks or actions. These actions differ in complexity, depending on the type of action and on the experience of the operator. The LIP variable acts as an indicator of this complexity. Neerincx and Besouw (2001) define LIP as being one of the 3 levels of the Skill-Rule-Knowledge framework of Rasmussen (Rasmussen, 1986).

Rasmussen discerns skill, rule and knowledge based actions.

- **Skill based actions** are actions that are carried out automatically, almost without thinking, e.g. to instantly hit the brake while driving when something crosses the road right in front of the car.

- **Rule based actions** require a bit more cognitive processing, but they concern problems that have routine solutions. These actions are based on simple if <condition> then <action> decision schemes, e.g. seeing a car coming from your right at an intersection and pressing the brake because cars coming from your right have right of way.
Knowledge based actions require more in depth problem analysis and solutions that require high cognitive processing, e.g. deriving the best route to your destination while taking different factors into consideration, such as possible traffic jams, shortest driving distance, etc.

An alternative to the strict 3-level LIP based on the model of Rasmussen, LIP can be expressed on a 5-point scale, also called a Likert scale (Likert, 1932).

LIP is defined as the average LIP of the actions executed during a period of time. If we assume an array of measurements \( i \) and we define \((st, et)\) as an arbitrary period between start time \((st)\) and end time \((et)\), LIP can be formally defined as given in table 2.3. A visual representation of this definition is shown in table 2.4.

\[
LIP(st, et) = \frac{\sum_{i.time \in (st, et)} i.LIP \cdot dist(i, i + 1)}{et - st} + begin - end
\]

\[
begin = (i_{min} - 1) \cdot LIP \cdot dist(st, i_{min})
\]

\[
end = i_{max} \cdot LIP \cdot dist(et, i_{max} + 1)
\]

\[
i_{min} = \min(i.time \in (st, et))
\]

\[
i_{max} = \max(i.time \in (st, et))
\]

\[
dist(x, y) = y.timestamp - x.timestamp
\]

**Table 2.3 Definition of Level of Information Processing**

The small vertical bars across this line represent measurement points \((i)\). To calculate TO or LIP over a period given by \((st, et)\), first \(main\) is calculated, then \(begin\) is added and \(end\) is subtracted from \(main\).

**Table 2.4: Visual representation of TO and LIP definitions**
Figure 2.2 shows an abstract representation of the CTL cube. If CTL points are calculated for a number of intervals, the taskload of the monitored operator over the course of a scenario can be plotted as a path through the cube. If this path is combined with measurements of effort and performance, this can give a meaning to certain regions in the cube. In figure 2.2, the green region represents the optimal state in which the operator performs at his best. The red regions are best avoided since they represent states in which the operator doesn’t have an optimal workload. Overload is the most obviously undesirable state: the operator is unable to cope with the situation and his performance will suffer. Underload is undesirable because after longer periods of inactivity, the operator can easily miss new information that can suddenly occur. Vigilance occurs when an operator is doing a very simple task for a long time, for example monitoring a meter for it to fall below a certain threshold. In the same vein as with underload, the operator may miss new information because of decreased attention (Neerincx et al., 2004).

Note that there are no values along the axes of the cube. It is not exactly defined where these interesting regions are located, mostly because the location of these regions varies among different persons. One of the challenges taken up in this thesis is to determine the location of these regions for a specific operator or a specific group of operators (stereotypes). Only then it can be detected when an operator is located in one of these regions and the appropriate action can be taken, in order to try to get the operator back into the optimal region again.

In the next section we further describe effort and performance. This can be interpreted as research into what needs to go into the operator model of the framework presented in section 2.1.
2.3 Measuring effort and performance

A way to determine undesirable regions in an operator’s CTL-space, are measurements of effort and performance. A relationship between effort, performance and workload is described by Veltman and Jansen (2005). This relationship is depicted in figure 2.3.

![Figure 2.3: The relationship between mental workload and performance](image)

An operator’s performance is determined by the task demands and the amount of effort the operator is investing. If the task demands are very low, people tend to get bored and lower their efforts so that performance suffers. If workload increases above normal, performance can still be kept satisfactory by increasing the amount of effort put into the task. This can only be done for a short period of time before the operator gets fatigued and performance goes down. If the workload becomes too high, overload will occur and the person will not be able to deal with the situation. In this case performance drastically decreases until it hits a minimum threshold.

With measurements of effort and performance, combined with the CTL measurements, it should be possible to determine the location of the mentioned regions in the cube for a certain person. When these regions are established, this information can be used to determine the necessity of adaptation and support in specific situations.

A lot of different measurements can be used to give an indication of effort or performance. A distinction can be made between three kinds of measurements: physiological effort, subjective effort and performance measurements. To determine what measurements are suitable for us to use, we consider three criteria to rate the different measures.

1. **Sensitivity.** How sensitive to workload changes is the measure?
2. **Obtrusiveness.** Does the measurement hinder the subject in his actions or his freedom of movement?
3. **Accessibility.** The TU Delft and TNO have tools and equipment available. We have little means of purchasing new ones, so we will have to work with what’s available.

In the following we will give a short introduction to the different measures and discuss their appropriateness according to the three criteria above. This section can also be used as a quick reference for future research.
2.3.1 Physiological measures

Physiological measures are measures of the physical state of a person. It has been shown that workload influences certain physical parameters. However, there is no physiological measure that can be directly interpreted as a workload indicator (Veltman and Gaillard, 1996; Wilson, 2001; Wilson and Russel, 2003; Miyake, 2000). Variations in physiological measures can be attributed to a lot of different factors, with workload being just one of them. Moreover, different measures provide insight in different aspects of workload, e.g. eye blink rate variations can correspond specifically with visual demands (Wilson, 2001; Wilson and Russel, 2003). We are interested in physiological indicators of workload, because they can possibly be interpreted as a measurement of effort.

**Heart Measures**

Heart measurements are popular physiological measurements for workload, probably because they are relatively easy to perform. An electrocardiogram is recorded using electrodes applied to the body. Heart measures yield some promising results (Veltman and Gaillard, 1996; Veltman and Gaillard, 1998; Wilson, 2001; Wilson and Russel, 2003; Miyake, 2000).

There are two common heart measures: heart rate (how fast the heart beats) and heart rate variability (beat-to-beat alterations in heart rate). Experiments show that both are sensitive to workload variations, though heart rate variability is supposedly only sensitive to big changes in workload (Veltman and Gaillard, 1998; Wilson, 2001).

- **Sensitivity:**
  High, widely used because of it.

- **Obtrusiveness:**
  Moderately obtrusive, electrodes are applied to the body.

- **Accessible:**
  Yes

**Respiratory measures**

Respiratory measures are another good candidate for workload estimation. They have about the same obtrusiveness as heart measurements. Generally, the inductive plethysmography method (Veltman and Gaillard, 1996; Veltman and Gaillard, 1998) is used. It makes use of two bands, one around the chest and one around the abdomen. The stretching of these bands can be used to derive several different respiratory parameters, as follows (Veltman and Gaillard, 1996; Veltman and Gaillard, 1998; Wientjes, 1992):
- Respiration Rate: Frequency of respiration
- Tidal Volume: Amplitude of respiration
- Inspiratory Flow: Tidal Volume / Inspiratory time
- Duty Cycle Time: inspiratory time / total cycle time
- Minute volume: Respiration rate * Tidal Volume.

Of these measures Duty Cycle Time and Minute Volume were found to be most sensitive.

- **Sensitivity:**
  High, widely used because of it.
Obtrusiveness:
Moderately obtrusive, elastic bands are applied to the body.
Accessible:
Yes

Eye Blinks
Eye blinking information can be collected using an electro-oculogram (EOG).
A few different parameters can be derived from an EOG, for example:
- Blink frequency
- Closing time
- Blink duration
- Blink amplitude
Eye blink features have been shown to be affected by workload changes (Veltman and Gaillard, 1998; Wilson, 2001; Wilson and Russel, 2003), however they are specifically sensitive to changes in terms of visual workload. This should be kept in mind when using these measures. Recording an EOG is quite obtrusive, since electrodes are applied to the face. This may hinder normal operation.

Sensitivity:
High, although especially influenced by visual workload.
Obtrusiveness:
Highly obtrusive, electrodes are applied to the face.
Accessible:
Yes

Gaze tracking
Using a special gaze tracking camera, a person’s point of gaze can be determined. It can be used to see what a person spends the most time looking at and how his point of attention changes over time. Some examples of it’s usage are: determining what parts of an interface attract the most attention (Marshall et al., 2003); or even using it as input device (Zhai et al., 1999), i.e. looking at a button in an interface long enough to press it. In terms of workload, the amount of changes in gaze points can be an indicator. If a person changes his attention point often, it is likely that the person has a higher workload than when he is staring at the same point. Also staring at the same point for longer periods of time can be an indication of vigilance; the person is waiting for something to happen. However this has not been researched and these are only assumptions.
Gaze tracking can be measured unobtrusively, because all that is needed is a special camera pointed at the operator’s face, so there is no direct contact with the subject. However, gaze tracking cameras tend to be large and that makes them impractical for using in environments with little space.

Sensitivity:
Unknown, no literature was found using gaze tracking as workload indicator.
Obtrusiveness:
Hardly obtrusive, the gaze tracking camera is not in direct contact with the person, but it has to be located somewhere in front of the person.
Accessible:
Yes
Pupil Dilation
The same device that is used for gaze tracking can also be used for measurements of the pupil. In the presence of effortful cognitive processing, the pupil responds rapidly with a reflex reaction (Marshall, 2002). These reactions can be counted to give an indication of workload. Pupil reflexes are also triggered by differences in light, so this should be taken into account.

Sensitivity:
High, provided the lighting conditions are fairly constant.

Obtrusiveness:
Hardly obtrusive, the gaze tracking camera is not in direct contact with the person, however it has to be located somewhere in front of the person.

Accessible:
Yes

Voice stress
According to Rothkrantz et al. (2004) acoustic properties of the voice can be used to derive workload. Quite a few acoustic parameters can be derived, for example jitter, fundamental frequency, intensity or speech rate. Research has shown that especially the parameters jitter and fundamental frequency are sensitive to workload changes. Speech can be unobtrusively recorded; however, an operator will not be engaged in speech all the time, so that makes it a less reliable indicator of workload.

Sensitivity:
High, but voice is not used constantly.

Obtrusiveness:
Hardly obtrusive, usually a headset is used to capture voice recording.

Accessible:
No/in the future

Facial expressions
Pattern recognition techniques can be used to detect a person’s facial expressions (Kuilenburg et al., 2005). While those expressions are usually analyzed to derive emotion, they may also give an indication of workload. However it will be hard to translate the expressions into a quantifiable measurement of workload.
We have agreed with Maya Pantic of the TU Delft to share the video recordings of the experiment(s) with her so she can do a facial expression analysis. However this will not be done within the scope of this project.

Sensitivity:
Unknown, subject of future research.

Obtrusiveness:
Unobtrusive, video footage of the face can be recorded using a tiny camera in front of the subject.

Accessible:
No/in the future
Face warmth
TNO recently announced the discovery that workload has an effect on a person’s facial warmth (Veltman and Vos, 2005). When recorded with a thermal camera, the research showed that a person’s nose tends to become significantly colder when engaged in high workload situations.

However, in a conversation with Hans Koopman from TNO he declared that the facial warmth method is not at all sensitive to small changes in workload. It can only tell the difference between very low and very high workload. Therefore it’s most likely not suitable for us to use.

*Sensitivity:*
Low, only big differences can be detected.

*Obtrusiveness:*
Hardly obtrusive, the thermal camera needs to be in front of the face.

*Accessible:*
Yes

Brain activity measures (EEG)
Brain activity is a popular method of workload assessment. It seems very intuitive to measure workload with brain activity. After all, cognitive processing is done in the brain, so differences in workload are likely to be reflected in brain activity. Brain activity can be measured using an EEG. This EEG is compiled using electrodes attached to different places on the head, measuring changes in current. Several studies have concluded that EEG measures are indeed sensitive to workload and relatively accurate as well (Wilson, 2001; Wilson and Russel, 2003; Prinzel et al., 1999).

However, obtrusiveness is quite an issue here. In order to do a decent EEG measurement, a lot of electrodes need to be used and it is not at all comfortable applying the electrodes to a persons scalp. This is a serious downside of EEG. An alternative to electrodes is a cap, but it’s less sensitive and still considerably obtrusive.

*Sensitivity:*
High, promising results in literature.

*Obtrusiveness:*
Very obtrusive, a considerable number of electrodes need to be applied directly to the scalp.

*Accessible:*
Yes

Blood pressure
Blood pressure is a physiological variable that has been measured in several workload related experiments (Veltman and Gaillard, 1996; Veltman and Gaillard, 1998; Miyake, 2000). However in (Veltman and Gaillard, 1996) and (Veltman and Gaillard, 1998) it is mentioned that blood pressure is not systematically affected by workload, so this method may not be useful.

*Sensitivity:*
Low, unsuitable for usage.

*Obtrusiveness:*
Moderate, measurement is done using a small cuff around the finger.

*Accessible:*
No
**Skin conductivity/perspiration**

Haag et al. (2004) suggest that measurements of skin conductivity can be used as an indicator of a person’s emotions. This is not the same as workload, but it is related. More research needs to be done into the usefulness of skin conductivity as a workload indicator. Changes in skin conductivity are caused by perspiration.

The obtrusiveness of skin conductivity measurements depends on what part of the skin is measured. The most common measurement is on the hands or fingers. This requires the subject to wear a special glove. This will hinder usage of the hand for typing, for example.

*Sensitivity:*
Unknown, no literature was found using skin conductivity as workload indicator.

*Obtrusiveness:*
Highly obtrusive, the glove will hinder usage of the hand.

*Accessible:*
No

**Gesture recognition**

Ehlert (2003) mentions research into gesture recognition. It is possible to recognize a person’s gestures from video images. Gestures could give an indication of a user’s mental state, for example nodding, or gestures of frustration. This is most likely too vague for us to use. Gestures give an idea of a user’s mental state, but can that be translated into a workload indication? Moreover, some people gesture a lot while others don’t gesture at all.

*Sensitivity:*
Unknown, no literature was found using gesture recognition as workload indicator.

*Obtrusiveness:*
Hardly obtrusive, gestures are detected with a camera.

*Accessible:*
No

**Muscle measurements (EMG)**

Lundberg et al. (1994) discuss the correlation between stress and muscular tension of the trapezius muscle (a large muscle between the back of the neck and the shoulders). The report mentions a significant increase in muscular tension in high stress situations.

*Sensitivity:*
Probably good, no literature was found using it.

*Obtrusiveness:*
Hardly obtrusive, gestures are detected with a camera.

*Accessible:*
No

**Cortisol**

Cortisol is a hormone that is released under stress. It could therefore give an indication of workload. In (Veltman and Gaillard, 1996), cortisol levels in saliva are measured before and after a workload scenario, but differences are hardly noticeable and absolute levels are very person dependent. Another way is measuring cortisol levels in urine (Lundberg et al., 1994).
It seems that cortisol is not a clear indicator of workload. Furthermore, cortisol can only be measured explicitly by taking urine or saliva samples and analyzing them separately, thus making them inconvenient and unsuitable for continuous, real-time measurements.

*Sensitivity*
Low, can only be measured afterwards, so it gives no continuous, real-time measurements.

*Obtrusiveness*
High, saliva or urine samples need to be analyzed.

*Accessible*
No

### 2.3.2 Subjective effort measures

Subjective measures ask for the opinion of a person about his own effort. However, these measures are subject to personal bias and different persons tend to have different criteria when filling in rating scales. Nevertheless, subjective measures are an important addition to taskload assessment and we want to incorporate them in our set of measures.

**NASA-TLX**

NASA-TLX (Hart and Staveland, 1988) is a popular subjective load scale. It uses several scales to express different aspects of workload and effort. This method has been tested and it is quite sensitive. However the amount of different scales that the subject has to fill in makes the scale not suitable for usage during the scenario, since it will take too much time.

*Sensitivity*
Good.

*Obtrusiveness*
High, in the sense that it takes long to fill in all scales.

*Accessible*
Yes

**Self Assessment Manikin (SAM)**

The Self-Assessment Manikin (SAM) method (Bradley and Lang, 1994) was developed to present a user with an intuitive way to rate his own emotion. It consists of 3 scales: pleasure, arousal and dominance. These scales together are said to give a good representation of user emotion and can be quickly filled in. This makes them feasible for real-time measuring. However, user emotion does not really correspond to effort, and that makes this method not really suitable.

The SAM scale is depicted in figure 2.4.
Sensitivity:
Low, it actually measures emotion.

Obtrusiveness:
Moderately obtrusive, scales can be filled in quite quickly, but it distracts from the actual task.

Accessible:
Yes

Workload Watch
At TNO, the Workload Watch program (Boer, 1997) was designed, running on a PDA. Every minute this program gives an auditory signal. This prompts the subject to fill in the complexity of the last minute and his effort over the last minute, both on a five-point Likert scale (Likert, 1932). These scales can be filled in quickly and it gives a straightforward measurement of effort.

Sensitivity:
High.

Obtrusiveness:
Moderately obtrusive, scales can be filled in quite quickly, but it distracts from the actual task.

Accessible:
Yes

Rating Scale Mental Effort (RSME)
RSME is a continuous scale for rating a person’s subjective effort (Zijlstra and Doorn, 1985). The RSME scale does not have a specific endpoint, making it very flexible, but somewhat harder to interpret automatically. An RSME scale is generally filled in at the end of an experimental session, not during the experiment itself. The RSME scale is depicted in figure 2.5.
2.3.3 Performance measures

Performance measures are used to determine how well the subject is executing his tasks.

Continuous memory task (CMT)
A continuous memory task (CMT) is a task that requires the subject to continually keep track of something and respond to a certain stimulus. An example of a CMT can be found in (Veltman and Gaillard, 1996). Here, pilots are flying a scenario in a flight simulator. During the scenario they are presented with letters from the alphabet on a
screen. They are required to press a certain button when one of predetermined target letters is presented. Furthermore, they need to keep track of how often the different target letters occur. When a target letter occurs for the first time, the button needs to be pressed twice. The operator’s reaction time and amount of errors are recorded. This gives an insight into the pilot’s workload.

While this is a good measure of performance, the obtrusiveness is unacceptably high. The CMT requires so much attention that it hinders the performance of the actual task severely.

Sensitivity:
High.
Obtrusiveness:
Highly obtrusive, hinders execution of actual task.
Accessible:
Yes

Expert ratings
Expert ratings ask the opinion of an expert about the performance of the subject. If the expert’s criteria are properly defined, so consistency is assured between the ratings on different subjects, expert ratings are an excellent way to determine an operator’s performance.

Sensitivity:
High, assuming good rating criteria.
Obtrusiveness:
Low, does not require subject’s attention.
Accessible:
Yes

Efficiency
Efficiency measures express how fast the subject performs his tasks. A person that performs tasks fast can do more work in a set amount of time. So efficiency is a useful performance measure. However a fast pace of work can go hand in hand with a high error rate, so efficiency measures should always be done together with effectiveness measures. Efficiency can be measured by observation.

Sensitivity:
High, if combined with effectiveness measures.
Obtrusiveness:
Low, does not require subject’s attention.
Accessible:
Yes

Effectiveness
Effectiveness measures express how well the subject performs his tasks. For good operation of any system it is very important that tasks are performed correctly. However taking to much time in order to perform tasks correctly is also undesirable. Therefore effectiveness measures should always be combined with efficiency measures.

Sensitivity:
High, if combined with efficiency measures.
Obtrusiveness:
Low, does not require subject’s attention.

Accessible:
Yes

### 2.3.4 Overview

To summarize this section, the explored measurements with their corresponding criteria are listed in table 2.4; scores range from -2 (bad/low) to 2 (good/high).

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Sensitivity</th>
<th>Obtrusiveness</th>
<th>Available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart measures</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Respiratory</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Blinking, EOG</td>
<td>1</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Gaze tracking</td>
<td>0</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Pupil dilation</td>
<td>2</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Voice stress</td>
<td>0</td>
<td>2</td>
<td>no/afterwards only</td>
</tr>
<tr>
<td>Facial expression</td>
<td>-1</td>
<td>2</td>
<td>no/afterwards only</td>
</tr>
<tr>
<td>Face warmth</td>
<td>0</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Brain activity (EEG)</td>
<td>2</td>
<td>-2</td>
<td>yes</td>
</tr>
<tr>
<td>Blood pressure</td>
<td>0</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>Skin conductivity</td>
<td>0</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td>Gesture recognition</td>
<td>-2</td>
<td>2</td>
<td>no/afterwards only</td>
</tr>
<tr>
<td>Muscles (EMG)</td>
<td>1</td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>Cortisol (saliva)</td>
<td>-1</td>
<td>-2</td>
<td>no</td>
</tr>
<tr>
<td>NASA-TLX</td>
<td>2</td>
<td>0</td>
<td>yes/afterwards only</td>
</tr>
<tr>
<td>Self Assessment Manikin (SAM)</td>
<td>-1</td>
<td>0</td>
<td>yes/afterwards only</td>
</tr>
<tr>
<td>Workload Watch</td>
<td>2</td>
<td>0</td>
<td>yes</td>
</tr>
<tr>
<td>Rating Scale Mental Effort (RSME)</td>
<td>2</td>
<td>0</td>
<td>Yes/afterwards only</td>
</tr>
<tr>
<td>Continuous memory task (CMT)</td>
<td>2</td>
<td>-2</td>
<td>yes</td>
</tr>
<tr>
<td>Expert ratings</td>
<td>2</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Efficiency</td>
<td>2</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Effectiveness</td>
<td>2</td>
<td>2</td>
<td>yes</td>
</tr>
</tbody>
</table>

*Table 2.4: Overview of measures*

This overview was taken into account when we selected measurements for our experiments (as described in chapter 4 of this report).
3. Design and implementation of the software

To investigate how the adaptive support framework can be filled in, we designed the SOWAT tool. This tool implements a part of the functionality of the allocator, combined with the operator model and the task model. This is meant in the sense that the tool can derive operator taskload, based on operator data and a task model. This chapter covers the design of the SOWAT tool. In section 3.1, the high level design of the tool is discussed. Section 3.2 describes the different input sources that can be used for the tool. In section 3.3, the features and usage of the tool are explained. Section 3.4 gives a stepwise procedure for analysis with the SOWAT tool. Finally, section 3.5 highlights some details of the implementation.

3.1 High level design

The purpose of the SOWAT tool is to combine measurements from different data sources, collected during experiments, to calculate the taskload flow of the monitored subjects. In figure 3.1, a schematic overview of the design of the SOWAT tool is given. Along the top, the different input sources are depicted (Observer, Adios, workload watch, expert ratings, mobi and tobii). The characteristics of the different input sources are discussed in more detail in the next section. These input sources are each processed by a parsing component within the SOWAT tool. The parsing components derive the relevant data from these input sources and this data is combined in the operator state log. The state log can be exported to an XML file, enabling the user of the tool to examine its contents, as shown on the right side of figure 3.1.

Then, using the task model, the taskload data can be derived from the operator state log. This data is derived for a user selected interval and in accordance with user selected parameters. This taskload data can be exported to a file. This is shown in the bottom left of the diagram. This taskload data is then visualized. The user can manipulate the visualizations using several parameters. Once the user is satisfied with the visualizations, they can be saved to image files.
3.2 Input sources

The SOWAT tool uses several different input sources to gather data. Not all of these sources are explicitly necessary for the SOWAT tool to be of use. The minimum requirements are to load at least a task model and either an Observer log file or an ADIOS log file. All other data sources are optional; however, some of the SOWAT features will not be available when not all of the sources are loaded.

This section covers the characteristics of the different input sources for the SOWAT tool.
3.2.1 The Observer

An important data source for the software tool is The Observer (http://www.noldus.com/site/doc200401012). This is a software package for behavioral analysis. The experimenter using the Observer can manually score the exact actions that his subject is performing. This can be done by watching the subject in real-time or using recorded video footage. All possible actions of interest have to be defined beforehand and then the execution of these actions by the subject can be annotated during the observation. A screenshot of the Observer software is shown in figure 3.2.

Because there is a direct relation between the values of the CTL variables and the performed tasks, it is very important to know accurately what task the operator is performing at all times. Ideally, this information should be provided by the ship’s system, but in this experimental setting this is not (accurately) possible. (Nevertheless we will look into this in 3.2.2). Therefore we need an alternative source for this information. The Observer is well suited for this purpose.

In the experiments conducted during this research, the monitored operators were filmed and this footage was carefully analyzed with The Observer software. The analysis from the Observer can be exported to a log file, containing all the actions of the operator(s) over the course of the experiment. These log files serve as input for the SOWAT tool.

A short sample from an Observer log file is shown below:

```
10:46:13.445177;"Adios 18";"Obs712Ods713";"schoehuizen";"score handheld";"State start";"
10:46:15.417610;"Adios 18";"Obs712Ods713";"potjer";"omroepen brandmelding 2-1";"State end";"
10:46:15.663595;"Adios 18";"Obs712Ods713";"potjer";"score handheld";"State start";"
10:46:16.093998;"Adios 18";"Obs712Ods713";"schoehuizen";"score handheld";"State end";"
10:46:16.259734;"Adios 18";"Obs712Ods713";"schoehuizen";"systeem melding";"State start";"
10:46:17.487793;"Adios 18";"Obs712Ods713";"potjer";"score handheld";"State end";"
10:46:18.183554;"Adios 18";"Obs712Ods713";"potjer";"systeem melding";"State start";"
10:46:24.547351;"Adios 18";"Obs712Ods713";"schoehuizen";"systeem melding";"State end";"
10:46:26.986758;"Adios 18";"Obs712Ods713";"potjer";"systeem melding";"State end";"
```

Columns in the log are separated by semicolons. The meaning of the columns in the log files is as follows:

- Timestamp (automatically generated);
- Observation session name (user defined);
- Observation filename (user defined);
- Subject name (user defined);
- Subject action (user defined);
- State end or state start (user defined);
- Comment (empty in this case) (user defined);

For the SOWAT analysis only the timestamp, subject name and state start/end columns are relevant.
3.2.2 The ADIOS software

The ADIOS software is a program under development by TNO, for the doctoral research of Marc Grootjen. This software package simulates the SCC (Ship Control Centre) of a Navy ship and is used in experiments about task allocation between operators of the SCC. For more details on the ADIOS project, please refer to chapter 4.1 of this report. A screenshot of the ADIOS software is shown in figure 3.3. The software generates a log file of, among other things, all interactions of the operator with the software. This log file can be used as input for our software tool to derive the actions of the operator over the course of the experiment.

While this is less accurate than the manually scored actions in the Observer log file, the log file is generated automatically and instantly, without a person having to explicitly observe the operator. Also it is more realistic, because if a taskload monitoring system is to be implemented on real ships in the future, this information needs to be generated by the ship’s systems automatically and cannot be expected to be manually authored. Therefore, the ADIOS logs can serve as an alternative to the Observer logs.

A short sample of an ADIOS log is shown below:

15:15:33: Actieve stap in procedure : Controleren effect manoeuvre
15:15:39: Procedurestap Controleren effect manoeuvre CheckState True
15:15:39: Last step checked Collision warning 3NM
15:15:40: Selecteer alarm Positie maken
15:15:40: Operator2 selected alarm Positie maken handled by Operator2: True
15:15:40: ProcedureBlad zichtbaar : False
15:15:40: Actieve stap in procedure : Positie fixen
15:15:40: ProcedureBlad zichtbaar : True
15:15:42: Procedurestap Positie fixen CheckState True
15:15:49: DelayedAlarm einde alarm|Collision warning 3NM
15:15:59: Procedurestap Positie plotten CheckState True
15:15:59: Last step checked Positie maken
15:16:01: Selecteer alarm Lage druk brandblussysteem
15:16:01: Operator2 selected alarm Lage druk brandblussysteem handled by

Each line starts with a timestamp, followed by a logged event. Not every line in the ADIOS log files is relevant for the SOWAT analysis. Most lines deal with the changes in the state of the ADIOS program itself, such as clicked buttons and received messages. Relevant lines for the SOWAT tool have one of the following forms:

- **Selecteer alarm x**: The user selected a new alarm to handle, this marks the start of a new taskset.
- **Actieve stap in procedure: x**: The user selected a new action (subtask) from the current taskset,
- **Last step checked x**: The user finished his current active task.

![The interface of the ADIOS software](image)

Figure 3.3: The interface of the ADIOS software
3.2.3 Workload watch

For the acquisition of subjective measurements, the Workload Watch (WLW) software from TNO was used. This is a program that runs on a pocket PC and it collects subjective ratings of a subject on complexity and effort. Every minute, it gives an auditory signal to get the subject’s attention. The subject is then asked to rate the complexity and his effort over the last minute, both on a five point Likert scale. When scoring their effort and performance, the software shows what was scored the last time, so the subject can score his current complexity and effort with the knowledge of what he rated a minute ago. This ensures more consistent scoring. The data collected by the Workload Watch software is written to a log. This logfile serves as input for the SOWAT tool. A line from a workload watch logfile is shown below:

9-2-2006 19:58:04 | Hoe complex vond u het werk de afgelopen minuut? | nauwelijks complex | 2343 | Hoe inspannend was het werk de afgelopen minuut? | niet inspannend | 11894 | Continue next trial

Columns are separated by ‘|’-characters. The meaning of the columns is as follows:

- Timestamp;
- Complexity question;
- Complexity question answer;
- Complexity question reaction time (in ms);
- Effort question;
- Effort answer;
- Effort question reaction time (in ms);
- Final column

We are interested in the answers to both questions. The answers are given in a verbose syntax. The scale is structured as follows:

(1) Niet complex/inspanded (Not complex / No Effort)
(2) Nauwelijks complex/inspanded (Hardly complex / Hardly any effort)
(3) Gemiddeld complex/inspanded (Moderately complex / Moderate effort)
(4) Behoorlijk complex/inspanded (Considerably complex / Considerable effort)
(5) Wel complex/inspanded (Highly complex / High effort)

3.2.4 Task model

Static information about the possible tasks that can occur during the measurements is required for the CTL calculations of the SOWAT tool. For this purpose we designed the XML task model. In the Task model, the decomposition of the available tasks into subtasks (actions) is defined. Also, for each action the corresponding complexity of that action, according to the task model author, is included in the model. The XML task model is defined using the following Document Type Definition (DTD):
Thus, we define a task list (task model) as containing zero or more tasks. Each task has a name and consists of zero or more actions. Each action has an Observer description (the name of the action within the Observer analysis), an ADIOS description (the name of the action within the ADIOS logs) and LIP values for 3 different operator skill levels. Each task is given an id-number using an attribute.

While XML is a convenient format to express and store structured data, it is not really convenient for humans to read and write. For this reason we implemented a task model authoring tool. This task model authoring tool represents the task model as a tree. The author can add and remove nodes and edit the contents of the textual nodes in order to represent the desired task model structure. The authoring tool ensures that during editing of the task model the DTD is always respected. A screenshot of the task model authoring tool is shown in figure 3.4.

The left panel of the Task Model editor shows the current task model tree. If the root node of the tree is selected, the “Add task”-button will become active, allowing the author to add a new task to the list. If a task-node is selected, the “Add action”-button will become active and enables the author to add an action to the currently selected task. The text of all ending nodes (colored green) can be edited with the “Edit text”-button. If a task or action node is selected, then pressing the “Delete node”-button will delete the selected node and its sub nodes from the tree. By enabling only the relevant buttons for the currently selected node, the DTD is never violated.

Using the “file”-menu, a new empty task model can be created, an existing task model can be loaded and the current task model can be saved to an XML file. The SOWAT software assumes that all actions that can possibly occur in the processed Observer or ADIOS logs are defined in the task model. Using the “File”-menu, an Observer logfile or an ADIOS log file can be loaded. The task model editor parses the Observer or ADIOS log and displays all actions occurring in the log in the right panel. These actions can be copy/pasted into the task model. This way it is easier for the author to ensure that all necessary actions are contained in the task model. For more detailed information about the task model editor, please refer to appendix A, which contains the user manual for the SOWAT tool.
3.2.5 User model

In the SOWAT tool, a distinction is made between operators of varying skill levels. This concept should logically be captured in the operator model (user model). However, since in the SOWAT tool the operator skill level is the only user aspect that explicitly modeled, this is embedded in the interface, as shown in figure 3.5. The skill level can be selected before plotting.

In future implementations of operator workload assessment tools, a more elaborate user model will be necessary. This is discussed further in chapter 5.2.1.
3.2.6 Physiological measures

In the experiments conducted during this project, we recorded different types of physiological measures: heart measures, respiratory measures and pupil dilation measures.

We recorded heart and respiratory measurements using the Mobi8 device (http://www.tmsi.com/pages/mobycindi.htm). To measure heartbeats, the subject had to place two electrodes on his body. One electrode was placed on the skin on the lower left ribs, just below the heart, the other electrode was placed on the skin over the right collarbone. Finally, a ground electrode was placed on the skin of the right side of the subject. For the respiratory measures, two elastic bands were used, one around the chest and one around the abdomen; this is called the inductive plethysmography method as mentioned in chapter 2.2.1. All electrodes and bands were connected to the mobi8 device. This device samples the current between the two heart electrodes and between the ends of both respiratory bands. This data was stored on an internal memory card. The collected data was exported from the memory card afterwards for further analysis. In Figure 3.6, a schematic representation of the heart and respiratory measurements using the Mobi8 is shown.

![Figure 3.6: Heart and respiratory measures with the Mobi8 device](image)

For the measurements of pupil dilation, we used the Tobii x50 Eye Tracker (www.tobii.com). This is a device primarily designed to record where a person is looking.
(gaze tracking), but the device can also be used for derivation of pupil dilation. The device in use is shown in figure 3.7.

![Figure 3.7: The Tobii eye tracker](image)

The data recorded by these devices needed to be analyzed by specialists at TNO, before it could be used as input. Unfortunately this is a time consuming process and it was not completed during the course of this project. For this reason, the SOWAT tool has only limited functionality for physiology. However, both the Mobi and Tobii data files can be loaded into SOWAT and visualized.

### 3.2.7 Expert ratings

Expert ratings are used to rate the performance of the subjects on each action. For this purpose, an expert rating authoring tool was designed. The tool is largely the same as the task model authoring tool discussed in section 3.1.4. The DTD is only slightly different from the task model DTD:

```xml
<!DOCTYPE tasklist [ 
<!ELEMENT tasklist (task*)> 
<!ELEMENT task (task_name, action*)> 
<!ELEMENT task_name (#PCDATA)> 
<!ELEMENT action (ObserverDescription, ADIOSDescription, LIP_levels)> 
<!ELEMENT ObserverDescription (#PCDATA)> 
<!ELEMENT ADIOSDescription (#PCDATA)> 
<!ELEMENT expertrating (#PCDATA)> 
<!ATTLIST task id CDATA #REQUIRED>
]>```
Again, we define a task list, containing zero or more tasks. Each task has a name and consists of zero or more actions. Each action has an Observer description (the name of the action within the Observer analysis), an ADIOS description (the name of the action within the ADIOS logs) and an expert rating.

A screenshot of the expert rating editor is shown in figure 3.8.

![Figure 3.8: Expert rating editor](image)

Just as in the task model editor, it is possible to parse an Observer log file or an Adios log file for the actions occurring in it. These will be shown in the right pane and can be copy/pasted into the task model. In addition to this, an existing task model can be imported. An expert rating tree is then automatically generated with the same task decomposition as the task model tree. The author then only has to fill in the expert rating values for each action. This is the most logical and easy way to create expert rating files, since usually a task model will be created first.

### 3.3 Features

The interface of the SOWAT tool is split into 2 sections: a data collection section and a visualization section. The data collection interface is shown in Figure 3.9.
In the upper left corner, the tab pages to navigate between the two sections of the SOWAT tool are located (region 1, figure 3.9). On the left side of the data collection interface, there are seven buttons with a corresponding text field (region 2, figure 3.9). Each of these buttons can be used to load one of the different input sources. The corresponding text field next to each button gives information about the (possibly) loaded input files. They display the filename of the loaded file or an error message if an incorrect file was loaded. Below, there is a button to unload all input files.

To the right of the input buttons, there is a section with tick boxes that indicate which of the input files have successfully been loaded (region 3, figure 3.9). Once enough input sources are loaded for calculation to be possible, the “Process input”-button in the upper right corner becomes active (region 6, figure 3.9). The following inputs can be loaded:

- Observer log file
- ADIOS log file
- Task model XML file
- Workload watch log file
- Expert rating XML file
- Mobi data file
- Tobii data file

For processing to be possible (and for the “Process input”-button to become active), as mentioned, at least a task model has to be loaded and either an Observer log file or an ADIOS log file. The other inputs are optional. Once the “Process input”-button is clicked, the loaded log files are used to generate the internal operator state log. The button
“Export to XML” (region 5, figure 3.9) will then become active and upon clicking it, a save dialog will be displayed for saving the operator state log to an XML file.

After pressing the “Process input”-button, there is data available for visualization. This is indicated by the “Plotting data available” tick box (region 4, figure 3.9). By clicking the visualization tab (region 1), the user navigates to the visualization section. The visualization interface of the SOWAT tool is shown in figure 3.10.

In the upper left corner of the visualization interface, the data selection section is located (region 1, figure 3.10). It shows the length of the recorded data and it enables the user to make a selection from that period for visualization. The user has to select the following parameters:

- The starting time (st) for the plotting period
- The ending time (et) for the plotting period
- The amount of CTL points to represent the selected period
- The skill level for the current operator (user model)
- The measurement to use for LIP calculation
- Whether or not to plot a line through the plotted points (can be toggled after visualization)

The user has 3 options for the LIP calculations, as shown in figure 3.11.

- If no Workload Watch (WLW) log has been loaded, only the LIP contained in the task model can be used and the other options will be disabled.
- When a Workload Watch log has been loaded, the LIP can also be calculated using the subjective complexity values contained in it.
- Finally, the user can also select to use the average of the task model values and the subjective values.

![LIP source selection](image)

*Figure 3.11: Selecting which source to use for the LIP calculation*

When using the task model (or the average) for calculating the LIP, if the SOWAT tool encounters a (sub)task that is not present in the task model, it will use the default LIP for that (sub)task. The default LIP is specified in the SOWATconfig.ini file that is in the same directory as the SOWAT executable. If the SOWATconfig.ini file is not present, it will use 3 as the default LIP value. After the plotting, a message will be displayed showing all switches to non-existing (sub) tasks, as shown in figure 3.12. This is an indication that the task model should be revised.

![Warning message](image)

*Figure 3.12: A switch to a non-existing (sub)task has occurred*

Once the desired data selection options have been provided, the “Plot”-button can be clicked. This will result in plots of the data in the presentation area (region 2, figure 3.10). First, a 3D CTL cube is plotted. Around the cube, the projections of the three sides of the cube are plotted, for easier reading. The 3D cube can be rotated by pressing the left mouse button in the cube and then moving the mouse while holding the left mouse button.

On the right side of the presentation area, additional plots displaying all recorded variables instantiated for the selected points are shown. The user can switch between the available plots using the tab pages. The tab pages available depend on the different input sources that have been loaded. The possible available tab pages are:

- CTL1: TO and LIP
- CTL2: TSS
- WLW: Subjective effort and Subjective complexity
- Expert: Expert rating
- Heart: Heart rate
- Resp: Upper and lower respiratory values
- Eye: Pupil size

In the lower left corner, a table is displayed containing the values of all variables for each cube point (region 3, figure 3.10). Selecting a row in this table will highlight the corresponding points in the plots, as shown in figure 3.13.
Figure 3.13: The points corresponding to the selected row in the table are highlighted in the plots.

After data has been plotted, the user has several options to manipulate the plots. In the data manipulation section (region 4, figure 3.10), the user can select how the points in the plots are colored. By default, the points are plotted with decreasing opacity.

If a workload watch file was loaded, the option to color the points according to the subjective effort rating becomes available. This will color the points in all plots in three colors: green (low effort), orange (medium effort) and red (high effort). The thresholds for the coloring of the points with subjective effort can be set using the controls in the middle of the data manipulation section.

If an expert rating file was loaded, the option to color the points according to the expert rating becomes available. This will color the points in all plots in three colors: green (high expert rating), orange (medium expert rating) and red (low expert rating). The thresholds for the coloring of the points with the expert rating can be set using the controls on the right side of the data manipulation section.
When the user is satisfied with the selected data, he can press the “Save statistics”-button (region 5, figure 3.10). This will open a save dialog to save a text file containing the table that has been plotted as well as statistics such as average LIP over the entire experiment, total TSS over the entire experiment etc. If the user presses the “Save statistics and graphs”-button, the statistics file will also be saved along with jpg-images of all plots.

A sample output of the SOWAT tool is shown below:

SOWAT statfile
Created at: 7/6/2006 4:25:12 PM
---

Length of recording: 00:17:19

[Processed Files]
Observer File:    F:\Adiosdata\adios 2 tm 30 (8).txt
Task Model File:  F:\Adiosdata\Adios1.xml
Workload Watch File: F:\Adiosdata\memorycard\Results Duijts op 09-02 08.59.txt
Expert Rating File: F:\Adiosdata\Expert ratings\Adios22_Duijts_Jepkes.XML
Mobi File:        F:\Physiology\Mobi_1.txt
Tobii File:       F:\Physiology\Tobii_1.txt

[TSS]
Total: 79
Average per minute: 4.56

[LIP]
LIP source: Task model
Average: 2.73

[TO]
Total: 82%

[Subjective Ratings]
Average subjective effort: 2.16
Average subjective complexity: 1.98

[Expert Ratings]
Average expert rating: 0.31

[Mobi Data]
Average Heart Rate: 69.86

[Tobii Data]
Average Pupil size Right: 4.30
Average Pupil size Left: 4.44

[Visualized Data]
Start time: 5:00
End time: 10:00
Skill level: intermediate
LIP source: Task model
Point Coloring: Gradient
### 3.4 SOWAT tool usage

In this section we describe a general procedure for analysis of experimental data, using the SOWAT tool. Some example analyses will be presented in chapter 4.

#### Step 1: Observer scoring

Generally, the Observer is used as basis of the SOWAT analysis. The subjects are filmed during the experiments; the experimenter now has to use the Observer software to score the actions of the subjects as accurately as possible. Once this is done, the Observer analysis has to be exported to a file.

If only ADIOS logs are used for analysis, this step can be skipped.

#### Step 2: Creating the task model

Next, the task model has to be created. The experimenter will already have a list of tasks that can possibly occur and their decomposition into subtasks, since this is the basis of the experiment itself. This information has to be entered into the task model. The easiest way to do this is to parse an Observer (and/or ADIOS) log file in the Task model authoring tool and to create a task model tree from it. This ensures that the tasks are named correctly in accordance with their occurrence in the Observer (and/or ADIOS) log files.

After the task model structure is complete, the LIP values of each action have to be filled in. If the group of test subjects is homogenous, then only one LIP value has to be filled in for each action. If the skill differs between the test subjects, then up to 3 different LIP values can be added to the task model for each action.

#### Step 3: Testing the task model

Before moving on, it is advisable to test if the task model contains all possible actions occurring in the Observer (and/or ADIOS) log files. To do this, the task model and an Observer (and/or ADIOS) log file should be loaded into the SOWAT tool and processed.

In the visualization section, a visualization of the entire period should be selected. If a nonexistent action is encountered, the SOWAT tool will give an error, indicating the name of the action. The task model can then be revised to also incorporate the missing actions. Repeat this procedure for different Observer (and/or ADIOS) log files.
Step 4: Creating expert rating files
Once the task model has been verified, it can be used to create expert ratings. Load the task model into the expert rating authoring tool, to generate an expert rating tree. The experimenter now has to fill in the expert ratings for each action.

Step 5: Analysis
Now all input sources are available and they can be processed by the SOWAT tool. Load all relevant input sources for the subject under analysis and process them. Manipulate the visualizations to investigate periods of interest and to color the graphs according to the performance and effort measures. This is best done while also watching the video footage through the Observer software (if available). This way it is clearer what is happening in the selected period.

3.5 Implementation details
In this section we will look into some implementation details of the SOWAT tool. The SOWAT tool was implemented in C#, using Microsoft Visual Studio .NET 2003.

3.5.1 Operator state log
The core of the SOWAT tool is the Operator state log. This records the state of the operator over the course of the experiment. The state log is essentially a list of state entries. A state entry has the following form:

```
structure State
{
    TimeStamp
    CurrentTask
    SubjectiveComplexity
    SubjectiveEffort
    ExpertRating
    HeartRate
    RespUpper
    RespLower
    PupilLeft
    PupilRight
}
```

The information in this log is derived from the different loaded input files. First, the Observer file (or ADIOS file) is parsed. For every state change detected in the Observer file, a log entry is made, containing the timestamp and the current task. If state changes are more that one second apart, a log entry is made for every second in between the state changes. When the list of entries has been created this way, the other input sources are parsed and for every state entry in the list, the relevant data is collected and added to the entries.
3.5.2 Calculating Taskload data

After the operator state log has been created, it is up to the user to select a period of interest and the desired number of points to represent that period. With these parameters, the taskload data points are derived from the operator state log. For each of the taskload data points, the values for the variables over that period are calculated. This is done in a similar fashion for all variables. As an example, the calculation of Time Occupied over a given period is given in pseudo code below.

```
GetTimeOccupied(beginTime, endTime)
{
    TimeOccupied = 0
    OccupiedTime = 0

    // find the first log entry with a timestamp greater than or equal to beginTime
    firstStamp = BinarySearchElement(beginTime);

    // find the first log entry with a timestamp greater than or equal to endTime
    afterLastStamp = BinarySearchElement(endTime);

    // if the selected interval is so small that it falls within 2 measurement points, then the answer is straightforward
    if (firstStamp == afterLastStamp)
    {
        if (StateLog[firstStamp-1].CurrentTask == "idle")
            TimeOccupied = 0
        else
            TimeOccupied = 100
    }
    else
    {
        // at least one measurement present within interval
        // calculate TO for points within interval up to one point outside interval
        for (i = firstStamp; i < afterLastStamp; i++)
        {
            if (StateLog[i].CurrentTask != "idle")
                OccupiedTime += StateLog[i+1].TimeStamp - StateLog[i].TimeStamp
        }
        // Add pre interval part
        if (firstStamp != 0)
        {
            if (StateLog[firstStamp - 1].CurrentTask != "idle")
                OccupiedTime += StateLog[firstStamp].TimeStamp - beginTime;
        }
        // Subtract post interval part
        if (StateLog[afterLastStamp - 1].CurrentTask != "idle")
            OccupiedTime -= StateLog[afterLastStamp].TimeStamp - endTime;
    }
}
```
// divide occupied time by total time
TimeOccupied = (OccupiedTime / (endTime - startTime)) * 100;

    return TimeOccupied
}

3.5.3 Visualizations

For the visualizations of the data in the tool, we used two open source ActiveX components by Nicolai Teofilov:
- NTgraph (http://www.codeproject.com/miscctrl/NTGraph_ActiveX.asp) was used for the 2D plots
- NTGraph3D (http://www.codeproject.com/opengl/ntgraph3d_atl.asp) was used for the 3D plot.

These components were implemented in C++, but with a few adjustments they were made suitable for usage in the C# environment. Most of these adjustments concerned the difference in data types used in C++ and C#.
4. Experiments and analysis

To test the practical use of the different taskload measurements and to obtain measurement data to base the design of the workload assessment tool on, three experiments were conducted. Section 4.1 describes ADIOS 1, an experiment conducted at the Royal Netherlands Navy institute. Section 4.2 describes OLA 2, an operator load assessment experiment on board of 3 Dutch navy frigates. These experiments were conducted for the promotion research of Marc Grootjen. I assisted in these experiments and was involved in the selection process of measurements. The design of the SOWAT tool is based on the information gathered in these experiments. After the design of the SOWAT tool, some improvements were suggested for the ADIOS software. These improvements enabled the SOWAT tool to derive taskload, using the ADIOS logs more accurately. With these improvements for the ADIOS software, the ADIOS 2 experiment was conducted for the research of Kevin Stolk and Machiel Markelbach. This experiment is described in section 4.3. With the data from this experiment, the SOWAT tool was improved. Finally, section 4.4 shows some examples of analysis using the SOWAT tool.

4.1 ADIOS 1

4.1.1 Experiment design

The goal of the ADIOS 1 experiment was to establish differences in the performance of operator couples, using different task allocation support functions.

For the experiment, participants with a maritime background were needed. To achieve a homogeneous group, only first and second year cadets were selected to participate. 64 participants, 56 males and 8 females between 18 and 27 years of age (mean 19.5, SD 1.63), of the Royal Netherlands Navy College were selected. The participants were obligated to participate. To stimulate their effort, 3 couples with the highest performance were entitled to a reward of 100 euros. The participants had to perform a computer task using an adaptive interface developed at TNO (the ADIOS software as described in section 3.1.2), the program simulates the working of the Ship Control Centre (SCC) of a navy ship. The participants had to solve the occurring problems together, as well and fast as possible. Every couple had to deal with 4 fire alarms, a low pressure alarm of the fire-fighting system, 3 alarms concerning the cooling system of the ship, one bilge water alarm and a high temperature diesel engine alarm. How to deal with these alarms was described in a predefined procedure. Depending on the performance of the couples, the scenario took about 20 minutes. Before starting the scenario, an instruction was given (60 minutes) and the participants used the system during a training scenario (20 minutes). Figure 4.1 shows a couple during the experiment. They were allowed to talk to each other about everything, however it was forbidden to look at each other’s screen.
The experiment consisted of three conditions with a between subjects factor ‘task-allocation (TA) support’ as independent variable:
1. No task allocation support, only participants together determine task allocation.
2. System provides advice for task allocation.
3. System automatically reallocates tasks, just informing the operator of its decision.

In addition to the role of the system in conditions 2 and 3, the operators were still able to reallocate tasks themselves. Timing of advice messages and automatic TA were the same and determined in advance using the CTL method (Neerincx, 2003). Table 4.1 shows the experimental design.

<table>
<thead>
<tr>
<th></th>
<th>No TA support</th>
<th>TA advice</th>
<th>TA auto</th>
</tr>
</thead>
<tbody>
<tr>
<td># couples</td>
<td>11</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

*Table 4.1: Experiment design*

### 4.1.2 Measured data

- All subjects and their computer screens were filmed during the entire experiment. Afterwards, these recordings were analyzed using the Observer software. With the Observer software it was recorded exactly what tasks the subjects were performing at all times during the experiment.
- During the experiment, the subjects were asked every minute to rate the complexity of the past minute and their effort during the last minute, using the Workload Watch program running on a PDA that was located on the desk of each subject.
- From about half of the couples, physiological data was collected. The reason it was only collected from half of the couples, was because most of the time two experiment sessions took place at the same time, while there was only equipment available for measuring one couple at a time. Using the Mobi8 device, heart and respiratory measures were collected as described in section 3.1.5 of this report.
• After the experiment, the subjects were asked to fill in a questionnaire, which contained questions about different aspects of the ADIOS software, specifically questions about the usability and usefulness of the alarm redistribution. Also, they were asked to rate their mental workload over the entire experiment on a BSMI scale as mentioned in chapter 2.2.2.
• (team)performance, measured in two ways:
  1. The number of correct actions
  2. The time used to solve the problems from the moment they appeared

4.2 OLA 2

4.2.1 Experiment design

The OLA 2 experiment evaluates the Ship Control Centre (SCC) of the Air Defense and Command Frigate (ADCF). For the experiment, 3 scenarios were designed with the method for Cognitive Task Analysis (Neerinck, 2003), imposing 3 different levels of CTL on the operators. These scenarios were performed by a total of 12 operator-manager couples (figure 4.2), on three different ADCFs.

![Figure 4.2: An operator-manager couple during the experiment](image)

Depending on the performance of the couples, each scenario took on average about 20 minutes. Before starting the scenario, a test scenario was done and an instruction was given.
The experiment took place at sea. The problems during the scenario were simulated, but all actions had to be operated on the actual systems.

4.2.2 Measured data

• During the experiment, 2 Observers were used to record 4 video and 4 audio streams.
• 2 audio streams were recorded with a high quality system, which will be used (in a later stage) for voice-stress analysis. The other 2 streams were recorded for evaluation purposes, together with the videos.
• 2 of the videos were frontal recordings of the operators. These videos will be used (at a later stage) for facial stress analysis. The other 2 videos were used for evaluation purposes.
• Directly after performing the scenario the participants evaluated it. During this evaluation they had to rate complexity and effort every minute using the WLW software on a handheld computer.
• Using the Mobi8 device, heart and respiratory measures were collected and using the Tobii, pupil dilation information was recorded.
• Participants were asked to rate their mental effort after each scenario on a BSMI scale as mentioned in section 2.2.2 of this report.
• Complexity was scored independently by experts on a five point scale and was added to the task model.
• Performance was measured in two ways
  1  The number of correct actions
  2  The time used to solve the problems from the moment they appeared

4.3 ADIOS 2

4.3.1 Experiment design

The ADIOS 2 experiment was similar to the ADIOS 1 experiment. The same software was used with some adjustments and improvements. The goal of the ADIOS 2 experiment was not to test differences in task allocation support functions (like the ADIOS 1 experiment), but to establish differences in the performance of operator couples, using different modes of automation.
The experiment consisted of three conditions with a between subjects factor ‘mode of automation’ as independent variable:
1. Normal mode, operators perform all tasks manually
2. Advice mode, operators are given advice by the system how tasks should be executed
3. Concur mode, the system performs tasks with approval by the operators

4.3.2 Measured data

The same data was collected as during the ADIOS 1 experiment. However, no physiological measures were taken.

4.4 Example analysis

The analysis of these experiments falls outside the scope of this project. The data gathered in these experiments was used to design and test the SOWAT tool. The SOWAT tool will be used for future analysis of this experimental data. Depending on what the experimenter is interested in, the SOWAT tool can be used for different types of analyses. Some examples of analyses are presented in the following.
Coloring using subjective effort:

The SOWAT tool can be used to visualize the relationship between the CTL values and the user’s subjective effort rating. This way, critical regions in the cube for specific operators can be located. To do so, we need to load an Observer log (or ADIOS log), the corresponding workload watch log and the task model. After selecting a period of interest and plotting, the threshold for coloring can be manipulated to best indicate the relationship. An example of this is shown in figure 4.3.

Here we plotted an entire session from the ADIOS 1 experiment (just over 17 minutes) for one subject, using 30 points. All points with a subjective effort of 2 or lower are colored green, all points with a subjective effort greater than 4 are colored red, all points in between are colored orange. In this specific example there is a nice correlation between the CTL values (LIP in particular) and the subjective effort ratings. This is especially the case when the subjective complexity is used as a LIP source, but also for the task model and average modes.
If the results using the task model are very different from the results using the subjective complexity for a large number of subjects, the experimenter may want to consider revising the task model, since his perception of the difficulty of the different tasks is then clearly different from the perception of the subjects.

*Coloring using expert ratings:*

SOWAT can also visualize the relationship between the CTL values and the expert ratings, which can also indicate CTL regions of interest. To do so, we need to load an Observer (and/or ADIOS) log, the corresponding expert rating file and the task model. For the ADIOS 1 experiment, the expert ratings were established as follows:

If the couple executed an action the wrong way, they got an expert score of 0. If the action was executed correctly, their score depended on the time they needed to complete the action. For each action there was a specified execution time. If the couple finished a task within this time, they got a score of 2. If they finished the task within 2 times the specified time, they got a score of 1, otherwise their score was 0.

An example of a plot from the ADIOS 1 experiment, colored with expert ratings is shown in figure 4.4. This example uses the same data as the previous subjective effort example. All points with an expert rating of 0 are colored red, all points with an expert rating greater than 0.9 are colored green, all points in between are colored orange. All points colored black have no expert rating and should be disregarded.

In this example, the correlation between the CTL values and the expert ratings is not clear. The coloring of the points seems quite random. This behavior was encountered for most expert rating colorings analyzed so far. This may indicate the need to change the way expert ratings are determined.
Visualizing operator taskload flow:

If not too many points are plotted at the same time and the line between points is activated, the taskload flow of the operator over the course of the experiment can be visualized. This is especially meaningful once interesting regions in the cube have been identified. To do this, an Observer log (or ADIOS log) needs to be loaded together with the task model. An example of an operator path through the cube is shown in figure 4.5. In this example, every cube point corresponds to 1 minute of time.
Figure 4.5: The path of an operator through CTL space

CTL information over the entire session:

In the case of the ADIOS 1 experiment, there are 3 different task allocation conditions. It is interesting to compare the values of the 3 CTL variables over the entire session for the couples in the different conditions. To do this, for each subject the corresponding Observer log should be loaded, together with the task model. After processing these inputs, the “save statistics” button will save the desired information (among other things) to a text file.

For the ADIOS 1 experiment, we list the average CTL values and average experiment times for the 3 different conditions in table 4.2.

<table>
<thead>
<tr>
<th></th>
<th>No TA support</th>
<th>TA advice</th>
<th>TA auto</th>
</tr>
</thead>
<tbody>
<tr>
<td>TO</td>
<td>86.05</td>
<td>85.8</td>
<td>84.94</td>
</tr>
<tr>
<td>TSS</td>
<td>85.55</td>
<td>76.15</td>
<td>78.82</td>
</tr>
<tr>
<td>LIP</td>
<td>2.90</td>
<td>2.86</td>
<td>2.93</td>
</tr>
<tr>
<td>Time</td>
<td>16:38</td>
<td>16:27</td>
<td>17:09</td>
</tr>
</tbody>
</table>

Table 4.2: CTL averages for the 3 conditions of the ADIOS 1 experiment

The table shows that the condition without TA support noticeably increased the amount of task switching. With the ADIOS 1 experiment, communication between the two subjects was scored as a separate task set. The increase in task switches for this condition may be caused by the increased need of communication between the subjects, when no task allocation support was available.
Comparing between couples:

In the experiments, couples had to work together to solve problems. Presumably the best way to do so is for them to evenly divide the work. It is interesting to compare the taskload of both subjects over the course of the experiment. As an example, we compare the taskload of the 2 members of a couple from the ADIOS 1 experiment below:

Member 1:

[TSS]
Total: 78
Average per minute: 4.50

[LIP]
LIP source: Average
Average: 2.86

[TO]
Total: 88%

[Subjective Ratings]
Average subjective effort: 2.75
Average subjective complexity: 2.68

<table>
<thead>
<tr>
<th></th>
<th>LIP</th>
<th>TSS</th>
<th>TO</th>
<th>Effort</th>
<th>Compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.79</td>
<td>2.00</td>
<td>20%</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>1.93</td>
<td>5.00</td>
<td>74%</td>
<td>1.00</td>
<td>1.98</td>
</tr>
<tr>
<td>3</td>
<td>2.17</td>
<td>7.00</td>
<td>97%</td>
<td>3.00</td>
<td>2.00</td>
</tr>
<tr>
<td>4</td>
<td>3.93</td>
<td>7.00</td>
<td>99%</td>
<td>3.00</td>
<td>3.96</td>
</tr>
<tr>
<td>5</td>
<td>3.32</td>
<td>5.00</td>
<td>93%</td>
<td>3.00</td>
<td>3.05</td>
</tr>
<tr>
<td>6</td>
<td>3.51</td>
<td>5.00</td>
<td>95%</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>7</td>
<td>2.77</td>
<td>8.00</td>
<td>94%</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>8</td>
<td>3.00</td>
<td>4.00</td>
<td>100%</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>9</td>
<td>3.96</td>
<td>2.00</td>
<td>100%</td>
<td>3.73</td>
<td>3.73</td>
</tr>
<tr>
<td>10</td>
<td>3.29</td>
<td>4.00</td>
<td>86%</td>
<td>3.31</td>
<td>3.31</td>
</tr>
</tbody>
</table>

[END]

Member 2:

[TSS]
Total: 79
Average per minute: 4.56

[LIP]
LIP source: Average
Average: 2.36

[TO]
Total: 82%
[Subjective Ratings]
Average subjective effort: 2.16
Average subjective complexity: 1.98

<table>
<thead>
<tr>
<th></th>
<th>LIP</th>
<th>TSS</th>
<th>TO</th>
<th>Effort</th>
<th>Compl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.53</td>
<td>5.00</td>
<td>60%</td>
<td>2.00</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>2.19</td>
<td>7.00</td>
<td>87%</td>
<td>3.99</td>
<td>2.00</td>
</tr>
<tr>
<td>3</td>
<td>2.71</td>
<td>3.00</td>
<td>83%</td>
<td>4.00</td>
<td>2.98</td>
</tr>
<tr>
<td>4</td>
<td>3.28</td>
<td>6.00</td>
<td>99%</td>
<td>1.41</td>
<td>2.14</td>
</tr>
<tr>
<td>5</td>
<td>2.52</td>
<td>3.00</td>
<td>100%</td>
<td>1.70</td>
<td>2.00</td>
</tr>
<tr>
<td>6</td>
<td>3.03</td>
<td>7.00</td>
<td>100%</td>
<td>2.67</td>
<td>2.67</td>
</tr>
<tr>
<td>7</td>
<td>3.51</td>
<td>3.00</td>
<td>99%</td>
<td>3.62</td>
<td>3.00</td>
</tr>
<tr>
<td>8</td>
<td>3.06</td>
<td>8.00</td>
<td>99%</td>
<td>3.47</td>
<td>3.00</td>
</tr>
<tr>
<td>9</td>
<td>3.13</td>
<td>5.00</td>
<td>94%</td>
<td>3.42</td>
<td>3.42</td>
</tr>
<tr>
<td>10</td>
<td>3.95</td>
<td>7.00</td>
<td>99%</td>
<td>4.37</td>
<td>4.37</td>
</tr>
</tbody>
</table>

[END]

In this example we see that both the TO and average LIP of member 1 are higher than those of member 2. Also subject 1 rates his effort and complexity higher. This can be an indication that the work could have been divided better, or that the perceived difficulty of the tasks by the subjects was different.
5. Conclusions and future work

5.1 Conclusion and discussion

With the implementation of the SOWAT tool, it has implicitly been shown that operator taskload can be detected, quantified and visualized. The tool successfully combines different measures into a quantification of taskload. Since taskload can be detected and quantified, it is possible to formulate task allocation and support strategies, based on thresholds of this taskload quantification. The SOWAT tool provides functionality to determine taskload thresholds for these strategies. This provides a first step towards the implementation of the adaptive support framework.

Two main applications for the SOWAT tool can be distinguished:
- Analyze data to establish critical and optimal regions in CTL space for (groups of) operators.
- Visualize an operator’s path through CTL space.

In future research, the focus should be on further expanding the possibilities to determine the interesting regions in the CTL cube and on implementing real-time task load assessment. This way different task allocation strategies, based on task load, can be constructed and tested.

This project is not the only attempt at operator load assessment with the intention to provide operator support. Fink et al. (2005) describe the design of a workload monitoring and visualization tool that combines different (physiological) measurements into a one-dimensional expression, used as indication of workload. This quantification is obtained by assigning weights to the different measurements and normalizing them into one value. Nickel and Hockey (2006) describe research into identifying operator load levels (risk states) using physiology. Risk states are forced by stepwise changes in operator load. For each state the physiological measures are analyzed. This way they attempt to link certain (person dependent) physiological characteristics to the different load levels.

What makes the method used in our research different is that it takes advantage of the task-driven nature of the domain. The CTL method expresses operator load in relation to the performed tasks. Once critical regions in CTL space are identified for a person with usage of effort and performance measures, the CTL measurements by themselves are enough to determine operator load and the need for support.

5.2 Future research and improvements

In this section we will suggest some interesting areas of future research and possible improvements to the SOWAT tool.

5.2.1 Adaptive Hypermedia and the DTA framework

In chapter 2, the DTA framework for dynamic task allocation is presented. At the start of this project, Dr. Alexandra Cristea pointed out that it could be interesting to compare the
DTA framework to the LAOS model from the Adaptive Hypermedia research area. While the actual integration of the Adaptive Hypermedia discipline with this research fell out of the scope of this project, we acknowledged the relevance of this similarity and performed a literature study comparing these two models. This study is presented here and it can serve as a starting point for future research into this subject.

Adaptive Hypermedia and the LAOS model
Adaptive hypermedia is a research field that arises because of the limitations of traditional “static” hypermedia. Traditional hypermedia present the same information in the same way to all users regardless of the differences between users (Brusilovsky, 2001). Adaptive hypermedia strives at dynamically presenting hypermedia content to the user by taking into account the user’s knowledge gained so far by the content he already read and taking into account user characteristics and preferences. Brusilovsky states that an adaptive hypermedia system has to satisfy 3 criteria: it is a hypermedia system, it has a user model and it is able to adapt the hypermedia using the user model. Some forms of adaptive hypermedia applications also adapt to the environment (Abi-Aad et al., 2003; Cheverst et al., 2002).

An adaptive hypermedia system often keeps track of the knowledge of the user by recording which parts of the hypermedia documents have already been read by the user. It guides the user’s flow through the hypermedia content according to this knowledge, for example by linking to the pages the user is ready to read, based on his current projected knowledge and by hiding links to parts of the hypermedia content that are not ready for the user to read. Also, adaptive hypermedia strives to adapt the presentation of the content and the flow through the content, based on user characteristics such as cognitive or learning styles (Stash et al., 2004) and other personal preferences. The ultimate goal of adaptive hypermedia is to optimize user efficiency in achieving his (information acquiring) goals by providing user support in the form of content and presentation adaptation.

At Eindhoven University, the LAOS model has been developed (Cristea and De Mooij, 2003). It is a five layer model for authoring adaptive hypermedia for the World Wide Web, based on the AHAM model (De Bra et al., 1999; Wu, 2002), which in turn is based on the Dexter model (Halasz and Schwartz, 1994). The five layer LAOS model is depicted in figure 5.2. The LAOS model is based on the book-presentation metaphor. When designing a presentation, be it a live presentation or a presentation on the web, the information from the presentation is coming from one or more books. The content of these books is then ‘filtered’ by the goal of the presentation. In other words, the relevant information for the presentation is derived from the book(s), according to the goals of the presentation. The presentation is adjusted to fit the intended audience and the presentation is constructed in a way to fit the presentation medium.
Figure 5.2: The LAOS model for Adaptive Hypermedia

We will explain the LAOS model using the book metaphor.

**Domain Model (DM):**
The domain model corresponds to the book(s). In the domain model, all information on the subject is specified. The basic “units” of information in the DM are concepts. A concept can be seen as a paragraph from a book. The DM also defines the relationships between the different concepts.

**Goal Model (GM):**
The goal model defines what part of the DM (the books) is relevant for this particular presentation, considering the goal of the presentation. So, it more or less makes a selection from the concepts in the DM.
This enables the Adaptive Hypermedia author to construct different presentations from the same DM. Also, creating one presentation based on different DMs is possible.

**User Model (UM):**
The user model keeps track of the information that has already been viewed by the user (the audience of the presentation). This information is used to determine what parts of the information are currently relevant for the user, based on what he has already seen. Additionally, the UM can define specific characteristics of the user, such as learning styles. These characteristics can also be used to adapt the presentation.

**Presentation model (PM)**
The presentation model defines the characteristics of the presentation medium and defines how the presentation can be adapted to it.

**Adaptation Model (AM):**
The adaptation model defines how the information from the user model is used to adapt the presentation and it defines how the information from the user’s interaction with the system is used to update the user model. So in the AM the adaptation strategies are defined.

**Comparing the models**
While the DTA framework and the Adaptive Hypermedia discipline, specifically the LAOS model, have different domains and background, at a high level they have the same purpose: Supporting the user in the best way possible in order to optimize his working performance. For the development of the DTA framework it can therefore be profitable to look at the adaptive hypermedia domain for ideas, especially for automation and user adaptation. At an abstract level, parallels can be drawn between some of the models contained in the DTA framework and the LAOS model:

**Operator Model and User Model**
Both the Operator model from the DTA framework and the User model from LAOS try to capture information about the characteristics of the user of the system.

**Task Model and Domain model:**
In the DTA Task model, the characteristics of all tasks that the operator may have to perform are modeled. In the Domain model, all of the content of the hypermedia system is defined. So both models define basic units of information.

**Allocator and Adaptation Model**
Both define the rules that state how the information in the models is used to generate the output of the system.

**System Model and Presentation Model**
In these models, the information about the characteristics of the system is defined. They both state what can be adapted in their respective systems.

**Goal Model:**
There is no direct parallel between the Goal Model and one of the models of the DTA framework. However, since the Goal Model essentially makes a selection of the information in the Domain Model, it can correspond to the currently active tasks in the
DTA framework. This is not explicitly modeled in the DTA framework, however this important piece of information should be either be a separate model or it should be contained in the Operator Model.

A significant difference between the DTA framework and LAOS is that in LAOS, the information in most of its models is defined beforehand by the designer of the AH system with a specific goal in mind. The volatile information is traditionally contained in the User Model and to a smaller degree in the Presentation Model.

In the DTA framework there is also some static information present (i.e. the Task Model), but most of the information is dynamic and can change during the usage of the system. Also the ways in which the information in the models is gathered and altered are more diverse. The diverse nature of state changes and the degree of volatility of the information, provide a challenge to apply the Adaptive Hypermedia paradigm to the DTA framework.

Now that a global high level comparison was made between the models, I suggest 2 characteristics of the Adaptive Hypermedia area that stand out as useful for investigating for usage in the DTA context.

- First, the concept of User Modeling. User modeling is one of the key concepts of Adaptive Hypermedia and it is interesting to look at user model ontologies used in Adaptive Hypermedia to base the design of the operator model in the DTA framework on.
- Second, the design of the Adaptation Model. In the adaptation model, the rules for adaptation of the system are defined. These rules are defined using an adaptation grammar. Such a grammar will also be needed for the DTA's Allocator.

Note that these characteristics comprise two of the three characteristics of the definition of an Adaptive Hypermedia system. Next we will discuss these 2 features in more detail.

**User modeling**

The most complete type of user models is defined via an ontology. An ontology formally defines a common set of terms that is used to describe and represent a domain ([http://www.w3.org/TR/webont-reg/](http://www.w3.org/TR/webont-reg/)). In Adaptive Hypermedia (and other disciplines using user models) sometimes User Modeling is done in the form of ontologies. The ontology describes how the user’s characteristics of interest are modeled and what the relationships between these characteristics are. Ontologies are expressed in an ontology language. A common language for expressing Ontologies is OWL, which stands for Web Ontology Language ([http://www.w3.org/TR/owl-features/](http://www.w3.org/TR/owl-features/)).

However, most of the time simpler types of User models are used, that still provide quite powerful adaptation possibilities. These User models simply contain a set of variables with corresponding values. In the implementation of the SOWAT tool, the skill level of the operator is modeled in this way. We define the skill level of the operator as being either beginner, intermediate or advanced. Based on this value we adjust the implicit complexity of the tasks the operator performs. If we model more aspects of the user in this way and we combine it with an adaptation language (as discussed further on), some rather powerful adaptation strategies can be expressed. This way it will for example be possible to define in the user model the CTL values corresponding to the critical regions in the cube for a specific operator and applying a certain adaptation strategy when these values occur.
Adaptation model Grammar

In LAOS, the way the User Model is updated and the way the adaptation of the Presentation to the User Model takes place, is defined in the Adaptation Model. In the Adaptation Model a set of adaptation rules is expressed, based on a grammar. These rules define the adaptation strategy. An adaptation engine is able to interpret the grammar and perform the actual adaptation.

At Eindhoven University the LAG model (Cristea and Calvi, 2003) is being developed as a refinement of the Adaptation Model. The LAG model defines 3 levels of expressivity that build on each other.

- At the lowest level, direct adaptation rules of the form `<if> condition then <action>` are defined. These rules for example update a specific user model variable. This level can be seen as the assembly language of the Adaptation model.
- On top of that, a context free grammar is defined that makes writing adaptation rules more intuitive for humans and makes reuse of these rules possible. This can be seen as the programming language of the Adaptation Model.
- Finally, a set of rules defined in the second layer can be grouped into strategies that can be combined and can be applied to the system. This can be seen as the programs of the Adaptation Model.

It would be beneficial for the development of the DTA framework to define such a grammar for the allocator. This way, different strategies can be implemented and tested easily. The LAG grammar can serve as a starting point for such a language. It will need to be extended in order to handle not only interface and user model adaptations, but also task allocation. As a simple example we express the switching of the system to a high level of automation, as a snippet of a strategy expressed with the help of the LAG grammar:

```
IF (enough (UM.LIP > UM.overload_LIP
         UM.TSS > UM.overload_TSS
         UM.TO > UM.overload_TO , 3))
(PM.automationlevel = high)
```

The code above checks if the current 3 CTL variables of the operator are above overload thresholds, as defined in the user model (operator model) and alters the automation level variable of the system (presentation model).

5.2.2 Real-time SOWAT

The current implementation of the SOWAT tool provides offline, post-hoc taskload analysis of experimental data. A new version of the SOWAT tool could provide the first step towards real-time taskload assessment.

The current SOWAT tool is able to derive an operators taskload by using a task model and a logfile from the ADIOS software (as covered in section 3.1.2). It would be very interesting to integrate the SOWAT tool with the ADIOS software, so that this can be done in real-time, during the actual operation of the ADIOS software.
Since the ADIOS software has functionality for task allocation, it is then possible to base this task allocation on the actual taskload of the operators. Currently this is done manually by the operators and automatically by the system at preset times. This way the effect of different allocation strategies on the performance of the operators can be investigated and a first taskload driven task allocation system can be designed.

5.2.3 Determining CTL interval

The SOWAT tool can visualize CTL over an arbitrary period, using an arbitrary amount of points. In order to base adaptive support on CTL values, at some point it has to be established what a suitable interval is for CTL measurements. If this interval is taken too large, the CTL data will be averaged out too much. On the other hand, if the interval is taken too small, the CTL data will fluctuate too much. Both will not produce suitable results for adaptive support. The interval should be established at a length where the data is not averaged out too much to contain useful taskload information and where the data does not fluctuate too much to be reliable.

In real-time task load assessment, this interval should be used as a moving time window.

5.2.4 Visualizations

In this implementation of the SOWAT tool, the visualization of the data is done in a straightforward manner, based on existing visualization components. There is room for improvement here, for example:

- A 3D plot specifically aimed at visualizing the CTL cube. Currently the 3D cube is not really easy to read. While this is solved to some extent by providing the 2D side projections, it can be done better. For example by enabling hiding of individual points and groups of points, in order to resolve the issue of overlapping points.
- Plotting the data of multiple operators in the same cube. This makes comparison of data easier.
- Plotting different data against each other. Currently, only the 3 CTL variables are plotted against each other, while all other data is plotted against time. Plotting 2 arbitrary variables against each other may yield some interesting results.

5.2.5 Interpretation of physiology

The role of physiological measures in the current version of the SOWAT tool is rather limited. The physiological variables are simply plotted together with the other data. Since physiological parameters can serve as an indicator of user effort (as discussed in section 2.2.1), the coloring of the cube points using physiological data can help provide insight into the location of interesting regions in the cube. However, before this information can be used in the SOWAT tool, the collected physiological measurements need to be processed by experts first, in order to derive the relevant physiological parameters. Also more different measurements could be added, considering the overview given in section 2.2.3.
It should be attempted to combine multiple physiological measurements into one single expression of physiological user effort and coloring cube points according to thresholds of this physiology index. Several techniques can be used for this, including artificial neural networks, pattern recognition and data mining (Wilson et al., 1999; Gevins and Smith, 2006; Liu et al., 2005; Fink et al., 2005).
References


Appendix A: SOWAT User Manual

This manual discusses the features and usage of the SOWAT tool. This information overlaps with the information contained in the main report. However this manual will discuss the features and usage of the tool in more detail and is intended as a separate document, accompanying the tool itself.

Index

1. Introduction .......................................................... 69
2. System requirements ............................................. 69
3. Installation .......................................................... 70
4. Data collection ...................................................... 70
  4.1 Data sources and characteristics .......................... 70
  4.2 The Data Collection interface .............................. 77
  4.3 Task Model Editor ............................................. 78
  4.4 Expert Rating Editor ......................................... 80
5. Visualization ....................................................... 82
6. SOWAT output ..................................................... 84
7. SOWATconfig.ini ................................................. 86
8. SOWAT tool usage ............................................... 87
9. Example data ...................................................... 88

1. Introduction

The SOWAT tool has been designed to process experimental data and to visualize operator taskload, together with other measurements of interest. With the SOWAT tool, interesting areas in an operator's taskload can be determined.
For operation of the SOWAT tool, familiarity with the Observer software, the CTL model and the other used measures is required.

2. System requirements

- Windows XP/2000
- Version 1.1 or higher of the .NET framework
- Screen resolution of at least 1024 x 768.

The SOWAT tool was tested on an AMD XP 1800+, running at 1.5 GHz, with 256 MB of RAM.
3. Installation

To install the SOWAT tool on your computer, simply execute the *SOWATSetup.msi* file. This will start up the installation program, guiding you through the installation process.

To uninstall the SOWAT tool, navigate to Add/Remove programs in the Windows control panel. Locate the SOWAT tool in the list of installed programs and press the uninstall button.

4. Data collection

The SOWAT tool is able to work with information from a variety of data sources. This section will cover the characteristics and assumptions about these data sources and it will explain the usage of the Data collection interface of the SOWAT tool. Furthermore the usage of the Task Model Editor and the Expert Rating Editor is explained.

4.1 Data sources and characteristics

This section describes the characteristics of the different input files for the SOWAT tool. Examples of all these files are included with the SOWAT software (see chapter 9).

4.1.1 Observer logs

Compatibility:
The SOWAT tool is able to work with exports from the Observer XT software, version 6.0. The SOWAT tool is not compatible with older versions of the Observer software. The compatibility of the SOWAT tool with future versions of the Observer depends totally on the changes made to the exportable logs in future Observer versions.

Assumptions:
For synchronization of the different input sources and for correct processing of the Observer logs, the SOWAT tool makes the following assumptions about the characteristics of the exported Observer logs:
- *The logs are exported with starting and stopping states included*. By default, the Observer only exports the starting states, so make sure that the checkbox for including the stopping states is checked when exporting from the Observer.
- *The first entry in the log indicates the start of the scenario*. The easiest way to do so, is to create a “Scenario” subject in the Observer and score a scenario start state. This is shown in the Observer log example below.

Example:
Here we list an example of a (part of an) Observer log file that can be processed by the SOWAT tool. An example Observer log file is included with the SOWAT tool (see chapter 9 of this manual).
### 4.1.2 ADIOS logs

#### Compatibility:
The SOWAT tool is able to work with log files from the ADIOS software, version May 2006. The SOWAT tool is not compatible with older versions of the ADIOS software. The compatibility of the SOWAT tool with future versions of the ADIOS software depends totally on the changes made to generated logs in future ADIOS versions.

#### Assumptions:
For synchronization of the different input sources and for correct processing of the ADIOS logs, the SOWAT tool makes the following assumptions about the characteristics of the ADIOS logs:

- The first entry in the log is the start of the scenario. This is true by design, since the first entry in the ADIOS log files is generated when the software is started, e.g. when the scenario begins. If for some reason this property does not hold (for example in future versions of the ADIOS software), delete rows from the ADIOS logs until the first row corresponds to the start of the scenario.

#### Example:
Here we list an example of a (part of an) ADIOS log file that can be processed by the SOWAT tool (some parts were truncated for readability). An example ADIOS log is included with the SOWAT tool (see chapter 9 of this manual).

```plaintext
14:48:47: Started on 02-06-2006
14:48:47: subscribing UserRole
14:48:47: subscribing AlarmID
14:48:47: subscribing Text
14:48:47: subscribing ProcedureList
14:48:47: subscribing AlarmType
14:48:47: subscribing Priority
14:48:47: publishing Checked
... ...
...
```
4.1.3 Task model

Compatibility:
For the XML task models, we defined a DTD (Document Type Definition). This DTD formally specifies the structure of an XML document. Task models designed according to this DTD are compatible with the SOWAT tool. The XML Task Model DTD is defined as follows:

```xml
<!DOCTYPE tasklist [ 
  <!ELEMENT tasklist (task*)> 
  <!ELEMENT task (task_name, action*)> 
  <!ELEMENT task_name (#PCDATA)> 
  <!ELEMENT action (ObserverDescription, ADIOSDescription, LIP_levels)> 
  <!ELEMENT ObserverDescription (#PCDATA)> 
  <!ELEMENT ADIOSDescription (#PCDATA)> 
  <!ELEMENT LIP_levels (beginner, intermediate, advanced)> 
  <!ELEMENT beginner (#PCDATA)> 
  <!ELEMENT intermediate (#PCDATA)> 
  <!ELEMENT advanced (#PCDATA)> 
  <!ATTLIST task id CDATA #REQUIRED> 
]> 
```

For more information about DTDs, refer to [http://www.w3schools.com/dtd/default.asp](http://www.w3schools.com/dtd/default.asp).

To guarantee that a task model respects the DTD, it is advised to use the Task Model Authoring tool provided with the SOWAT tool. This tool will be discussed in section 4.3 of this manual.

Remarks:
- *Missing LIP values.* If LIP values are omitted (or misspelled) for some or all of the actions in the task model, the SOWAT tool will use the default LIP as defined in the SOWATConfig.ini file for these actions. If no SOWATConfig.ini file exists, the default LIP is defined as 3.
- **Missing actions.** If the SOWAT tool encounters an action that is not present in the task model, the default LIP as defined in the SOWATconfig.ini file is used. If no SOWATconfig.ini file exists, the default LIP is defined as 3.

To avoid unwanted behavior in the calculations, make sure that all actions and LIP values are entered in the task model. For more information about the SOWATconfig.ini file, refer to chapter 7 of this manual.

**Example:**
Here we list an example of a (part of an) XML task model (some parts were truncated for readability). An example task model file is included with the SOWAT tool (see chapter 9 of this manual).

```xml
<?xml version="1.0" encoding="utf-8" ?>
<tasklist>
  <task id="0">
    <task_name>Fire 4-65</task_name>
    <action>
      <ObserverDescription>ventilatie crashstop 4-65</ObserverDescription>
      <ADIOSDescription>Fire 4-65 ventilatie crashstop</ADIOSDescription>
      <LIP_levels>
        <beginner>1</beginner>
        <intermediate>1</intermediate>
        <advanced>1</advanced>
      </LIP_levels>
    </action>
    ...
  </task>
</tasklist>
```

**4.1.4 Workload Watch logs**

**Compatibility:**
The SOWAT tool is able to work with log files from the Workload Watch software, version January 2006. The SOWAT tool is not compatible with older versions of the Workload Watch software. The compatibility of the SOWAT tool with future versions of the Workload Watch software depends totally on the changes made to generated logs in future Workload Watch versions.

**Assumptions:**
For synchronization of the different input sources and for correct processing of the Workload Watch logs, the SOWAT tool makes the following assumptions about the characteristics of the Workload Watch logs:
- The first entry in the log is the start of the scenario. In the experiments, it is mandatory that the Workload Watch software is started at the same time the scenario starts.
- Separate logs caused by the crashing of the Workload Watch software. Unfortunately, the Workload Watch software occasionally crashes. This will result in multiple log files for a single session. These log files should be merged together chronologically into a single file. This file can be used as input for the SOWAT tool.

**Example:**
Here we list an example of a (part of an) ADIOS log file that can be processed by the SOWAT tool (two lines shown here are actually one single line in the log files). An example Workload Watch file is included with the SOWAT tool (see chapter 9 of this manual).

```
9-2-2006 20:52:48 | Hoe complex vond u het werk de afgelopen minuut? | niet complex | 2428| Hoe inspannend was het werk de afgelopen minuut? | niet inspannend | 610| Continue next trial
9-2-2006 20:53:50 | Hoe complex vond u het werk de afgelopen minuut? | niet complex | 2114| Hoe inspannend was het werk de afgelopen minuut? | niet inspannend | 168| Continue next trial
9-2-2006 20:54:58 | Hoe complex vond u het werk de afgelopen minuut? | nauwelijks complex | 6879| Hoe inspannend was het werk de afgelopen minuut? | nauwelijks inspannend | 162| Continue next trial
```

### 4.1.5 Expert ratings

**Compatibility:**
For the XML expert ratings, we defined a DTD (Document Type Definition). This DTD formally specifies the structure of an XML document. Expert ratings designed according to this DTD are compatible with the SOWAT tool. The XML Expert rating DTD is defined as follows:

```xml
<!DOCTYPE tasklist [ 
  <!ELEMENT tasklist (task*)> 
  <!ELEMENT task (task_name, action*)> 
  <!ELEMENT task_name (#PCDATA)> 
  <!ELEMENT action (ObserverDescription, ADIOSDescription, LIP_levels)> 
  <!ELEMENT ObserverDescription (#PCDATA)> 
  <!ELEMENT ADIOSDescription (#PCDATA)> 
  <!ELEMENT expertrating (#PCDATA)> 
]>
```

For more information about DTDs, refer to [http://www.w3schools.com/dtd/default.asp](http://www.w3schools.com/dtd/default.asp).
To guarantee that an expert rating respects the DTD, it is advised to use the Expert Rating Authoring tool provided with the SOWAT tool. This tool will be discussed in section 4.4 of this manual.

**Remarks:**
- Actions with no expert ratings. If it is desired not to give an expert rating for specific actions (for example communication), the expert rating for these actions should be set to -1. Actions with an expert rating of -1 will be ignored in the expert rating calculations for the CTL points.
- **Missing actions.** If the SOWAT tool encounters an action that is not present in the expert rating file, the expert rating for that period will be set to -1.
- **Missing expert ratings:** If an expert rating for a specific action is omitted or misspelled, the expert rating for that action will be set to -1.

To avoid unwanted behavior in the calculations, make sure that all actions and expert rating values are entered in the expert rating file.

**Example:**
Here we list an example of a (part of an) XML expert rating file (some parts were truncated for readability). An example expert rating file is included with the SOWAT tool (see chapter 9 of this manual).

```xml
<xml version="1.0" encoding="utf-8" >
<tasklist>
  <task id="0">
    <task_name>Fire 4-65</task_name>
    <action>
      <ObserverDescription>ventilatie crashstop 4-65</ObserverDescription>
      <ADIOSDescription>Fire 4-65 ventilatie crashstop</ADIOSDescription>
      <expertrating>0</expertrating>
    </action>
    ...
  </action>
  ...
</tasklist>
```

### 4.1.6 Mobi files

**Compatibility:**
The SOWAT tool is able to work with Mobi log files similar to the one listed in the example below. An example Mobi file is included with the SOWAT tool.

**Assumptions:**
- *The pulse is used to indicate the start of the scenario.* The Mobi hardware has a pulse button. If this pulse button is pressed, this is noted in the recorded data. The SOWAT tool assumes that the first encountered pulse signal in the log file indicates the start of the scenario. If this is not the case, lines should be removed from the start of the log file to ensure that the first encountered pulse corresponds with the start of the scenario.

**Example:**
Here we list an example of a (part of a) Mobi log file. An example Mobi log file is included with the SOWAT tool (see chapter 9 of this manual).
Output data file:
/m0/E:\TNO\OLA\TEAM1/KPL1S1.txt
0  0   2002   -1774  2047
250  0   -1676  -154  2047
500  5964  -1509   -84  -768
750  5964   -1307   -7  -2047
1000  5511  -1368  -34  -2047
1250  5504  -1463  -96  -2047
1500  5504  -1539  -149  -2047
1750  5504  -1579  -181  -2047
2000  5350  -1599  -196  -2047
2250  5258  -1631  -206  -2047
2500  5258  -1693  -219  -2047
2750  5258  -1744  -227  -2047
3000  5312  -1757  -241  -2047
3250  6122  -1783  -237  -2047
3500  6122  -1802  -180  -2047
...
...

4.1.7 Tobii files

Compatibility:
The SOWAT tool is able to work with Tobii log files similar to the one listed in the example below. An example Tobii file is included with the SOWAT tool (see chapter 9 of this manual).

Assumptions:
- The start of the Tobii log is the start of the scenario. The Tobii device can be started with a script executed by the Observer on the start of an observation. This ensures that the Tobii log is synchronized with the Observer. The SOWAT tool assumes that this method is used.

Remarks:
- Missing values. The Tobii device can not always accurately measure the pupil size. The accuracy of every measure is indicated by a validity code in the logfile. The SOWAT tool disregards measures with a validity code of 2 and higher, as suggested in the Tobii manual. Missing pupil values are recorded as -1.

Example:
Here we list an example of a (part of a) Tobii log file (two lines shown here are actually one single line in the log file). An example Tobii log file is included with the SOWAT tool (see chapter 9 of this manual).
4.2 The Data Collection interface

To load all input files into the SOWAT tool for analysis, the Data Collection Interface is used. The data collection interface is shown in Figure 1.

![Figure 1: The data collection interface of the SOWAT tool](image)

In the upper left corner, the tab pages (region 1, figure 1) can be used to navigate between the Data Collection and Visualization sections of the SOWAT tool. On the left side of the data collection interface, there are seven buttons with a corresponding text field (region 2, figure 1). Each of these buttons can be used to load one of the different input sources. The corresponding text field next to each button gives...
information about the (possibly) loaded input files. They will display the filename of the loaded file or an error message if an incorrect file was loaded. A button to unload all the loaded input files is located at the bottom of region 2. Please note that an Observer log and an ADIOS log can never be loaded at the same time. Loading an ADIOS log when an Observer log has already been loaded will result in unloading the Observer log file and vice versa.

To the right of the input buttons, there is a section with tick boxes that indicate which of the input files have successfully been loaded (region 3, figure 1). Once enough input sources are loaded for calculation to be possible, the “Process input”-button in the upper right corner becomes active (region 6, figure 1). For processing to be possible, at least a task model and either an Observer log file or an ADIOS log file need to be loaded.

Once the “Process input” button is clicked, the loaded log files are used to generate the internal operator state log. The button “Export to XML” (region 5, figure 1) will then become active and upon clicking it, a save dialog will be displayed for saving the operator state log to an XML file. After pressing the “Process input”-button, there is data available for visualization. This is indicated by the “Plotting data available” tick box (region 4, figure 1). By clicking the visualization tab (region 1, figure 1), the user navigates to the visualization section. The visualization section of the SOWAT tool is described in chapter 5 of this manual.

4.3 Task Model Editor

For the creation of task models, the Task Model Editor is included with the SOWAT software. The Task Model Editor provides a graphical environment for the authoring of task models. The editor makes the creation of task models easier and moreover, it assures that the required structure of the task models is always respected.

A screenshot of the Task Model Editor is shown in figure 2. The left panel shows the currently loaded task model as a tree. Adding and removing nodes and changing the value of some of the nodes can be done with the buttons to the right of the tree view. Which buttons are active, depends on the currently selected node in the tree.

- If the root node of the tree (named “tasklist”) is selected, the “Add task” button is active. Pressing this button will add a new task to the tree.
- If a task node is selected, the “Add Action” button is active. Pressing this button will add a new action to the currently selected task.
- If one of the value nodes is selected (value nodes are colored green), the “Edit text” button is active. Pressing this button will enable the user to enter a value for the selected value node. Pressing Alt-e when a value node is selected has the same effect and allows for quicker editing.
- If a task node or action node is selected, the “Delete node” button is active. Pressing this button will delete the currently selected node and all its sub nodes.
- The “Print tree” button is always active. It opens a printer dialog, enabling the user to print the entire tree. This gives an easier overview of the structure of the tree.
The user has several options using the “File” menu. The “File” menu is shown in figure 3.

- **New tasklog** will create a new empty tasklog. The current tasklog will be discarded.
- **Open existing Tasklog** opens an existing tasklog for editing.
- **Save current Tasklog** will save the current tasklog to a file for usage with the SOWAT tool.
- **Parse Observerlog** will open an Observer logfile. All actions occurring in that logfile will be displayed in the right panel of the Task Model Editor interface.
- **Parse ADIOS log** will open an ADIOS logfile. All actions occurring in that logfile will be displayed in the right panel of the Task Model Editor interface.
- **Exit** closes the Task Model Editor.
For the SOWAT tool to correctly recognize and process the actions it encounters in the used Observer and/or ADIOS logfiles, it is necessary that the names of these actions in the task model match the names of these actions in the logfiles. The easiest way to ensure this is to use the Parse Observerlog and Parse ADIOSlog functions from the menu as described above. The actions occurring in these logs will be shown in the right panel. These names can then be copy/pasted into their correct locations in the task model.
For more information about the structure of the task models, please refer to section 4.1.3 of this manual.

4.4 Expert Rating Editor

For the creation of expert rating files, the Expert Rating Editor is included with the SOWAT software. The Expert Rating Editor is very similar to the Task Model Editor. The Expert Rating Editor provides a graphical environment for the authoring of expert ratings. The editor makes the creation of expert ratings easier and moreover it assures that the required structure of the expert ratings is always respected. A screenshot of the Expert Rating Editor is shown in figure 4. The left panel shows the currently loaded expert rating model as a tree.

Adding and removing nodes and changing the value of some of the nodes can be done with the buttons to the right of the tree view. Which buttons are active, depends on the currently selected node in the tree.
- If the root node of the tree (named “tasklist”) is selected, the “Add task” button is active. Pressing this button will add a new task to the tree.
- If a task node is selected, the “Add Action” button is active. Pressing this button will add a new action to the currently selected task.
- If one of the value nodes is selected (value nodes are colored green), the “Edit text” button is active. Pressing this button will enable the user to enter a value for the selected value node. Pressing Alt-e when a value node is selected has the same effect and allows for quicker editing.
- If a task node or action node is selected, the “Delete node” button is active. Pressing this button will delete the currently selected node and all its sub nodes.
- The “Print tree” button is always active. It opens a printer dialog, enabling the user to print the entire tree. This gives an easier overview of the structure of the tree.
The user has several options using the “File” menu. The “File” menu is shown in figure 5.

- **New Expertrating** will create a new empty expert rating. The current expert rating will be discarded.
- **Open existing Expertrating** opens an existing expert rating for editing.
- **Save current Expertrating** will save the current expert rating to a file for usage with the SOWAT tool.
- **Import from Taskmodel** will load an existing taskmodel and create a new expert rating with the same structure as the taskmodel.
- **Parse Observerlog** will open an Observer logfile. All actions occurring in that logfile will be displayed in the right panel of the Expert Rating Editor interface.
- Parse ADIOS log will open an ADIOS logfile. All actions occurring in that logfile will be displayed in the right panel of the Expert Rating Editor interface.
- Exit closes the Expert Rating Editor.

For the SOWAT tool to correctly recognize and process the expert ratings for the actions it encounters in the used Observer and/or ADIOS logfiles, it is necessary that the names of these actions in the expert rating match the names of these actions in the logfiles. The easiest way to do this is to first create a task model using the Task Model Editor described in section 4.4. Then, using the Import from Taskmodel function from the menu, a new expert rating is created with the same structure as the loaded task model. This way only the ratings have to be filled in.

Another way to do this is to use the Parse Observerlog and Parse ADIOSlog functions from the menu as described above. The actions occurring in these logs will be shown in the right panel. These names can then be copy/pasted into their correct locations in the expert rating.

For more information about the structure of the expert ratings, please refer to section 4.1.5 of this manual.

5. Visualization

After all the necessary input sources have been loaded into the SOWAT tool with the Data Collection interface described in section 4.2, the data can be visualized using the Visualization interface. The visualization interface is shown in figure 6.
In the upper left corner of the visualization interface, the data selection section is located (region 1, figure 6). It shows the length of the recorded data and it enables the user to make a selection from that period for visualization. The user has to select the following parameters:
- The starting time \((st)\) for the plotting period
- The ending time \((et)\) for the plotting period.
- The amount of CTL points to represent the selected period
- The skill level for the current operator (user model)
- The measurement to use for LIP calculation
- Whether to draw a line through the plotted points or not (can be altered after plotting)

The user has 3 options for the LIP calculations, as shown in figure 7. If no Workload Watch (WLW) log has been loaded, only the LIP contained in the task model can be used and the other options will be disabled. When a Workload Watch log has been loaded, the LIP can also be calculated using the subjective complexity values contained in it. Finally, the user can also select to use the average of the task model values and the subjective values.

![Figure 7: Selecting which source to use for the LIP calculation](image1)

When using the task model (or the average) for calculating the LIP, if the SOWAT tool encounters an action that is not present in the task model, it will use the default LIP for that (sub)task. The default LIP is specified in the SOWATconfig.ini file. More information about this file can be found in chapter 7 of this manual. After the plotting, a message will be displayed, showing all switches to non-existing actions, as shown in figure 8. This is an indication that the task model should be revised.

![Figure 8: A switch to a non existing (sub)task has occurred](image2)

Once the desired data selection options have been provided, the “Plot”-button can be clicked. This will result in plots of the data in the presentation area (region 2, figure 7). First a 3D CTL cube is plotted. Around the cube, the projections of the three sides of the cube are plotted for easier reading. The 3D cube can be rotated by pressing the left
mouse button in the cube and then moving the mouse while holding the left mouse button.

On the right side of the presentation area, additional plots displaying all recorded variables set out against the points are shown. The user can switch between the available plots using the tab pages. The tab pages available depend on the different input sources that have been loaded. The possible available tab pages are:

- CTL1: TO and LIP
- CTL2: TSS
- WLW: Subjective effort and Subjective complexity
- Expert: Expert rating
- Heart: Heart rate
- Resp: Upper and lower respiratory values
- Eye: Pupil size

In the lower left corner, a table is displayed containing the values of all variables for each cube point (region 3, figure 7). Selecting a row in this table will highlight the corresponding points in the plots.

After data has been plotted, there are several options to manipulate the plots. In left part of the data manipulation section (region 4, figure 7), the user can select how the points in the plots are colored. By default, the points are plotted with decreasing opacity. If a workload watch file was loaded, the option to color the points according to the subjective effort rating becomes available. This will color the points in all plots in three colors: green (low effort), orange (medium effort) and red (high effort). The thresholds for the coloring of the points with subjective effort can be set using the controls in the middle of the data manipulation section.

If an expert rating file was loaded, the option to color the points according to the expert rating becomes available. This will color the points in all plots in three colors: green (high expert rating), orange (medium expert rating) and red (low expert rating). The thresholds for the coloring of the points with the expert rating can be set using the controls in the right part of the data manipulation section.

When the user is satisfied with the selected data, he can press the “Save statistics” button (region 5, figure 7). This will open a dialog to save statistics about the current analysis to a text file. If the user presses the “Save statistics and graphs” button, the statistics file will also be saved along with jpg-images of all plots. The output of the SOWAT tool is described in chapter 6 of this manual.

All plots can be saved manually by pressing the small save button in the lower right corner of all plots.

6. SOWAT output

After the visualization of the data using the SOWAT tool, statistics about the used data can be saved to a text file. The amount of information contained in this file depends on which input sources have been loaded. A sample output of the SOWAT tool is shown below:

SOWAT statfile
Created at: 7/6/2006 4:25:12 PM
---

Length of recording: 00:17:19
[Processed Files]
Observer File: F:\Adiosdata\adios 2 tm 30 (8).txt
Task Model File: F:\Adiosdata\Adios1.xml
Workload Watch File: F:\Adiosdata\memorycard\Results Duijts op 09-02 08.59.txt
Expert Rating File: F:\Adiosdata\Expert ratings\Adios22_Duijts_Jepkes.XML
Mobi File: F:\Physiology\Mobi_1.txt
Tobii File: F:\Physiology\Tobii_1.txt

[TSS]
Total: 79
Average per minute: 4.56

[LIP]
LIP source: Task model
Average: 2.73

[TO]
Total: 82%

[Subjective Ratings]
Average subjective effort: 2.16
Average subjective complexity: 1.98

[Expert Ratings]
Average expert rating: 0.31

[Mobi Data]
Average Heart Rate: 69.86

[Tobii Data]
Average Pupil size Right: 4.30
Average Pupil size Left: 4.44

[Visualized Data]
Start time: 5:00
End time: 10:00
Skill level: intermediate
LIP source: Task model
Point Coloring: Gradient

<table>
<thead>
<tr>
<th></th>
<th>LIP</th>
<th>TSS</th>
<th>TO</th>
<th>Effort</th>
<th>Compl.</th>
<th>Expert</th>
<th>Heart</th>
<th>RespUp</th>
<th>RespLo</th>
<th>PupR</th>
<th>PupL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.92</td>
<td>8.00</td>
<td>100%</td>
<td>2.34</td>
<td>2.34</td>
<td>1.00</td>
<td>67.93</td>
<td>-1372.7</td>
<td>-981.4</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>2</td>
<td>3.87</td>
<td>6.00</td>
<td>100%</td>
<td>3.00</td>
<td>3.00</td>
<td>0.00</td>
<td>67.79</td>
<td>-1283.2</td>
<td>-607.5</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>3</td>
<td>4.45</td>
<td>4.00</td>
<td>99%</td>
<td>3.25</td>
<td>3.00</td>
<td>0.63</td>
<td>63.65</td>
<td>-1023.0</td>
<td>-813.6</td>
<td>4.40</td>
<td>4.60</td>
</tr>
<tr>
<td>4</td>
<td>3.58</td>
<td>2.00</td>
<td>98%</td>
<td>4.00</td>
<td>3.00</td>
<td>1.65</td>
<td>79.35</td>
<td>-778.9</td>
<td>-956.1</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>5</td>
<td>3.74</td>
<td>6.00</td>
<td>100%</td>
<td>3.95</td>
<td>3.00</td>
<td>0.37</td>
<td>70.08</td>
<td>-1462.5</td>
<td>-617.4</td>
<td>4.63</td>
<td>5.67</td>
</tr>
<tr>
<td>6</td>
<td>2.51</td>
<td>10.00</td>
<td>98%</td>
<td>3.00</td>
<td>3.00</td>
<td>0.27</td>
<td>69.63</td>
<td>-1439.9</td>
<td>-866.2</td>
<td>4.40</td>
<td>4.17</td>
</tr>
<tr>
<td>7</td>
<td>3.00</td>
<td>2.00</td>
<td>100%</td>
<td>3.00</td>
<td>3.00</td>
<td>0.00</td>
<td>67.90</td>
<td>-1709.5</td>
<td>-967.2</td>
<td>5.39</td>
<td>5.65</td>
</tr>
<tr>
<td>8</td>
<td>2.68</td>
<td>8.00</td>
<td>88%</td>
<td>3.83</td>
<td>3.83</td>
<td>0.21</td>
<td>72.98</td>
<td>-1545.5</td>
<td>-1027.5</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>9</td>
<td>3.75</td>
<td>4.00</td>
<td>100%</td>
<td>4.00</td>
<td>4.00</td>
<td>0.00</td>
<td>70.45</td>
<td>-1358.8</td>
<td>-999.5</td>
<td>-1.00</td>
<td>-1.00</td>
</tr>
<tr>
<td>10</td>
<td>3.33</td>
<td>10.00</td>
<td>99%</td>
<td>4.74</td>
<td>4.74</td>
<td>0.00</td>
<td>75.31</td>
<td>-1603.0</td>
<td>-905.5</td>
<td>-1.00</td>
<td>4.46</td>
</tr>
</tbody>
</table>

[END]
The header displays the creation date of the statistics file and the total length of the recording, as shown in the top of the data selection section of the visualization interface.

Processed files show which files were used as input for this analysis.

TSS shows the total amount of task switches and the average task switches per minute over the entire scenario.

LIP shows what source was used to calculate the LIP values and the average LIP value over the entire scenario.

TO shows the percentage time occupied over the entire scenario.

Subjective ratings show the average subjective effort and average subjective complexity over the entire scenario.

Expert ratings show the average expert rating over the scenario (not counting periods with an expert rating of -1).

Mobi data shows the average heart rate over the entire scenario.

Tobii data shows the average pupil size for both eyes over the entire scenario.

Visualized data shows the settings used for the visualization as selected in the visualization interface:

- Start time
- End time
- Operator skill level: beginner, intermediate or advanced
- Source for LIP values: Task model, Workload Watch or average
- Coloring of points: Gradient, Subjective Effort or Expert ratings
- Thresholds for the coloring of the points (if applicable)

Visualization images shows the relative path to the exported images of the plots (if any).

Error messages show the encountered errors during processing (if any).

7. SOWATconfig.ini

To set the values of some of the parameters used within the SOWAT tool, the SOWATconfig.ini file can be used. This is a simple text file that can be edited with any text editor, e.g. Windows Notepad. The file has the following structure:

defaultLIP=3
maxExpert=2.0
minExpert=0.0
greenExpert=1.5
redExpert=0.5
maxSubjectiveEffort=5.0
minSubjectiveEffort=0.0
redSubjective=4.0
greenSubjective=2.0

- defaultLIP sets the LIP value that the SOWAT tool will use, if it encounters an action that is not present in the task model. If an action is present in the task model, but the corresponding LIP is omitted or misspelled, this value will also be used as LIP for that action.
- maxExpert and minExpert set the maximum and minimum values for the selectors for the expert rating coloring thresholds in the Visualization interface.
greenExpert and redExpert set the default values for the selectors for the expert rating coloring thresholds in the Visualization interface.

maxSubjectiveEffort and minSubjectiveEffort set the maximum and minimum values for the selectors for the subjective effort coloring thresholds in the Visualization interface.

redSubjective and greenSubjective set the default values for the selectors for the subjective effort coloring thresholds in the Visualization interface.

The SOWATconfig.ini file is located in the program directory of the SOWAT tool. If the SOWATconfig.ini file is not present, the values as listed in the example above will be used.

8. SOWAT tool usage

This section describes a general procedure for analysis of experimental data, using the SOWAT tool.

Step 1: Observer scoring
Generally, the Observer is used as basis of the SOWAT analysis. The subjects are filmed during the experiments; the experimenter now has to use the Observer software to score the actions of the subjects as accurately as possible. Once this is done, the Observer analysis has to be exported to a file.
If only ADIOS logs are used for analysis, this step can be skipped.

Step 2: Creating the task model
Next, the task model has to be created. The experimenter will already have a list of tasks that can possibly occur and their decomposition into subtasks, since this is the basis of the experiment itself. This information has to be entered into the task model. The easiest way to do this is to parse an Observer (and/or ADIOS) log file in the Task model authoring tool and to create a task model tree from it. This ensures that the tasks are named correctly in accordance with their occurrence in the Observer (and/or ADIOS) log files.
After the task model structure is complete, the LIP values of each action have to be filled in. If the group of test subjects is homogenous, then only one LIP value has to be filled in for each action. If the skill differs between the test subjects, then up to 3 different LIP values can be added to the task model for each action.

Step 3: Testing the task model
Before moving on, it is advisable to test if the task model contains all possible actions occurring in the Observer (and/or ADIOS) log files. To do this, the task model and an Observer (and/or ADIOS) log file should be loaded into the SOWAT tool and processed. In the visualization section, a visualization of the entire period should be selected. If a nonexistent action is encountered, the SOWAT tool will give an error, indicating the name of the action. The task model can then be revised to also incorporate the missing actions. Repeat this procedure for different Observer (and/or ADIOS) log files.
Step 4: Creating expert rating files
Once the task model has been verified, it can be used to create expert ratings. Load the task model into the expert rating authoring tool, to generate an expert rating tree. The experimenter now has to fill in the expert ratings for each action.

Step 5: Analysis
Now all input sources are available and they can be processed by the SOWAT tool. Load all relevant input sources for the subject under analysis and process them. Manipulate the visualizations to investigate periods of interest and to color the graphs according to the performance and effort measures. This is best done while also watching the video footage through the Observer software (if available). This way it is clearer what is happening in the selected period.

9. Example data

In the program directory of the SOWAT tool, there is a subdirectory called “Example Data”. This subdirectory contains some example input files for the SOWAT tool. Loading this data into the SOWAT tool gives an opportunity to get familiar with the features of the SOWAT tool. Furthermore, these example input files can be used to determine if other input files are suitable for the SOWAT tool, by comparing their structure to the example files. The “Example Data” folder contains 2 subfolders: “Observerlog example” and “ADIOSlog example”. All files in each of the folders can be loaded into the SOWAT tool together.