Chapter in Handbook of Cognitive Task Design, Erik Hollnagel (ed.)
To be published by Lawrence Erlbaum Associates in the fall of 2003
(*this is not the final version!*)
Reference: Neerincx, M.A. (in press). Cognitive task load design: model, methods and examples. In: E. Hollnagel (ed.), *Handbook of Cognitive Task Design.* Chapter 13 (pp. 283-305). Mahwah, NJ: Lawrence Erlbaum Associates.

Title: **Cognitive task load analysis: allocating tasks and designing support**

Author: Mark A. Neerincx
TNO Human Factors
P.O. Box 23
3769 ZG Soesterberg
The Netherlands
Phone: +31 346 35 62 98
Fax: +31 346 35 39 77
Email: neerincx@tm.tno.nl

Table of contents

Abstract ... 1
Introduction ... 2
Cognitive task load ... 4
  Task demands ... 4
  Three dimensional "load space" ... 5
  Analysis method ... 8
Analysis of task allocations: An example ... 9
  Specification ... 10
  Assessment ... 11
  Conclusions ... 12
High-demand situations and cognitive support ... 13
  Time occupied ... 13
  Level of information processing ... 13
  Task-set switching ... 16
Design of cognitive support: An example ... 17
  User interface design ... 18
  Task Level ... 18
  Communication Level ... 24
  Evaluation ... 28
Discussion ... 29
Conclusions ... 30
References ... 31
Acronyms ... 33

Cognitive Task Load Analysis: Allocating Tasks and Designing Support

MARK A. NEERINCX

TNO Human Factors
P.O. Box 23, 3769 ZG, Soesterberg, The Netherlands

Abstract

We present a method for Cognitive Task Analysis that guides the early stages of software development, aiming at an optimal cognitive load for operators of process control systems. The method is based on a practical theory of cognitive task load and support. In addition to the classical measure percentage time occupied, this theory distinguishes two load factors that affect cognitive task performance and mental effort: the level of information processing and the number of task-set switches. Recent experiments provided empirical support for the theory, showing effects of each load factor on performance and mental effort. The model can be used to establish task (re-)allocations and to design cognitive support.

This chapter provides an overview of the method's foundation and two example applications. The first example is an analysis of the cognitive task load for a future naval ship control centre that identified overload risks for envisioned operator activities. Cognitive load was defined in terms of task demands in such a way that recommendations could be formulated for improving and refining the task allocation and user-interface support. The second example consists of the design and evaluation of a prototype user interface providing support functions for handling high-demand situations: an information handler, a rule provider, a diagnosis guide and a task scheduler. Corresponding to the theory, these functions prove to be effective, in particular when cognitive task load is high. The user interface is currently being implemented at a bridge of an icebreaker.

The two examples comprise an integrated approach on task allocation enhancement and design of cognitive support. The theory and method are being further developed in an iterative cognitive engineering framework to refine the load and support model, improve the empirical foundation and extend the examples of good practices.

Introduction

In different work domains, such as defence, public safety, process control and transport, the need to improve the deployment of human knowledge and capacities is increasing. Whereas safety requirements increase (due to both company and general public policies), less personnel should be able to do the concerning tasks by concentration of work in one central control room and further automation. For example, fewer personnel will have to manage high-demand situations and supervise complex automated systems in future ships of the Royal Netherlands Navy. Tasks that were allocated to separate jobs are currently being combined into new, enriched jobs (e.g. adding platform supervision to navigation tasks on a ship's bridge). In addition to selection and training, adequate (dynamic) task allocation and computer support can help to realise the required human performance level. This chapter presents a method for refining task allocations and designing support functions that extend human knowledge and capacities.

The proposed method for cognitive task load design is based on recent research in the field of Human-Computer Interaction (HCI) and the notion that you need both a model of the environment and the cognitive processes involved to enhance the design of human-computer work. We can make effective use of theories of cognitive processing if we also have validated theories or descriptions of the world on which cognition operates including the interactions (Green, 1998). When we have an adequate description of the environment in which cognition operates, then a human viewed as a behaving system might prove to be quite simple. Or as Simon (1981) stated: "The apparent complexity of human behaviour over time is largely a reflection of the complexity of the environment in which a person finds himself". However, you still need some cognitive theory, as simple as it may be, to distinguish the environmental components that affect human cognitive task performance. Such a theory should not purely focus on task performance at the micro-level and only be validated with isolated laboratory tasks as common in basic (experimental) psychological research. The validation of the models should incorporate essential interactions with the real-world environment in which the tasks are performed at the level of real-world operational requirements. Such descriptions enable statements on human task performance by accounting adequately for how context and actions are coupled and mutually dependent (cf. Hollnagel, 1998). Unfortunately, there is not one context-independent, comprehensive theory on human cognition that can be applied for a complete and do-able analysis of complex work demands and it will not be present in the near future. For example, detailed specifications of cognitive processes such as the unified theories of Newell (1990) and the multiple-resource model of Wickens (1992) insufficiently address the dynamic task demands on human-problem solving in a naval command centre. The solution is not to wait till such a theory has been developed, but to develop limited or practical theories that apply to the specific domain or environmental description that is part of it (cf. Green, 1990). Such a theory should include accepted features of cognition such as limited processing capacity, be validated in the context of a specific domain and possibly group of task performers, and provide predictions of the task performance within this domain (cf. the Simple Model of Cognition, Hollnagel, 1998).

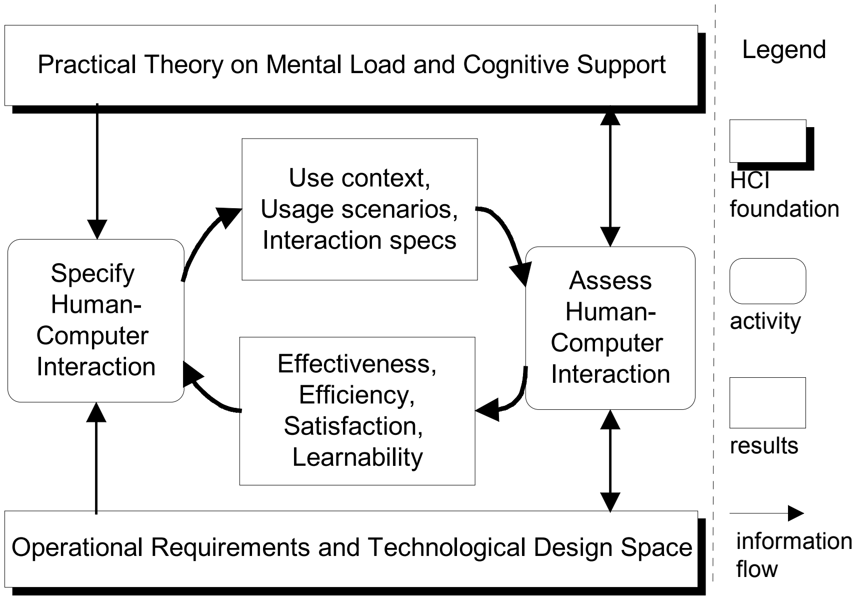

Figure 1: Addressing human-factors knowledge and environmental constraints in an iterative system development process.

This chapter presents a practical theory of cognitive task load and computer support for process-control tasks. The theory has been integrated in a cognitive engineering framework consisting of the specification and assessment of computer-supported work (figure 1). According to this framework, assessments guide the iterative HCI-refinement process (including possible adjustments of operational requirements), *and* provide empirical data for improving the theory and its quantification in a specific application area (e.g., a "mental load standard" for railway traffic control; Neerincx & Griffioen, 1996). First, we will summarise the theory on cognitive task load and exemplify the corresponding method for task-allocation assessment in an early system development stage. Subsequently, we extend the theory with cognitive support concepts and show that the theory and method can also be applied for user-interface design. For explanatory objectives, task allocation and user-interface design are dealt with separately. At the end of this chapter, we will elucidate that the two examples comprise *one* practical theory and method to guide an iterative process of human task and interface design.

Cognitive task load

Task demands

Recently, we developed and validated a practical theory for cognitive load and support in process-control tasks (Neerincx et al., 2000; Neerincx & Van Besouw, 2001). The theory includes a model of cognitive task load that comprises the effects of task characteristics on performance and mental effort. According to this model, cognitive task load is a function of the percentage time occupied, the level of information processing and the number of task-set switches. The first classical load factor, *percentage time occupied,* has been used to assess workload in practice for time-line assessments. 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 address the cognitive task demands, our load model incorporates the Skill-Rule-Knowledge framework of Rasmussen (1986) as an indication of the *level of information processing.* At the skill-based level, information is processed automatically resulting into actions that are hardly cognitively demanding. At the rule-based level, input information triggers routine solutions (i.e. procedures with rules of the type 'if <event/state> then <actions>') resulting into efficient problem solving in terms of required cognitive capacities. At the knowledge-based level, based on input information the problem is analysed and solution(s) are planned, in particular to deal with new situations. This type of information processing can involve a heavy load on the limited capacity of working memory.

To address the demands of attention shifts, the model distinguishes *task-set switching* as a third load factor in the performance of process control tasks. Complex task situations consist of several different tasks, with different goals. These tasks appeal to different sources of human knowledge and capacities and refer to different objects in the environment. We use the term task set to denote the human resources *and* environmental objects with the momentary states, which are involved in the task performance. Switching entails a change of applicable task knowledge on the operating and environment level. Figure 2 shows a model for multiple task performance that we will use to specify the goal-directed and situation-driven elements of computer-supported human behaviour. Three abstraction levels of human behaviour are distinguished. First, an *activity* is the combination of tasks and actions that are performed to accomplish a general goal in a definite period and for a specific scenario (e.g. damage control on a cargo ship in stormy weather during night). A scenario consists of an initial state of the ship and environment, and a sequence of events that trigger tasks. Second, *tasks* are performed to accomplish a sub-goal (e.g. restore propulsion engine). The term task set is used to denote the composite of goal, knowledge and processing resources of the operator, and the corresponding objects in the environment. A task is activated by an event (e.g. engine shutdown) and/or a predefined goal of the task executor. Third, *actions* are the elements of observable behaviour that affect the state of a specific object in the environment. The process state determines which action is active or should be activated.

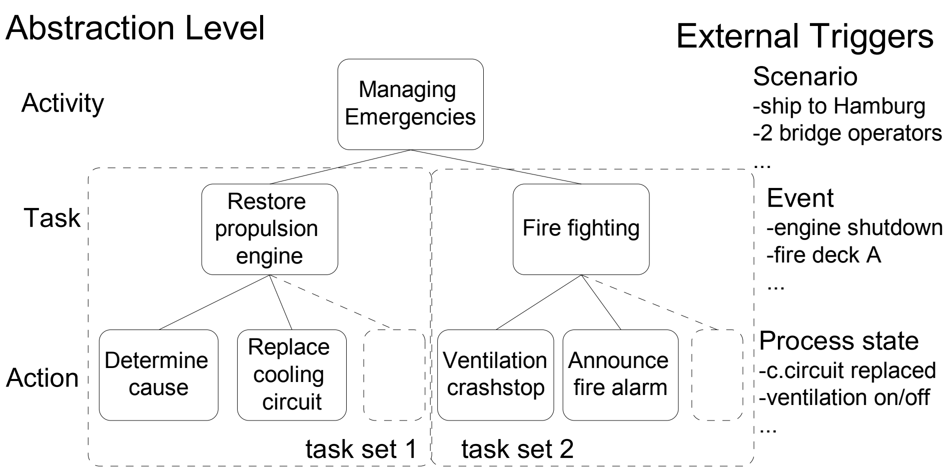

Figure 2: The descriptive framework for multiple task performance by one operator (who has to deal with an engine shutdown and fire on deck A in the example).

Three dimensional "load space"

The combination of the three load factors determines the cognitive task load: the load is high when the percentage time occupied, the level of information processing (i.e. the percentage knowledge-based actions) and the number of task-set switches are high. Figure 3 presents a 3-dimensional "load" space in which human activities can be projected with regions indicating the cognitive demands that the activity imposes on the operator. It should be noted that these factors represent task demands that affect human operator performance and effort (i.e. it is *not* a definition of the operator cognitive state). In practice, operator activities will not cover all possible regions in the cube of figure 3. A higher level of information processing may cause the time occupied to increase. Also a larger amount of task-set switches may cause the time occupied to increase because the costs of these switches are so severe that the operator needs more time to execute the task. The cognitive task load analysis of this chapter aims at a cube that is "empty" for the critical regions such as distinguished below. For remaining critical situations, it aims at empowering the operators so that they can meet the specific demands.

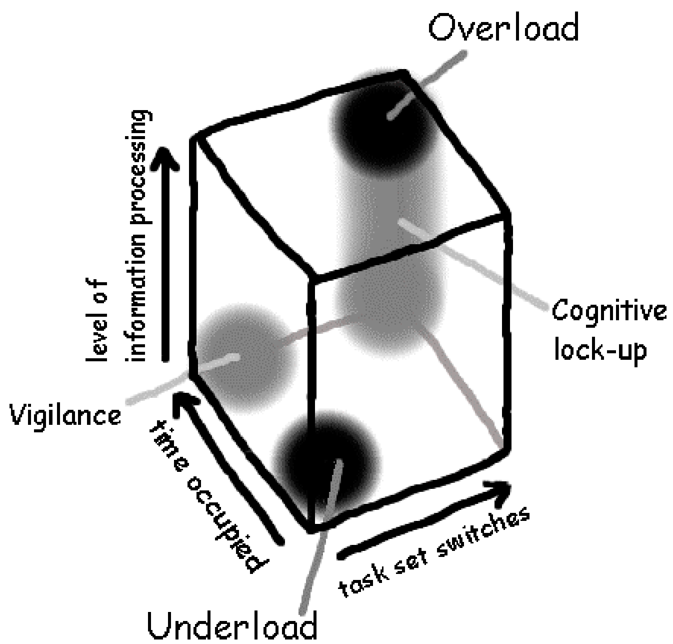

Figure 3: The three dimensional model of cognitive task load with four general problem regions.

Table 1: Overview of four negative effects of cognitive task demands for a certain task period.

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

It should be noted that the effects of cognitive task load depend on the concerning task duration (see table 1). In general, the negative effects of under- and overload increase over time. *Under-load* will only appear after a certain work period, whereas (momentary) *overload* can appear at every moment. When task load remains high for a longer period, carry-over effects can appear reducing the available resources or capacities for the required human information processing (Rouse, 1988). *Vigilance* is a well-known problematic task for operators in which the problems increase in time. Performance decrease can already occur after 10 minutes when an operator has to monitor a process continuously but does not have to act (Levine et al., 1973; Parasuraman, 1986). Vigilance can result in stress due to the specific task demands (i.e. the requirement to continuously pay attention on the task) *and* boredom that appears with highly repetitive, homogeneous stimuli. Consequently, the only viable strategy to reduce stress in vigilance, at present, appears to be giving the freedom to stop when people become bored (Scerbo, 2001). Recent research on *cognitive lock-up* shows that operators have fundamental problems to manage their own tasks adequately. Humans are inclined to focus on one task and are reluctant to switch to another task, even if the second task has a higher priority. They are stuck to their choice to perform a specific task (Boehne & Paese, 2000) and have the tendency to execute tasks sequentially (Kerstholt & Passenier, 2000).

In a sequence of experiments, we validated the 3-dimensional load model. For example, a study in the high-fidelity, "one to one", control centre simulator of a Royal Netherlands Navy frigate showed substantial performance decrease in the problem areas of figure 3 (i.e. for specific load values). In general, empirical research should provide the data to establish the exact boundaries of the critical regions for a specific task domain (e.g. by expert assessments and/or operator performance evaluations).

Analysis method

Neerincx et al. (2000) describe an analysis method for human-computer task performance to establish the task demands in term of the three load factors in figure 3. This method combines specifications of task-set classes with specific task-set instances (i.e., activities). Figure 4 shows the specification process of the goal- and situation-driven task elements with their mutual relations.

A *task decomposition* describes the task classes. It defines the breakdown structure of tasks and provides an overview of the general task objectives assigned to persons and related to detailed descriptions of human actions with the corresponding information needs.

An (hierarchically ordered) *event list* describes the event classes that trigger task classes and provides an overview of the situation-driven elements. An event is referring to a change in an object that can take place in specific situations and has specific consequences.

Scenarios describe sequences of event instances with their consequences on a time line and the initial state of the objects that events act upon. They show a particular combination of events and conditions that cause one line of user actions. Scenarios can be a valuable tool to envision computer-supported work, and to estimate the costs and benefits of the support for the human task performance (Carroll, 1995). In particular, scenarios can provide a "bridge" between the software engineering focus on software's functionality and human factors focus on users' goals and information needs (cf. "use cases" in object-oriented software engineering). In our method scenarios consist of a sequence of events that occur in a specific state. The purpose of this description is not restricted to the general mission of an abstract function, such as damage control. Scenarios are formulated for a large set of action triggering events and, therefore, they can be rather specific.

A set of *Basic Action Sequences* (BAS) describes the general relationships between event classes and task classes as general procedures without situation specifics. Action sequence diagrams define the dynamic or control aspects of task performance, i.e. the events and conditions that trigger task execution sequences. These diagrams are a combination of specifications for time-lines (e.g., Kirwan & Ainsworth, 1992), operational sequences (e.g. Kirwan & Ainsworth, 1992), and co-operation processes (e.g. Neerincx & de Greef, 1998).

A set of *Compound Action Sequences* (CAS) describes the relationships between event instances and task instances for specific situations as activities with the corresponding interface support. Per scenario, the BASs of the events are instantiated and integrated into a CAS that do not contain feedback loops and selection mechanisms (such as if x then y), so that the time-line can be established per chart (for a proposed CAS format, see figure 7).

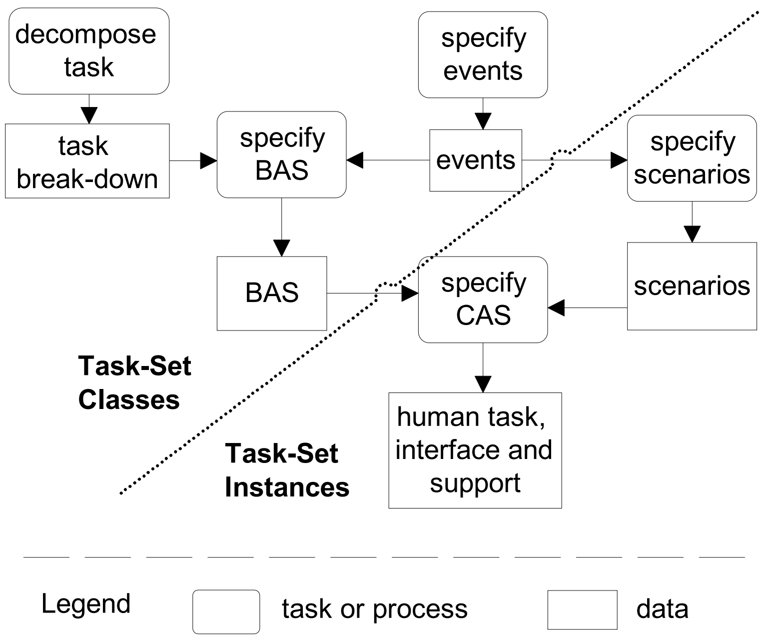

Figure 4: Processes and data flows for the specification of task demands in the analysis stage of user interface development.

Analysis of task allocations: An example

Recently, the Royal Netherlands Navy developed the Integrated Monitoring and Control System (IMCS) for a new ship: the Air Defence and Command Frigate. The IMCS is a large and complex system that is used in diverse situations and work settings for supervision and control of the platform systems and damage control. In the first phase of the IMCS development, the high-level system requirements were provided as Government Furnished Information. This section gives a brief overview of the method for cognitive task analysis that was applied to assess these system requirements (for details, see Neerincx et al., 2000).

Specification

To make the task demands explicit for the IMCS operators, we followed the specification process of figure 4. First, we specified the *task-set classes*. At the top-level, the task decomposition distinguished four platform functions: provide survivability, provide mobility, and support hotel functions, weapons and sensors. In the first instance, the task breakdown stopped when all task allocations to the control-centre crew and IMCS could be designated within it. After establishing the jobs as a set of tasks that have to be performed by one person, decomposition continued until the subtask can be mapped on a specific IMCS support function of the Air Defence and Command Frigate or defined as a 'pure' human action, so that either the human-computer interaction or an observable human action is specified.

Subsequently, we transformed the general task-set specifications into *task-set instances* and "envisioned" about 40 different scenarios. Two critical scenarios that differed from each other fundamentally were selected for further analysis. The first, 'fire in the galley in harbour', consists of an 'unexpected' calamity with extra complications occurring in a quiet situation and a small crew, while the second consists of a very severe damage in war-time, comprising a hectic situation that the complete crew must be able to deal with. The two selected scenarios were transformed into CASs using the BASs for handling the events of this scenario. Navy experts estimated for each action the fastest and slowest performance and the level of information processing according to the Skill-Rule-Knowledge-framework. This resulted in two CASs per scenario: one consisted of the fastest performances and the other the slowest. The actual Action Sequences consisted of a number of actors, the action times were presented in them, and the individual BASs were coded separately to get an overview of the number of switches between task sets in an activity. For scenario 1, the fast CAS lasted about 18 minutes and the slow version about 56 minutes. For scenario 2, the fast CAS lasted about 25 minutes and the slow version about 72 minutes.

Assessment

In the *first step* of the assessment, general patterns and extremes of task load were identified. The percentage time occupied, the percentage knowledge-based actions and the number of task-set switches for each person of the future ship control-centre crew, can be directly derived from the CASs. Differences in the *time* a person is occupied appeared mainly between scenarios. There was a large variance: between 8% to 70% occupied. Time occupied did not appear to be a cause of overload on its own, because it remained below the critical level of 70-80% (Beevis, 1992). Overall, the tasks of managers *and* operators showed a large *knowledge-based* component for which system and process knowledge is required. The work is complex and cognitively demanding for the complete crew. Managers' tasks comprise primarily planning, supervision, priority determination and co-ordination, while the operators have to assess, diagnose and solve specific technical problems. The future situation in the Air Defence and Command Frigate requires that the operators have knowledge of specific parts of the platform control system and tasks they are involved in. In the Cognitive Task Analysis method, a *task set* is defined as a BAS for a specific event. For example, when the same BAS appears more than once in a CAS, they are viewed as different task sets, because they apply to different events (i.e. different objects in the environment). Task-set switching proved to appear rather often. In particular, for manager 1, the number of task-set switching showed to be high in the fast scenario 2: 54 switches in an hour (i.e. a switch every 67 seconds). The number of switches increased in scenario 2 when action times were longer, because the operational requirements were more difficult to satisfy in this condition.

In the *second step* of the assessment, situations of momentary overload were identified. The CASs show the action times of each person and the interrelationships between the actions: the critical path. Often, more than one person is on the critical path, so that sub-optimal performance of one person at a specific moment will often have a major effect upon the overall SSC-crew performance. Therefore, it is of utmost importance to detect possible peak loads for all persons. Compared to the general load, for momentary peak loads the time scale of occurrences of almost continuous knowledge-based actions with a lot of switches is much shorter (between 5 and 15 minutes) and the load limit is higher. For example, the momentary load of operator 2 in the fast condition of scenario 2 proved to be relatively high. In a period of five minutes, he should have to switch every 20-second to a new task set that comprises almost always a knowledge-based action. It can be expected that he will not be able to fulfil these task demands and, because he is on the critical path, this will have an impact on the overall crew performance. Further, in this period the operator performs 19 interactions with the IMCS. For this specific action sequence of operator 2, it is very important that the dialogue with the IMCS is as efficient as possible, i.e., the user interface structure should map very well on this sequence.

Conclusions

The assessment of the four CASs showed possible risks for overload that are mainly caused by the composite of measures on time occupied, percentage of knowledge-based actions and the number of task-set switches. Because cognitive load was described in terms of task and interface characteristics, recommendations could be formulated for task allocation and interface design to diminish these risks. The IMCS specification for the future Air Defence and Command Frigate should be improved by describing a general coherent user interface structure and establishing the dialogue principles for its components. In particular, the interface should enable efficient task-set switching and may even provide support to keep track of task sets that 'run in the background' and to return to these task sets. It should be noted that the IMCS requirements specification defines a number of individual support components from sensor-fusion and filtering mechanisms to damage control advice functions. Each function will probably have a positive effect on the local task performance. However, an overview on the interrelationships and the combined effects of these functions is lacking. For example, a general "high-level" user interface structure is lacking and the management of the support functions can be a load factor in itself (i.e., the control of the envisioned information presentations). To establish support for the managers of the crew, to diminish *peak loads* and to prevent human biases such as *cognitive lock-up*, the combined effects and integration of the support in the overall task performance should be defined explicitly. Human-centred development of interactive systems requires an iterative design process in which cognitive engineers provide the required human factors input in terms of guidelines, user interface concepts, methods and facilities (cf. ISO 13407). Because the IMCS-development had already been started and the development process defined, it was difficult to bring these insights into this process.

High-demand situations and cognitive support

Neerincx & de Greef (1998) propose a model-based design approach that aims at human-computer co-operative problem solving by integrating cognitive support into the user interface. Based on this approach and the load theory (figure 3), we developed a framework that distinguishes a small set of user-interface and cognitive-support functions with specific high-level design principles. Below, we will present the framework and, in the subsequent section, we will provide an example design consisting of these functions.

Time occupied

There is a trade-off between the benefits of support facilities and the interaction costs. In particular when the time pressure is high and the user has only a small part of his or her cognitive resources available to manage and consult such facilities, the benefits should outweigh the costs substantially. The additional time required for interacting with the support facility should be small compared to the execution time for the primary task. We distinguish four general design principles to reduce the interaction load that apply to the *user interface*:

1. *User adaptation*. The user interface design should take account of both the general characteristics of human perception, information transfer, decision-making and control, and the specific user characteristics with respect to education, knowledge, skills and experience.
2. *Goal conformance*. The functions and function structure of the user interface should map, in a one-to-one relation, on users' goals and corresponding goal sequences. Functions that users don't need should be hidden for these users.
3. *Information needs conformance*. The information that is provided by the user interface should map, in a one-to-one relation, on the information needs that arise from users' goals. Irrelevant information should not be presented to the users.
4. *Use context*. The human-computer interaction should fit to the envisioned use context and/or situation (e.g. a speech interface should not be designed for noisy environments).

Level of information processing

Based on the Skill-Rule-Knowledge-framework of Rasmussen (1986), we distinguish four support functions: rule provision, information handling, problem exploring, and rule checking. Below we will discuss these functions, following the human information processing steps of figure 5.

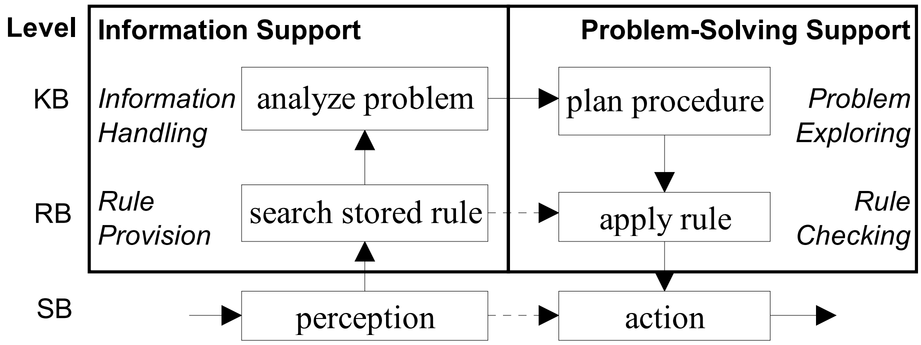

Figure 5: Four cognitive support functions at the knowledge-based (KB), rule-based (RB) and skill-based (SB) level. The broken arrows represent "short-cuts" in the human information processing based on training, experience and/or support.

The *Rule Provision* function provides the normative procedure for solving (a part of) the current problem. Due to training and experience, people develop and retain procedures for efficient task performance (i.e. they apply the rule-based "short-cuts" of figure 5). Performance deficiencies may arise when the task is performed rarely so that procedures will not be learned or will be forgotten, or when the information does not trigger the corresponding procedure in human memory. For these situations, rule provision aims at supplementing human procedural knowledge. We distinguish four *design principles*:

1. The support function should take the initiative to provide information at the right time. Consequently, the user does not need to know when and how to search for information and does not need to invest in these actions.
2. Rule provision should consist of context-specific, procedural task knowledge. The advice is minimal, not more than necessary. Each individual (sub)procedure should however describe a complete problem-solving path to accomplish the (sub)goal.
3. The user interface of the rule provision facility should be an well-integrated part of the human-machine dialogue. A minimal and consistent interaction requires little knowledge and contributes to efficient task performance.
4. The advice should be provided in such a way that the user remains in

the loop of the overall activity by interactive involvement in the process of action executions as part of a task procedure.

The *Information Handling* support filters and combines information to improve situation awareness, i.e. knowledge of the state of the system and its environment (Endsley, 1995). Due to the increasing availability of information, situation awareness can deteriorate without support. Sensor information should therefore be combined into alarms that are structured according to their function, such as fire control, propulsion and energy supply. Furthermore, Information Handling can support the operators in keeping overview by making the structure of the complete system explicit at a global level and by indicating functional relationships between system components. Taken together, Information Handling support should enhance information acquisition and recall in such a way that situation awareness is optimal for the current task performance. Three *design principles* for this type of support can be formulated:

1. An information handling support function should provide an overview of state and process variables, showing the correspondence to system's components (i.e. structure) and the fluctuations in time (history).
2. Alarms should be prioritised according to the current situation and provide information about how to (start to) solve the problem. Important alarms should stand out and irrelevant alarms should be hidden.
3. The support should enable fast and easy access to the requested information with adequate orientation cues and state explanation. It should correspond to the optimal search strategy for the specific task and situation, i.e., support several accurate information acquisition processes of users.

Problem Exploring comes into play when there is not a complete (executable) procedure available to deal with the current alarms and situation. First, the problem and solution space has to be analysed. Subsequently, based on information about the environment (state, process) and information from memory, a procedure must be planned for solving the problem. Based on a mental model (i.e. an internal representation of the system), the person sets local goals, initiates actions to achieve them, observes the extent to which the actions are successful and, if needed, poses new subgoals to minimise the discrepancy between the present state and the desired state. A problem-exploring function consists of a knowledge-based component that can execute some problem-solving activities such as the generation of hypotheses and the selection of an urgent and most promising one. Another possibility is to provide predictions of future states based on the current user actions (e.g. predictive displays, Wickens, 1992). The benefits of such predictions can be very large for a number of tasks if the "predicted path" is explicitly presented and well integrated into the overall presentation of state information. We distinguish three design principles for Problem Exploring support:

1. The user must understand what the support function is doing and why, so that, for example, the user will remain in the loop of task execution.
2. The problem-solving process of the support function should be compatible with user's problem-solving process and enable the involvement of specific user's capabilities.
3. For providing predictions of future states, the "predicted path" should be explicitly presented and well integrated into the overall presentation of state information.

Rule Checking functions recognise when the human operator has strayed from the normative problem-solving path, and help to reach a more correct task outcome (Silverman, 1992). However, as task difficulty increases, a point will be crossed at which subject-matter experts can no longer be assisted by Rule Checking alone. Thus, under conditions of high task load this kind of support seems not to be optimal. A further restriction is that the users must have some knowledge to start their task execution. If they do not know which goals to achieve, then they cannot be critiqued. In general, the first three principles that were identified for Rule Provision apply also to Rule Checking.

Task-set switching

Task-set switching support should comply with the following design principles:

1. For the momentary activity, it should provide an up-to-date overview of the tasks with corresponding actions, including the current status of the activity and the status of each task.
2. The current priority of each task should be shown. Changes in priority should be communicated to the users, so that they can keep a correct, up-to-date situation awareness.
3. Humans are inclined to focus on the tasks they are working on, neglecting tasks with a possible higher priority ("cognitive lock-up", see previous section). The support functions should check if users do the required abstraction from action level to the task and activity level.

Design of cognitive support: An example

The previous section distinguished three load factors and a small set of corresponding support functions that can reduce the negative effects of each factor on task performance and mental effort. The present section will provide an example design of the user's task, support and interaction for the ship control *system manager* that is being developed in the European ATOMOS and DISC projects. To exemplify the design method, we will provide small (simplified) task descriptions for four support functions that reduce the negative effects of a specific load factor on an integrated ship's bridge (see table 2):

- An *information handler* that supports task-set integration for keeping overview of the overall system's state. Based on the system structure and current events, this function provides an overview of alarms and a set of integrated views (i.e., system overviews).
- An *emergency scheduler* that supports task-set switching by providing an overview of prioritised alarms that have to be handled or are being handled, based on the overview of alarms.
- A *rule provider* that supports task-set switching by providing the context-specific procedures for each emergency with the current state of each procedure. Context information comprises the state of the objects in the task set, such as the position of the ship (e.g. harbour at open sea), the location of an emergency (e.g. a fire in room 12) and the maintenance data of a machine (e.g. pump X replaced last week). The combination of emergency scheduler and rule provider provides the action plan for the operator.
- A *diagnosis guide* that supports task-set integration for alarm handling. This guide consists of an overview of possible symptom-cause relations between alarms, e.g. ordered by probability. Based on context information, the alarm overview and the system structure, relations between the alarms are proposed, and based on the settled relations, the context information is refined.

Table 2: Four example support functions that reduce the negative effects of each load factor.

Load factor	Support type	Support function
Time occupied	Information handling	Information handler
Level of information processing	Rule provision Problem exploring	Rule Provider Diagnosis Guide
Task-set switching	Task managing	Emergency Scheduler

User interface design

In general, user interface design follows the general top-down process of software development resulting in user interface specifications at three abstraction levels (Neerincx et al., 2001). On the first level, based on users' goals and information needs, the system's functions and information provision are specified (i.e. the *task level* of the user interface is determined). On the second level, the control of the functions and the presentation of the information is specified (i.e. the "look-and-feel" or the *communication level* of the user interface is established). On the third level, the interface design is implemented in a specific language on a specific platform (i.e. the actual interface or the *implementation level* of the user interface is established). The distinction between these three specification levels is not intended to advertise the "old" waterfall software life cycle. It is based on the notion that software development is a top-down *and* iterative process. For example, specific software components can be developed down to the implementation level after which a re-analysis phase starts for these components and/or new components. The next two subsections present the specifications of the user interface at the task and communication level respectively.

Task Level

Usability at the task level is established by mapping user's goals --and corresponding goal sequences-- on system's operations, and mapping user's information needs on system's information provision. First, such an analysis must specify the functions that users need for specific goals and the corresponding goal- and situation-driven action sequences. Functions not needed should be hidden for the users (i.e. the minimal interface, Carroll, 1984). Second, the mapping of user's information needs on system's information provision is performed by means of an information analysis. Such an analysis establishes which information is required to accomplish a specific goal at a specific moment. Corresponding to the example analysis of task allocations in the Air Defence and Command Frigate, we followed the specification process of figure 4 for the design of cognitive support in the integrated ship's bridge.

Decomposition of Tasks. We specified and, subsequently, enriched the task decomposition with data flow diagrams showing the information that have to be communicated between the tasks. The data flows between human tasks and machine tasks provide a high-level specification of the user interface (see Figure 6).

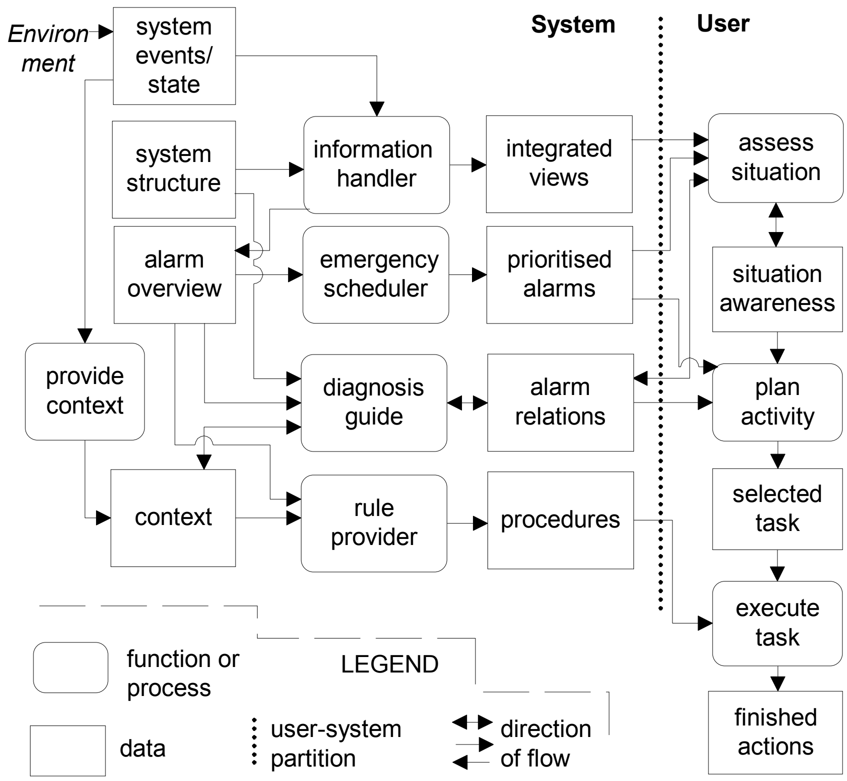

Figure 6: The processes and data flows for the support of operator's situation assessment, activity planning and task execution.

Specification of Events. A distinction can be made between internal events (i.e. events that arise in the ship itself such as a malfunction in the electricity supply) and external events (i.e. events that arise in the ship's environment such as the appearance of another ship). These classes can be subdivided further with increasing level of detail. Table 3 shows an example of a part of an event list containing events of task sets that have to be accomplished with different applications, so that they can be used to show the interface-support functions of the system manager.

Table 3: Example event list

Event category			**Basic event**	**Example event**	**Example consequence**
Extern	Weather	…	…	Storm X approaching	Deviation from route
	Sea state	Current	…	…	…
		Traffic	Vessel passing	…	none
			Vessel gives way	Vessel X from port Y gives way	…
			Vessel changes to collision course	Vessel X from port Y changes to collision course	A collision occurs
Intern	Cargo		Gas-leak	Release of toxic fumes	…
	System Failure	Propul-sion	Engine shutdown	A temperature rise caused shut-down engine X	Ship can not maintain it's current speed
			Temp. above max.level.	Engine X's temperature exceeds set point	Engine speed is limited
		Collision Avoidance	…	Collision sensor X malfunction	A collision occurs
		Navigation	…	Ship off course	Ship can't arrive at ETA
		Electric	Short circuit	Short circuit in cooling pump X	Engine's cooling circuit out of order
			Fire	Fire in the engine room	Engine speed is limited

Specification of Basic Action Sequences. Below, we will show an example action sequence in which two simple procedures (BASs) for fire control and dealing with an engine shutdown are integrated. The fire control procedure consists of alert crew, close section, plan attack route, close doors, extinguish fire, plan smoke removal, close doors and remove smoke. The shutdown procedure consists of detection, determine cause, solve problem and restart engine.

Table 4: Scenario 1 "Fire in the cooling-system of the engine"

Initial state Ship is en route to Hamburg; there are two operators present on the bridge.			
Time	Event		
21.54	Short circuit	Location	Engine's cooling- pump in engine room
		Details	Short circuit causes a fire in the pump, which is located in the cooling system.
		Consequences	Cooling system will not work and the engine temperature will increase.
		Source	None (event is not detected by system)
22.03	Fire	Location	Engine room
		Details	A pump in the engine room is on fire
		Consequences	Unknown
		Source	Smoke detector of Fire Control System
22.06	Max. temp. engine	Location	Engine room
		Details	The temperature of the engine increased beyond the set point.
		Consequences	The engine shuts down after a period of high temperature.
		Source	Propulsion management system
22.08	Engine shut-down	Location	Engine room
		Details	The temperature was too high for the critical period.
		Consequences	The vessel cannot maintain its current speed.
		Source	Propulsion management system

Specification of Scenarios. To specify the interface support of the System Manager, scenarios are created with multiple events from different domains. Table 4 presents an example of such a scenario: its initial state, the events with time of appearance, location, details, consequences and source (i.e., the detector that provides the event to the operator). This scenario contains a number of different events that require integration of (sensor) information, diagnosis, procedural task knowledge and task-set switching. Therefore, this scenario can be used to envision the support functions of the system manager. A short circuit in a cooling pump causes a fire and the corresponding fire alarm is presented to the operator. This short circuit also causes a malfunctioning cooling circuit, so that the temperature of the engine increases and is above the maximum level for a specific duration, resulting in the presentation of the corresponding temperature alarm. The high temperature causes an engine shutdown and the corresponding alarm is presented to the operator. First, the scenario imposes task-set switching between dealing with the fire and the engine shutdown. Second, the scenario imposes task-set integration: the operator has to abstract from the fire and shutdown tasks to infer the relations of these separate alarms. Subsequently, the shutdown task can be continued with the correct procedure.

Specification of Compound Action Sequence. Scenario 1 is transformed into a Compound Action Sequence (CAS) using the procedures (BASs) for handling the events of this scenario. The CAS provides an overview of actions and processes performed during the scenario. Figure 7 presents the CAS for scenario 1. The following components can be distinguished:

- *Actors.* In this CAS there are two actors (the System and the Operator). The System actor is subdivided in two columns (the Applications and the System Manager). The Operator column is subdivided in columns representing executions of three tasks: the fire, shutdown, and assess-and-plan task.
- *Timeline.* The vertical axis shows the timeline. It should be noted that the timeline is not linear (a block sometimes represents 6 minutes, while at another place it represents ½ minute), so that the diagram can zoom in on a time-period where more actions are performed.
- *Events.* A column is used to present events that occur during the scenario. Figure 7 shows four events: short circuit, fire, maximum temperature engine exceeded, and engine shutdown. The first one is not detected by the actors (i.e. no flow of actions/processes result directly from this event).
- *Actions.* At a certain moment the operator performs actions belonging to a task (column).
- *Processes.* The system manager and applications run several processes to support the users with their task performance.
- *User-system partition.* The broken line between the two actors shows the user-system partition (i.e. the human-machine interaction at the task level). The user interacts with multiple processes at the same time. For example, during the fire task the operator uses the rule provision process of the System Manager. At the same time the user performs operations using the Fire Control System application. In particular, Figure 7 shows the interaction with the support functions for task-set switching and task-set integration: Information Handler (IH), Scheduler (SC), Rule Provider (RP) and Diagnosis Guide (DG).

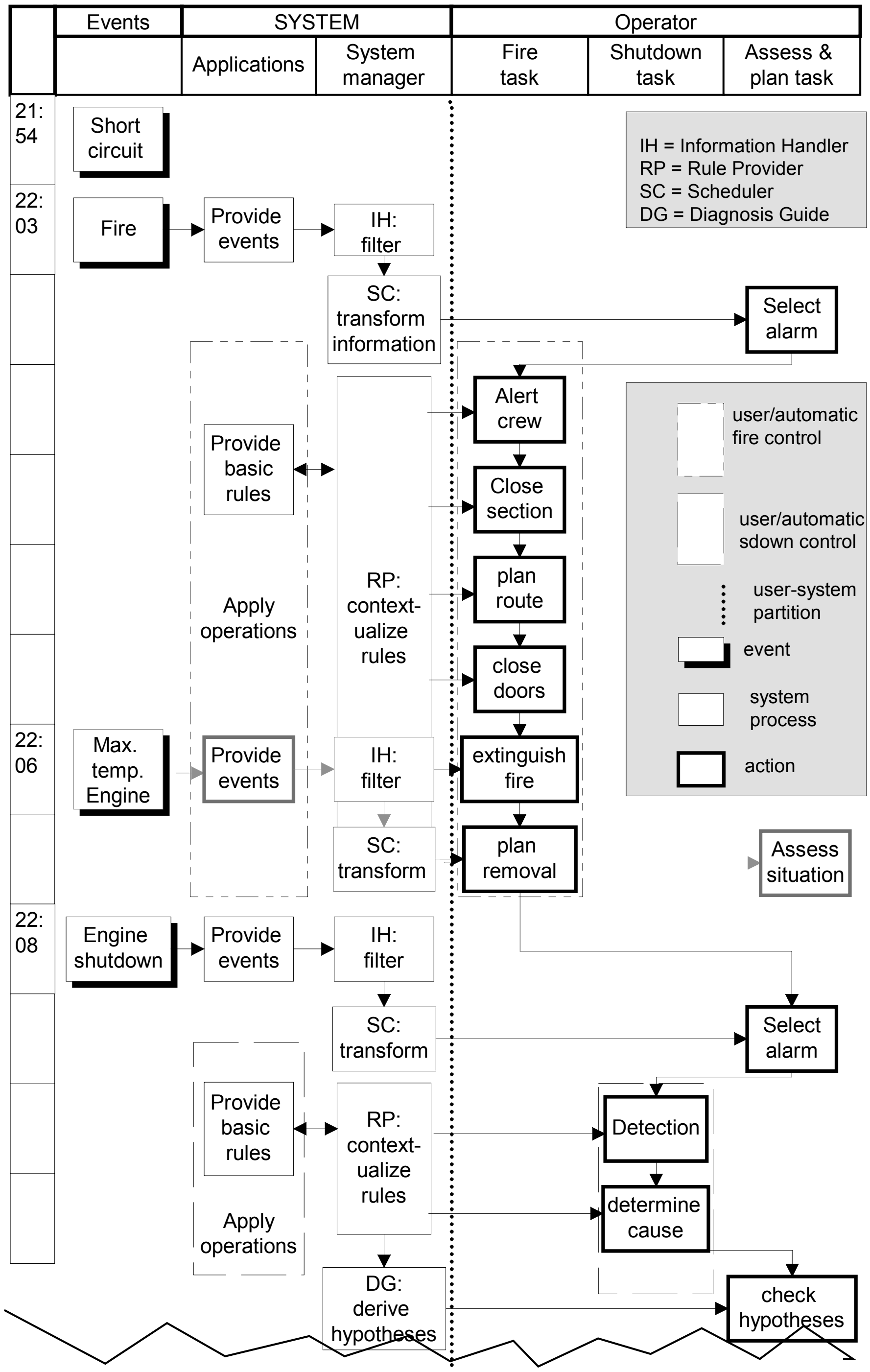

Figure 7: Part of the Compound Action Sequence (CAS) for scenario 1.

Communication Level

The previous subsection specified the user interface at the task level: a high-level description of the human-computer interaction and the role of the system manager. This specification will be transformed to a low-level description of the presentation and control elements based on user requirements, HCI standards and technical constraints. The plain interface is an interface for operators who have always sufficient knowledge and capacities available to execute their tasks (Neerincx & de Greef, 1998). The system manager integrates context-specific task support into this interface to complement possible knowledge and capacity deficiencies, resulting into the support interface of the *system manager*. First the plain interface of the application manager will be described and, subsequently, the support that is integrated into this interface by the system manager.

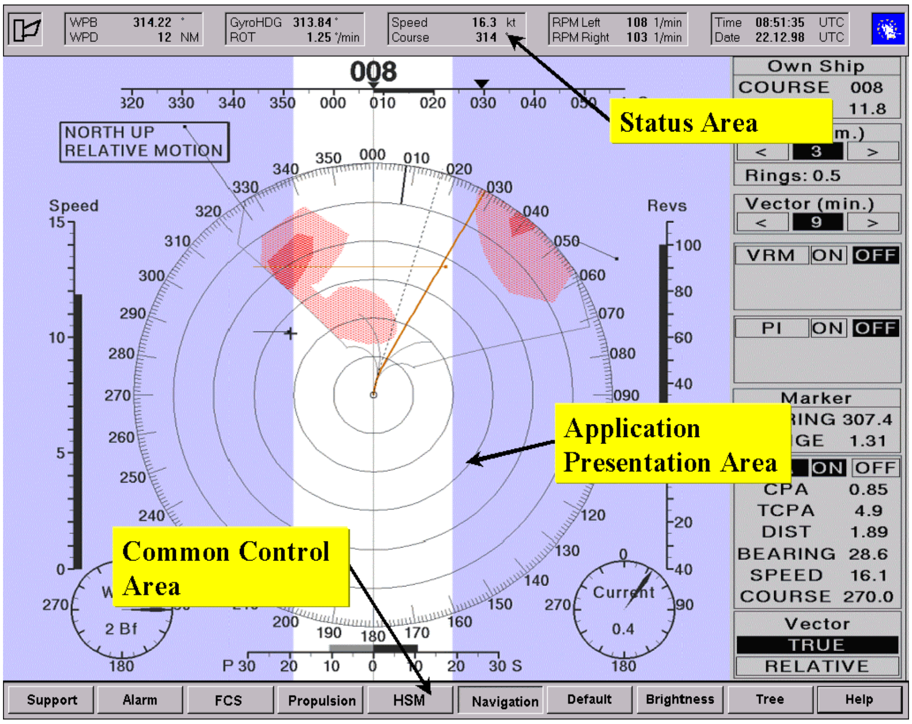

Figure 8: The "plain" interface of the Application Manager (with an extra support button).

Plain Interface. Figure 8 shows the plain user interface that consists of three parts. The *status area* is shown on the top part of the screen. This area has a fixed size, is always available and is never covered by overlapping windows or dialogue boxes. The main role of the status area is to present real-time status and alarm information. An 'alarm bell' is shown on the left side of the status area. The alarm bell is used to indicate the alarm status: no emergencies (grey), emergencies (red), new emergencies (a short period of blinking red/grey accompanied by a modulated sound signal), and priority raise of emergencies (blinking and sound).

The *application presentation area* in the middle of the screen is used to present the active application (Figure 8 shows the navigation application).

The *common control area* at the bottom of the screen has a fixed size, is always available and is never covered by overlapping windows. Its main use is to switch from one application to another. The first button on the left is added to the application manager interface to switch the system manager's support on or off (see below).

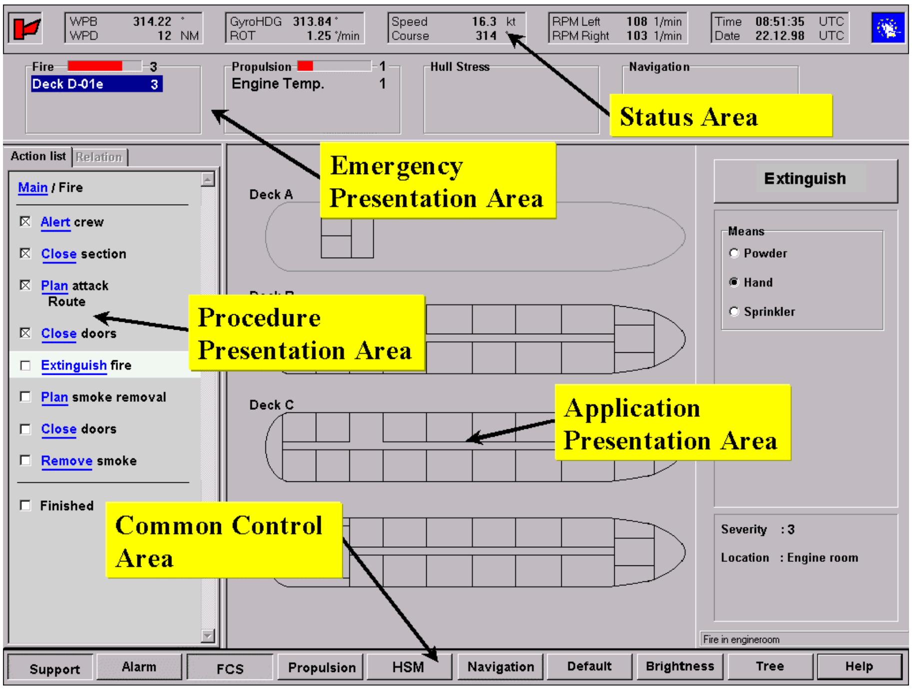

Figure 9: The user interface in support mode.

Support Interface. Figure 9 provides an example of the user interface in support mode (i.e. an example interface state during fire control). Two new areas can be identified in this mode. An emergency presentation area situated directly underneath the status area and a procedure presentation area on the left side of the application presentation area. The application presentation area is somewhat smaller than in the plain interface to make room for the two new areas.

A first type of *Information Handling* appears in the emergency presentation area combining low-level alarms (e.g. sensor values) and categorising the resulting emergencies into groups (In the current example: Fire, Propulsion, Hull Stress and Navigation). In this way, the amount of alarm information on the screen is reduced compared to the "classical" alarm lists. This report presents an example for the system manager's user interface for a workstation with only one display. When the operator can use two displays, the emergency presentation area can be presented on a separate one. Another type of Information Handling support is offered by integrated views in the application tree (for example an integrated view of the propulsion, electricity, track control and cargo to show the interdependencies). Such a view offers an overview of state and process variables, the fluctuations in time *and* the relevant relations between different systems (applications).

Scheduling support is located in the emergency presentation area. It is possible that two emergencies of the same type occur simultaneously. The emergencies are ordered in the appropriate group according to time of occurrence. Each emergency is presented as a hyperlink that "loads" the corresponding procedure in the procedure presentation area. Selection of an emergency is indicated by 'inverted' video. Next to each emergency a number is given to indicate the priority of that particular emergency. Next to the group name a priority indicator (horizontal bar) and the corresponding number are given, showing the highest priority of that group (the fire group has the priority of "3" in the example of Figure 9).

The emergency presentation area provides the overall work plan for emergency handling. It can be viewed as a shared work plan if multiple operators are involved in such tasks: e.g. one operator who executes a fire procedure and another operator who prevents a collision. These operators can help each other or switch tasks. Suppose that the operator who is engaged in a fire fighting procedure needs advice from his colleague. The colleague can then obtain the same view as the other operator (but to prevent confusion he can not operate the other operators procedure or application) and advise him. When the original 'fire fighter' changes to another emergency (for example the collision prevention), the procedure is released and the other operator can continue at the right place in the procedure because of the checkmarks made by the other operator.

The procedure presentation area provides web-browsing functionality and contains two tabs: 'Action List' and 'Relation'. *Rule Provision* support is provided by the first tab that presents a list of all actions that must be performed to deal with the selected emergency:

- For every emergency, the content of the page will be dynamically created: actions that apply to the current emergency and context are selected, the objects of these actions are instantiated and the result is presented in the page. In this way, a context-specific concrete procedure can be formulated (Neerincx & De Greef, 1998). Each procedure step can contain hyperlinks that refer to the corresponding sub-procedure or to an explanation of the content.
- Checkmarks can be placed in the appropriate checkboxes to indicate that a specific step in the procedure was completed. The background of the following step in the procedure is highlighted while the background of the other actions is grey (i.e. the first not check-marked step is highlighted). It is important to note that the user and the system manager can place checkmarks (in different colours). Each procedure ends with a 'Finished' step to assure that the emergency is really solved and to prevent that an emergency 'suddenly' disappears from the scheduler without the operator being aware of it. States of procedures are preserved (by means of the checkmarks) till the emergency is finished.
- The relevant application for the current (i.e. highlighted) step is activated and presented automatically in the application presentation area.

The 'Relation' tab in the procedure presentation area contains *diagnosis guidance*, i.e., hypotheses about relations between two or more of the present emergencies (one of these emergencies must be the currently active emergency). Unlike the 'Action List' tab that is always available, the 'Relation' tab is not available most of the time (indicated by being dim). It becomes available (the name blinks a few times and is not dimmed) whenever one or more hypotheses about emergencies are discovered by the system manager (in the procedure presentation area of figure 9):

- Each hypothesis can contain hyperlinks that refer to a procedure to evaluate that hypothesis (when no procedure is available more information about the hypothesis will be presented).
- A toggle switch is presented before each hypothesis to indicate whether the hypothesis is true (Y), not true (N) or still open (no button pushed). Each diagnosis of the relations ends with a 'Finished' checkbox.
- After the diagnosis finished, the system manager will indicate and explain whether the choices were consistent in a dialogue box. When the choices were not consistent, the 'Relation' tab is shown again after

this box. When the choices were consistent, the dialogue box proposes changes in the context model that correspond to the hypotheses assessment. The operator can either approve of these changes (by clicking <OK>) or don't approve (by clicking <Not OK>). When the operator presses <OK>, the changes are made and immediately used to dynamically improve the procedures; the operator is returned to the 'Action List' tab that contains the improved procedure. When the operator presses <Not OK>, the changes are not made and he is returned to the 'Relation' tab.

Evaluation

A *storyboard* visualises a number of possible consecutive steps within an interaction sequence situated in typical use context. We created storyboards for the support interface in order to collect feedback from domain experts and possible users, and to prioritise the support functions taking technological design constraints into account. Our storyboards consisted of a sequence of screendumps (such as figure 9) for a specific scenario or compound action sequence with a narrative description of the human-computer dialogue. In the first implementation stage, another project partner implemented the most cost-effective and feasible functions in a demonstrator. For example, the implementation of a diagnosis guide was postponed, because empirical foundation of its effectiveness was lacking and the required technology is rather advanced.

In correspondence with the cognitive engineering approach of the introduction in this chapter (see figure 1), we did a first internal analytical *assessment* of the user interface specification, till the level of storyboards, to test if it complies with the user requirements and HCI guidelines. It was shown that the guidelines were generally well addressed. The user interface design can be viewed as a manifestation of current human-computer interaction knowledge and corresponds with new interface proposals for naval ship control centres and manned space labs (Neerincx et al., 2001). The support is expected to diminish cognitive task load in critical situations, to facilitate dynamic allocation of tasks to multiple users and to keep users in the loop of human-machine task execution. A small number of experiments validated central design principles and showed positive effects of the Rule Provider (Neerincx & de Greef, 1998; Neerincx et al., 2001). In an evaluation with 57 navy cadets of the Royal Netherlands Navy, the support functions leaded to a substantial increase in operator performance, especially at high task load. We did not find negative "out-of-the-loop effects" of the support, like "automatic" following of incorrect advices or decreased situation awareness (Grootjen et al., 2002). Currently, part of the support is being implemented on the bridge of an icebreaker in the European ATOMOSIV project.

Discussion

This chapter presented an overview of a practical theory on cognitive task load and support that is being developed in a cognitive engineering framework (figure 1). Recent experiments in laboratory *and* real-world settings showed the large effects of the three load factors on task performance and mental effort, and provided empirical foundation of the cognitive support functions. The theory has been integrated in a method for cognitive task analysis that can be used in the early stage of system development processes. In this chapter we chose to give an overview of the theory and method, and some example applications. The empirical foundation is being provided elsewhere (e.g. Neerincx & van Besouw, 2001; Grootjen et al., 2002).

Recently, the cognitive task load model was integrated into an ERGOtool for the assessment of (future) assembly lines in industry via questionnaires and scenario observations. The ERGOtool is being applied and tested at different European companies (Van Rhijn et al., 2001). In addition to this tool, we are developing simulations of the 'Compound Action Sequences', that can be used to "refine" the task allocation and to assess the effects of support functions in an early software development stage. As another example, we applied the analysis method for Cognitive Task Analysis to assess the task load of the operator in the future control room for a 6-kilometre motor traffic tunnel (the "Westerscheldetunnel"). The analysis identified a number of bottlenecks that should be resolved in order to guarantee adequate operator performance. In sum, current and near-future research will provide results to improve the method and its theoretical and empirical foundation. Persons who were involved in its development have mainly applied the method so that we have (yet) limited experience of its "usability" (but see for example van Rhijn et al., 2001). The "complete" Cognitive Task Analysis method comprises a lot of information with respect to the "load and support" theory, the specification and assessment techniques, the relevant user interface guidelines and standards, and the increasing set of examples. For the European Space Agency (ESA), we developed a customised version of the method for designers of the user interfaces in the International Space Station. To make the method easy to access, maintain and update, it is being provided as an interactive guide, a Web-based usability handbook, providing example applications and validations (i.e. enhancing the HCI foundation for space lab interfaces). We expect to assess its usability in the near future.

Our analysis approach focuses on (external) cognitive task factors at a "meso-level" (i.e. we do *not* deal with macro-level work schedules in terms of days or micro-level dialogue issues in terms of milliseconds). A current project will further explicate the relations of task demands (e.g. of communication tasks) with internal cognitive and emotional states. More elaborate micro-level analyses as supported by the Information Processing model of Hendy & Farrell (1997) are related to ours, but are difficult to apply for complex real-world task environments (cf. Gray & Kirschenbaum, 2000; Nardi, 1996). In future, they may become more practical, e.g. by simulation environments, and help to further analyse task elements for specific details.

Conclusions

This chapter presented a practical theory on mental load and cognitive support in process-control tasks. The theory comprises a model of cognitive task load that describes load in terms of three behavioural factors: the percentage time occupied, the level of information processing and the number of task-set switches. The higher the value of each factor, the higher the load will be. Furthermore, the practical theory distinguishes specific cognitive support functions that affect these values. For example, procedural support will diminish the *level of information processing* if it complies with a number of guidelines. A second example is task management support that will diminish the *task-set switching* demands if it fulfils a compound set of user requirements. The theory of mental load and cognitive support comprises the effects of task and interface characteristics on performance and mental effort to enable an iterative process of human task and interface design. For this objective, the theory is integrated into a method for cognitive task analysis that can be applied in the first phase of software development for the assessment of task allocations and high-level system specifications, and for the design of cognitive support functions.

In a first example, the method was applied to the assessment of the specifications of the Integrated Monitoring and Control System of the future Air Defence and Command Frigate. As a result of the analysis, particularly the large number of task-set switches was identified as an overload risk for envisioned activities in the future Ship Control Centres. Because mental load was described in terms of task and interface characteristics, recommendations could be formulated for task allocation and interface design.

In a second example, the Cognitive Task Analysis method was applied to develop a support interface for the integrated ship's bridge. An evaluation with a prototype interface showed the benefits of the support (Grootjen et al., 2002). Based on this design and evaluation, the ATOMOSIV project will implement a support interface on an icebreaker.

Taken together, these examples comprise an integrated approach on task allocation enhancement and design of cognitive support. Recent results of this approach are promising, providing for example indications that cognitive support can help to realise adequate (dynamic) task allocations.

Acknowledgements. Research is teamwork. We are very grateful to the Royal Netherlands Navy for their contribution to the Cognitive Task Analysis. In particular, we would like to thank Eddie Flohr and Wim Helleman for their provision of domain knowledge and their comments on previous parts of the method. At TNO Human Factors, Jasper Lindenberg, Mark Ruijsendaal and Sasja van Besouw provided substantial contributions to this research.

References

Beevis, D. (Ed.) (1992). *Analysis techniques for man-machine systems design, Vol 1 & 2*, NATO/Panel 8-RSG.14, Technical Report AC/243(Panel 8)TR/7. Brussels, NATO.

Boehne, D.M. & Paese, P.W. (2000). Deciding whether to complete or terminate an unfinished project: a strong test of the project completion hypothesis. *Organizational Behavior and Human Decision Processes*, 2, 178-194.

Carroll, J.M. (1984). Minimalist design for active users. In B. Shackle (Ed.), *Proceedings of Interact'94*. Amsterdam, North-Holland. pp. 219-225

Carroll, J.M. (Ed.) (1995). *Scenario-based design: envisioning work and technology in system development*. New York, etc., Wiley & Sons Inc.

Endsley, M.R. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37, 32–64.

Gray, W. D., & Kirschenbaum, S. S. (2000). Analyzing a novel expertise: An unmarked road. In Schraagen, J.M.C., Chipman, S.E. & Shalin, V.L. (Eds.), *Cognitive Task Analysis*. Mahwah, NJ, Erlbaum.

Green, T.R.G. (1990). Limited theories as a framework for human-computer interaction. In D. Ackerman & M.J. Tauber (Eds.) *Mental models and human-computer interaction 1*. Amsterdam, The Netherlands, Elsevier Science Publishers.

Green, T.R.G. (1998). The conception of a conception. *Ergonomics*, 41, Vol. 41, 143-146.

Grootjen, M., Neerincx, M.A., Passenier, P.O. (2002). *Cognitive task load and support on a ship's bridge: Design and evaluation of a prototype user interface*, INEC 2002 Conference "The marine engineer in the electronic age".

Hendy, K.C. & Farrell, P.S.E. (1997). *Implementing a model of human information processing in a task network simulation environment.* Report DCIEM No 97-R-71. North York, Ontario, Canada, Defence and Civil Institute of Environmental Medicine.

Hollnagel, E. (1998). *Cognitive Reliability and Error Analysis Method (CREAM).* Elsevier Science Ltd, Oxford, UK.

Kerstholt, J.H., & Passenier, P.O. (2000). Fault management in supervisory control: the effect of false alarms and support. *Ergonomics, 43*(9), 1371-1389.

Kirwan, B. & Ainsworth, L.K. (Eds.) (1992). *A guide to task analysis*. London, Washington DC, Taylor & Francis.

Levine, J.M., Romashko, T. & Fleishman, E.A. (1973). Evaluation of an abilities classification system for integrating and generalizing human performance research findings: an application to vigilance tasks. *Journal of Applied Psychology*, 58(2), 149-157.

Nardi, B.A. (ed.) (1996). *Context and consciousness. Activity theory and human-computer interaction.* Cambridge, MA, the MIT press.

Neerincx, M.A. & van Besouw, N.J.P. (2001). Cognitive task load: a function of time occupied, level of information processing and task-set switches. In D. Harris (ed.), *Engineering Psychology and Cognitive Ergonomics, Volume Six: Industrial Ergonomics, HCI, and Applied Cognitive Psychology* (Chapter 31). Aldershot, etc., Ashgate. pp. 247-254.

Neerincx, M.A., van Doorne, H. & Ruijsendaal, M. (2000). Attuning computer-supported work to human knowledge and processing capacities in Ship Control Centres. In Schraagen, J.M.C., Chipman, S.E. & Shalin, V.L. (Eds.), *Cognitive Task Analysis*. Mahwah, NJ, Erlbaum.

Neerincx, M.A. & de Greef, H.P. (1998). Cognitive support: extending human knowledge and processing capacities. *Human-Computer Interaction*, 13(1), 73-106.

Neerincx, M.A. & Griffioen, E. (1996). Cognitive task analysis: harmonising tasks to human capacities. *Ergonomics*, 39(4), 543-561.

Neerincx, M.A., Ruijsendaal, M. & Wolff, M. (2001). Usability Engineering Guide for Integrated Operation Support in Space Station Payloads, *International Journal of Cognitive Ergonomics,* 5(3), 187-198.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA, Harvard University Press.

Parasuraman, R. (1986). Vigilance, monitoring, and search. In Boff, K.R., Kaufman, L. & Thomas, J.P. (eds.), *Handbook of perception and human performance*, volume 2: cognitive processes and performance, chapter 43. New York, Wiley.

Rasmussen, J. (1986). *Information processing and human-machine interaction: an approach to cognitive engineering.* Amsterdam, Elsevier.

Rhijn, G. van, Looze, M. de, Lingen, P. van, Tuinzaad, B. & Leskinen T. (2001). IFAC symposium on Human Machine Systems (HMS 2001), 18-20 September 2001, Kassel, Germany.

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

Scerbo, M.W. (2001). Stress, workload and boredom in vigilance: a problem and an answer. In Hancock, P.A. & Desmond, P.A. (ed.), *Stress, Workload and Fatigue.* Mahwah, New Jersey, Lawrence Erlbaum Associates.

Silverman, B.G. (1992). Human-computer collaboration. *Human-Computer Interaction*, 7, 165-196.

Simon, H.A. (1981). *The sciences of the artificial, second edition.* Cambridge, etc., England, The MIT Press.

Wickens, C.D. (1992). *Engineering Psychology and Human Performance (2nd ed.).* New York, NY, HarperCollins Publishers Inc..

Acronyms

ATOMOS	Advanced Technology to Optimise Manpower On board Ships
BAS	Basic Action Sequence
CAS	Compound Action Sequence
DISC	Demonstrator Integrated Ship Control
DG	Diagnosis Guide
HCI	Human-Computer Interaction
IH	Information Handler
IMCS	Integrated Monitoring and Control System
KB	Knowledge-Based
RB	Rule-Based
RP	Rule Provider
SB	Skill-Based
SC	Scheduler