

A Framework for Dynamic Task Allocation

Instantiated for Cognitive Task Load-based Adaptive Automation

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science (MSc), 30 ECTS
August 21, 2013

Student:	Tinka R.A. Giele
Number:	3373517
University:	Utrecht University
Faculty:	Humanities
Master Programme:	Cognitive Artificial Intelligence
Track:	Logic and Intelligent Systems
Organization:	TNO
Project:	NIFTi
Supervisor TNO & second reviewer:	Tina Mioch
Supervisor Utrecht University & first reviewer:	John-Jules Ch. Meyer
Third reviewer:	Vincent van Oostrom
Period:	29/01/2013 - 31/07/2013



TNO innovation
for life



Universiteit Utrecht

Abstract

As technology advances, artificial agents such as robots are increasingly deployed to work on tasks in complex and dynamically changing environments. Often these sophisticated robots work together with human agents in a team. Because of these developments, the need for research into cooperation in mixed human-robot teams is increasing. An important aspect of cooperation is task allocation. Static task allocation is often not sufficient for dynamically changing environments, so dynamic task allocation is needed.

In this thesis, a high-level framework for dynamic task allocation, aimed at improving team performance in mixed human-robot teams is presented. The framework details how context information can be used to find possible role assignments for actors and to evaluate these role assignments. The framework describes the important concepts in context information that influence team performance and can be used to dynamically allocate tasks. Secondly, the framework details how to use these role assignments with evaluation to find the optimal task allocation for a team.

One of the important factors in context information is the cognitive task load of a human agent. Cognitive task load is an important predictor of human performance and is dependent on the tasks that are assigned to a human. The framework is used as a base for designing a model for adaptive automation. The model takes into account the cognitive task load of an operator and the coordination costs of switching to a new task allocation. Based on these two context factors it finds the optimal level of autonomy of a robot, separately for all tasks that need to be executed.

This model is instantiated for a single human agent cooperating with a single robot in the urban search and rescue domain. A small experiment is conducted aimed at testing the model. Some encouraging results are found: the cognitive task load of participants mostly reacted to the model as intended. Furthermore, important focus points for improving the model are identified such as taking into account more context information, e.g. capabilities (human vs. robot) and preferences.

Contents

1	Introduction	7
2	Previous Work	10
2.1	Teams	10
2.2	Team Performance	11
2.3	Cognitive Task Load	13
2.4	Dynamic Task Allocation	14
2.5	Adaptive Automation	16
3	Framework for Dynamic Task Allocation	19
3.1	Relation to Previous Work	19
3.2	Overview of the Framework	23
3.3	Formalization of the Framework	25
3.3.1	Environment Models	26
3.3.2	Actor Models	26
3.3.3	Task Analysis	29
3.3.4	Situation Analysis	29
3.3.5	Option Generation and Weighing	30
4	Model for Adaptive Automation	34
4.1	General Description of Model	34
4.2	Levels of Autonomy	35
4.2.1	General Description of Modeling Levels of Autonomy	35
4.2.2	Formalization of Modeling Levels of Autonomy	36
4.3	Cognitive Task Load	38
4.3.1	General Description of Modeling Cognitive Task Load	39
4.3.2	Formalization of Modeling Cognitive Task Load	39
4.4	Coordination Costs	42
4.4.1	General Description of Coordination Costs	42
4.4.2	Formal Description of Coordination Costs	43
4.5	Utility Function	45
5	Experiment	47
5.1	Instantiating the Model for the Urban Search and Rescue Domain	47
5.1.1	Urban Search and Rescue	47
5.1.2	Automation Analysis	49
5.1.3	Cognitive Task Load	53
5.2	Method	56
5.2.1	Design and Hypotheses	56
5.2.2	Participants	57
5.2.3	Task	57
5.2.4	Materials and Set-up	58

5.2.5	Procedure	61
5.2.6	Measurements	62
5.3	Results	63
5.3.1	General Results	63
5.3.2	Hypothesis 1: The Model Reallocates Tasks at the Right Moment	65
5.3.3	Hypothesis 2: The Model Chooses Appropriate Reallocations	66
5.3.4	Hypothesis 3: The Real Shift in CTL Corresponds to the Predicted Shift in CTL	67
5.4	Conclusion (Experiment)	72
6	Discussion and Future Research	73
6.1	Framework for Dynamic Task Allocation	73
6.2	Model for Adaptive Automation triggered by Cognitive Task Load	73
6.2.1	Trust in Model	73
6.2.2	Factors in Choosing a Task Allocation	74
6.2.3	Configuration	76
6.2.4	Representation and Notification	76
6.3	Experiment Urban Search and Rescue	76
6.3.1	Error in Prediction	77
6.3.2	Lack of Training and Frustration	78
6.3.3	Variation in CTL	78
6.3.4	Short Term Interaction	79
6.3.5	Measuring performance	79
7	Conclusion	80
	Bibliography	82
	Appendix 1: Glossary	88
	Appendix 2: Casting the Task Allocation Problem to an Instance of the Set Partitioning Problem	90
	Appendix 3: Results of Domain Independent Pruning of Possible Role Assignments	93
	Appendix 4: Task description for participants	95
	Appendix 5: Pseudo Code Description of the Model	102
	Appendix 6: Questionnaire - Before Experiment	103
	Appendix 7: Questionnaire - After Experiment	106
	Appendix 8: CTL Data Experiment Runs	110

List of Figures

1	Kozolowski’s framework for adaptive team performance	12
2	The 3D Cognitive Load Space, as proposed by Neerincx [8]	14
3	Overview of Gerkey’s taxonomy [51] of dynamic task allocation problems	20
4	A high level overview of Gerkey’s analysis [51] of dynamic task allocation problems	21
5	Gerkey’s analysis [51] of dynamic task allocation problems	21
6	Overview of the proposed framework.	23
7	The task of navigation at three different levels of autonomy.	36
8	Values of problem region boundaries in the Cognitive Load Space	40
9	Some pictures taken during NIFTi experiments	48
10	Overview of the different tasks in a typical USAR mission	50
11	Overview of Colin’s [34] CTL model’s process.	54
12	A screenshot of the observer interface.	59
13	Some screenshots of the USARsim environment.	60
14	A screenshot of the TrexCOP application.	60
15	Overview of the experimental set up.	61
16	Scatter plot of relation real and predicted LIP values.	68
17	Scatter plot of relation real and predicted MO values.	69
18	Scatter plot of relation real and predicted TSS values.	69
19	Scatter plot of relation real and predicted shifts in LIP.	70
20	Scatter plot of relation real and predicted shifts in MO.	71
21	Scatter plot of relation real and predicted shifts in TSS.	71

List of Tables

1	An overview of the tasks at different levels of autonomy.	50
2	The relation between different levels of autonomy for navigation (Nav) and obstacle detection/avoidance (Obs).	52
3	The relation between different levels of autonomy for navigation (Nav) and receiving and processing information from others (Info).	52
4	The configuration for the CTL model.	55
5	Responses to statements concerning timing	65
6	Responses to statements concerning appropriateness of reallocations	66
7	An overview of the tasks at different levels of autonomy and their occurrence.	110

Acknowledgements

First and foremost I would like to thank Tina Mioch for motivating me to take on the challenge of doing this research. Six months ago I had the idea and enthusiasm, but without her motivation I would not have dared to think I would be able to conduct this research in six months. The weekly sessions with feedback always inspired me to move forward and try to achieve more.

Also, I would like to thank John-Jules Meyer for his time and feedback throughout these months and Vincent van Oostrom for his willingness to be third reviewer.

Further thanks go out to all people that helped me in some way or another with this research. For example Pjotr van Amerongen, Machiel Pronk, Martijn van den Heuvel and Wessel Staal for their help on technical challenges, Thomas Colin and Mark Neerincx for advice on how to use the cognitive task load model, Robine Donken for statistical analysis advice and all interns and friends that participated in the (pilot) experiment (and often managed to come to Soesterberg for me on extremely short notice when other participants canceled).

Last but certainly not least, I would like to thank Timo Vosse and Christian Willemsen for their efforts as Wizards of OZ in the experiment. This effort was very much appreciated and along with their great enthusiasm helped me run a successful experiment.

1 Introduction

Teams are an integral part of society. Professional teams, such as firefighters responding to an emergency, are important in organizations. But also in personal life teams occur often, for example in sports or when a group of friends helps someone move to a new place. The importance of teams has been recognized and embraced with enthusiasm in the last few decades [1]. Teams have proven to be more than the sum of their parts, meaning a team as a whole can achieve more than its members in isolation. In teams, a good allocation of tasks is important and helps team performance. Task allocation can be static, this means all roles are divided before starting the tasks and do not change during execution of tasks. But when an environment is dynamic or states¹ of team members change, properties of tasks can change and switching the task allocation could be beneficial. This so-called dynamic task allocation can also benefit team performance [2].

For example, imagine two people that need to relocate a lot of books. The books need to be put into boxes per subject, the subject should be written on a label, and then the boxes should be loaded into a car. One of the persons (Albert) is stronger and thus more suited to carry boxes, the other (Ben) will pack and label the boxes. After a while of carrying and packing books, Albert is getting very tired and needs to sit down (change in state of team member). Ben finds some books on geography, which he knows very little about (dynamic environment) and has trouble sorting and labeling them. Dynamic allocation of tasks would now allow Albert (who happens to be a geographer) and Ben (who is well rested) to switch tasks, which will improve team performance (number of books getting boxed, correctly labeled and loaded in to the car).

Dynamic task allocation is not only important for human teams, but also for mixed human-robot teams [3] and robot-only teams [4]. Making a human team member responsible for dynamically allocating tasks, causes extra workload [5]. If we do not want to cause extra workload, the dynamic task allocation should be automated. To be able to implement this, we need to devise models that dynamically allocate tasks. These models should describe how to improve team performance, using dynamic allocation of tasks. As to our knowledge, the underlying principles needed to design these models have not been explicated yet, thus the main research question of the current study is:

How can we use dynamic allocation of tasks to improve team performance?

To answer this question, a framework is needed that describes important general concepts that are of influence on task allocation as to improve team performance. This framework can serve as a base for building models that make use of specific concepts, for specific domains.

The process in which robots² in a mixed human-robot team operate under automated dynamic task allocation is called adaptive automation if these robots can be assigned

¹For example: cognitive state, affective state or physical state.

²As a shorthand, the word robot is used whenever we mean all non-human mechanical actors. For example, these could also be a virtual agents.

different levels of autonomy on tasks. Instead of just assigning a task to either a robot or a human actor, intermediate levels of autonomy can be defined in which joint effort by the robot and human (an operator) is needed to complete the task [6]. An example is the task of navigation of a robot. We can assign the task of navigation fully to the operator (tele-operation), which means the robot is fully non-autonomous on that task. We can also give the robot full autonomy in navigation, letting it decide for itself where it drives and how it gets there. An intermediate level of autonomy could be navigation through waypoints in which the operator sets waypoints for the robot and the robot drives autonomously along these points. Recent research shows that dynamically adapting autonomy levels of robots could help towards optimizing team performance, when this process is automated [7].

An important challenge in adaptive automation is deciding when to change the level of autonomy of the robot. A trigger for this could be the cognitive task load of the operator [8]. Firstly, this is because cognitive task load has an influence on performance [9, 10]. If the cognitive task load of a human is kept at a moderate level, performance of this human is optimal [8]. If the cognitive task load of a human is low for some time, underload might occur, decreasing performance due to boredom. Conversely, high cognitive task load might mean task overload for the human, decreasing performance because the human is unable to accomplish all tasks he has been assigned.

Secondly, cognitive task load itself can be influenced by adapting levels of automation. Adapting the autonomy level of a robot could be used to balance task load for the operator, as level of autonomy and operator task load are inversely correlated if other factors remain stable [11, 12]. But the same does not hold for the relation between autonomy levels and operator performance. Setting robot autonomy very high might cause human-out-of-the-loop problems, which decrease operator performance. Conversely, setting autonomy very low might cause task overload for the operator, decreasing performance. We need to devise a model that describes how to adjust autonomy levels of the robot, triggered by the cognitive task load of the operator, as to improve performance.

Adaptive automation triggered by cognitive task load of the operator is thus likely to improve the performance of the operator. This does not necessarily imply, however, that it will also improve team performance. Team performance does not rely solely on operator performance, but also on performance of the robot team member(s). Furthermore, team performance is not just an addition of the performances of all the team's members [2], but likely includes factors that go beyond the individual level, for example communication. How team performance is defined, depends on the specific task. When we have a framework for dynamic task allocation, this can be used as a basis for building a model that describes how to set the levels of autonomy on different tasks for the robot(s), triggered by the cognitive task load of the human actors. We can evaluate our model by instantiating it for a specific task, and experimentally validate the effects of the adaptive autonomy.

We define the following subquestions:

- Can we build a general framework describing the important concepts that influence team performance, that can be used to dynamically allocate tasks?

- Can we apply the framework to design a model for adaptive automation, triggered by cognitive task load?
- Can we validate the effects of our model in an experimental setting?

The focus of the current study is directed at designing and formalizing the general framework and the model for adaptive automation. The model will be instantiated for an experimental setting and a small experiment testing the model will be conducted. This is the first step towards experimentally validating that the model improves performance.

The remainder of this thesis is structured as follows: We will start by giving a review of the previous work done in relevant fields in Section 2. The general framework that describes the components for deciding on an allocation of tasks will be described in Section 3. After this, we give an example of how the framework can be applied, building a model for adaptive automation, triggered by cognitive task load in Section 5. Next, we describe an experiment validating the model, instantiated in the urban search and rescue domain in Section 5. Some points for discussion and directions for future research are mentioned in Section 6 and lastly conclusions are given in Section 7. If any concepts seem unclear or ambiguous, the reader is encouraged to check the glossary in [Appendix 1](#).

2 Previous Work

In this section, previous work is discussed. Firstly, some general work on teams is discussed in Section 2.1. After this, as we aim to improve team performance using dynamic task allocation, previous work on team performance is discussed in Section 2.2. One important influence on performance is cognitive task load [9,10]. Previous work on this is discussed in Section 2.3. In Section 2.4 we will discuss previous work on dynamic task allocation, specifically how tasks are reallocated (for example based on cognitive task load). If we want dynamic task allocation to not cause higher task load for a human team member, we should automate the process [5]. Previous work on adaptive automation, the process in which robots in a mixed human-robot team operate under automated dynamic task allocation, is discussed in Section 2.5.

2.1 Teams

Teams are groups consisting of two or more actors that set out to achieve a common goal. Furthermore, to achieve this goal, several different tasks need to be done or different roles need to be fulfilled. These different tasks¹ enforce the actors to coordinate their plans. The coordination process in which the different tasks that a team needs to get done are assigned to the different actors in the team is called task allocation. In human teams, dynamically adjusting the allocation of tasks, facilitated by backing-up behavior², is a critical component of team behavior and helps performance [2].

As technology advanced, the possibility for virtual agents (and robots) to work in teams on complex tasks arose. The area of work associated to this is called multi-agent systems. Multi-agent systems research involves building autonomous agents that can cooperate, coordinate and negotiate with other agents [13]. When a team sets out to work in complex and dynamic environments, they need some general notions of how a team works, so that they can monitor the team's performance and reorganize when necessary [14]. This need for explicit models of teamwork inspired a surge of research.

An early and very influential framework for teamwork was put forward by Cohen & Levesque: the joint-intentions framework [15]. Cohen & Levesque argue that teamwork is work performed by individuals sharing certain mental properties, which leads them to jointly intend to perform a collective action. This joint intention affects and is affected by properties of individual team members, such as individual intentions and individual commitment. The benefit of teamwork, according to Cohen & Levesque, is that a goal can be achieved in a more robust way as the load is shared.

Another influential step in teamwork research is the model introduced by Tambe called STEAM [16]. STEAM is an implemented, general model of teamwork. STEAM uses the notion of Cohen & Levesque's joint intention as a basic building block; teamwork consists of agents building a hierarchy of joint intentions. STEAM supports a team's communication, performance monitoring, and reorganization when necessary, driven by

¹We do not talk about roles, because tasks are the more specific concept as roles can be seen as sets of tasks.

²Backing-up behavior (or back-up behavior) involves helping other team members perform their tasks.

their joint intentions. As the model is general (vs. domain specific) the teamwork behavior supported is flexible, allowing teams to deal with novel and unexpected situations.

The above framework and model explicitly describe how teams work, namely how agents (come to) work together to achieve some goal. They do not specify however, what constitutes a successful team which performs well on achieving their goal¹. In other words, they do not explicitly describe what factors exist that influence the teamwork process in such a way, that the performance on achieving the common goal is affected. The next section aims to find out which factors influence team performance.

2.2 Team Performance

In this section, we will discuss some frameworks that describe how team performance arises. Team performance is a measure of how well the common goal is achieved. This relies not only on individual performance, but also largely on team level concepts like communication between actors and task allocation [2].

Early frameworks describing team performance commonly follow the Input-Process-Output structure, for example the framework proposed by McGrath [17, 18]. McGrath describes three input concepts in his framework: individual level factors (e.g. cognitive ability), group level factors (e.g. cohesiveness) and environmental factors (e.g. resource availability, task difficulty). These three factors are input for the team’s interaction processes and the output concept is team performance. Many empirical studies have been based on this framework, but recently the supposition that Input-Process-Output frameworks like McGrath’s are an insufficient way to represent influences on team performance is gaining support [19]. Several reasons are summarized by Ilgen et al. [19]: Firstly, the term interaction process excludes emergent properties, for example cohesiveness of a team. These sort of emergent properties have an influence on interaction processes, but can also be influenced by them and they are not processes themselves. Secondly, feedback loops should not be excluded. For example, team performance itself can serve as an input for interaction processes, as it can have an influence on these. Thirdly, the Input-Process-Output structure suggests linear progression, but interactions between various inputs and processes or between different processes should also be considered possible.

Kozlowski and DeShon [20] propose a framework for team performance that focuses on adaptive performance. Adaptive performance is the ability to react to changing demands, which is needed in dynamic environments. Their framework can be seen in Figure 1. The concepts Kozlowski and DeShon use are quite similar to the concepts used by McGrath. The antecedents in Kozlowski and DeShon’s framework are almost the same as the input concepts in McGrath’s framework. These antecedents are individual differences, team characteristics, and situational demands. The processes are specified into team regulation and individual regulation. Regulatory processes aim to detect whether there are any discrepancies between expected outcomes and actual outcomes. What sort of relations exists between antecedents, regulatory processes and adaptive performance

¹A team performs well on achieving their goal if, for example, they achieve it efficiently and effectively.

is not specified. For this reason, the framework does not suffer from the same downsides as frameworks following the Input-Process-Output structure. However, the fact that the relations are not detailed is a downside in itself as it makes the framework hard to use as a base for empirical research.

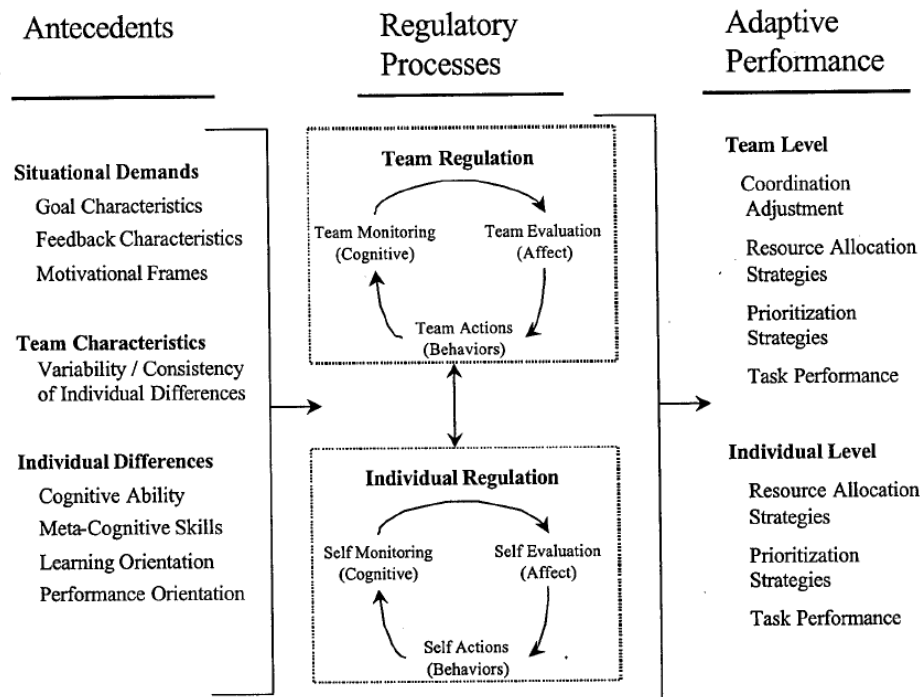


Figure 1: A framework for adaptive team performance, as proposed by Kozlowski [20].

In human teams, dynamic task allocation is triggered when regulatory processes reveal discrepancies between expected outcomes and actual outcomes (either at the team or individual level). Dynamic task allocation aims at solving the discrepancy by reallocating certain tasks to different actors. Dynamic task allocation benefits team performance [2].

A specific discrepancy between expected and actual outcomes that can occur is individual performance degradation. The individual performance of team members has an influence on team performance [2]. To improve team performance, it would be beneficial to prevent individual performance degradation rather than to wait for regulatory processes to detect it.¹ We thus need to find out what circumstances cause the degradation and trigger reallocation of tasks when these circumstances arise.

A vast amount of research has focused on the factors that influence individual performance. Examples of factors include attention, workload, skill, stress, fatigue, lifestyle, personality, age and mood [22]. Another important influence on individual performance for humans is cognitive task load [8].

¹Also, team performance is usually not (yet) measurable in real time [21].

2.3 Cognitive Task Load

An important factor for dynamic task allocation in human teams is cognitive task load [1]. A model of cognitive task load was proposed by Neerincx [23]. The model describes how task characteristics are of influence on individual performance and mental effort. According to Neerincx, cognitive task load can be described as a function over three metrics. Firstly, the percentage of time that a person is occupied by his/her tasks. Secondly, the number of task-set switches which is the number of times that a person has to switch between different tasks. Thirdly, the level of information processing that is needed for the current tasks the person is doing. The level of information processing is a metric based on the skill-rule-knowledge framework described by Rasmussen [24]. Skill-based tasks, such as riding a bike require almost no cognitive resources as they rely on well-learned skills. Rule-based tasks require the application of rules or instructions to accomplish a task and thus impose some cognitive load. Knowledge-based tasks impose the most cognitive load as no specific rules can be followed and thus higher level reasoning is needed to accomplish the task.

The influence of the three cognitive task load metrics on performance is visualized in Figure 2. Based on the three metrics, Neerincx defined four problem regions where performance degradation might occur, as represented by the gray areas in Figure 2. Firstly, when the percentage of time occupied is high but the level of information processing and the number of task-set switches are low problems related to vigilance can occur. Performance degrades because alertness decreases as the task¹ is boring: it continually needs attention, but no cognitively demanding actions are required. Secondly, when all three metrics have a low value underload might occur. Thirdly, when all three metrics have a high value overload might occur. Fourthly, when the percentage of time occupied and the number of task-set switches are high, but the level of information processing is medium or low, cognitive lock-up can occur. Cognitive lock-up is a state in which people are reluctant to switch tasks. Performance could decrease if other tasks need attention but do not get this due to the reluctance to switch.

The longer a person’s cognitive task load is in a problem region, the more negative the effect on performance will be. Typically, vigilance and underload problems occur only after some time, while overload and cognitive lock-up problems can occur even if the cognitive task load has only been in the problem region for a short time [8].

The cognitive task load model of Neerincx has been experimentally validated in the naval domain [23]. Furthermore, the model was used to design interfaces providing cognitive support. These interfaces are shown to increase performance [25]. Lastly, the model was used to predict performance in the naval domain using a classifier. Accuracies of up to 86% were achieved [9, 10].

Since cognitive task load is based on characteristics of the tasks a person is currently doing, it can be influenced by changing these tasks. Also, cognitive task load has an influence on performance. Combining these two facts yields that cognitive task load is a factor that can be used for dynamic task allocation aimed at improving performance.

¹Or a small set of tasks.

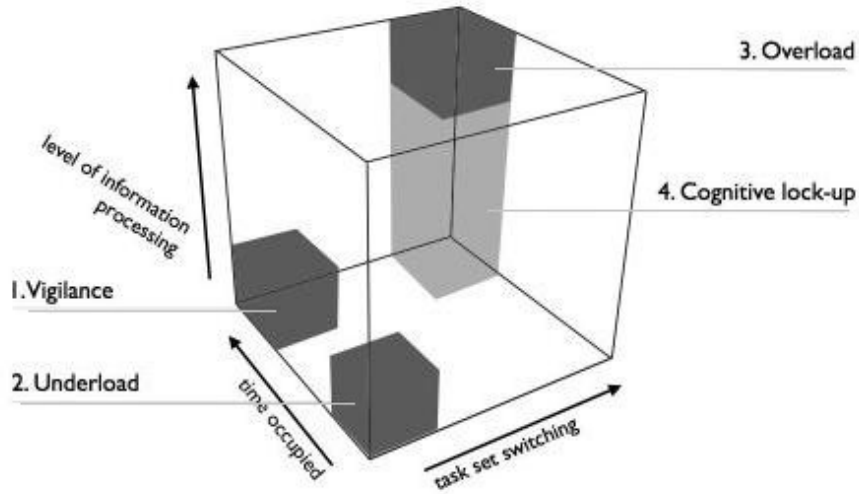


Figure 2: The 3D Cognitive Load Space, as proposed by Neerincx [8], which is a visualization of the influence of the three cognitive task load metrics on performance. Four general problem regions where performance degradation is likely to occur are depicted (shown in gray).

2.4 Dynamic Task Allocation

Dynamic task allocation benefits team performance [2]. Dynamic task allocation can be effectuated in numerous ways. Two of the questions that are of importance in distinguishing different dynamic task allocation strategies are discussed in this section.

Firstly, who has the responsibility to dynamically allocate tasks? This responsibility includes deciding whether a task allocation is needed, and if it is needed, deciding on a reallocation. Responsibility can be distributed (over the whole team), or it can be centralized. Distributed responsibility for dynamic task allocation has the disadvantage that it causes extra workload for (human) team members [5]. Centralized coordination in which the dynamic allocation of tasks is fully automated does not have this problem. Automated centralized coordination implies there is a system, for example a virtual agent, that is the only one that decides when a task allocation is needed and decides on a reallocation of the tasks. A problem with centralized coordination could be that it is unfeasible to implement for very large teams. Also, human team members could have issues if task reallocations are not clearly communicated or if it is unclear why they are made.

The second question concerns what influences exist on the decision for a specific reallocation of tasks. Inagaki [26] argues that a dynamic form of comparison allocation is the best strategy for task allocation. Comparison allocation means tasks are allocated based on capabilities of actors. This as opposed to leftover allocation (automating all tasks that can be automated) or economic allocation (automating all tasks that costs less to automate than hiring a human operator) which do not focus on improving team performance.

How capable an actor is to do a certain task can differ statically, for example due to whether the actor is a human or robot or depending on the actor’s individual properties such as intelligence. It can also differ dynamically, as environmental changes might occur changing the capabilities needed for a task or the state of an actor [26]. Below we provide a summation of the general factors that influence the capability of an actor to do a task (and thus influence task allocation), with a non-extensive list of specific examples used in relevant studies¹ for each general factor. The influences on team performance are usually divided into three categories, as we have seen in Section 2.2: individual level factors, team level factors and environmental factors. Individual level factors are important for task allocation and so are environmental factors. For clarity, we separate task level factors from environmental factors as they are both important, but very different, factors influencing task allocation. Group level factors are an important influence on team performance, but can not directly be used to allocate tasks, as tasks are allocated over individuals (or small sets of individuals).²

Influences on task allocation:

- Individual level factors, which can be both static or dynamic. Static individual factors are cognitive, emotional and physical properties and abilities. Dynamic individual factors are cognitive, emotional and physical states. For example (grouped per study):
 - Skills [27] (cognitive ability)
 - Self-regulation [20] (cognitive/emotional ability)
 - Conscientiousness, agreeableness, extraversion, emotional stability [28] (cognitive/emotional ability)
 - Cognitive task load [8] (cognitive state)
 - Current location, availability (specifically whether the actor is doing a task and if so, the priority of this task) [29] (cognitive/physical state)
 - Current tasks [5] (cognitive/physical state)
 - Experience (quality, quantity, success rate), ability (resources, training, handling, damage), individual state (personality, emotion, fatigue) [30] (cognitive/emotional/physical ability/state)
 - Location, tiredness of the person, specialty [31] (cognitive/physical ability/state)
 - Trust [32] (emotional state)
- Environmental factors, which are usually dynamic. For example (grouped per study):

¹Specific examples are gathered from studies about task allocation in either all human, mixed human-robot or all robot/agent teams.

²If we want to use group levels factors to allocate task, we need to do this indirectly by using the individual level factors that contribute to these group level factors.

- Weather conditions, noise [33]
- Traverse-ability [34]
- Task factors, which are usually dynamic. For example (grouped per study):
 - Resource requirement of task [30]
 - Difficulty of task (Level of information processing, time occupied) [8]
 - Required skills for task [27]
 - Location of task, priority of task [29]
 - Location of task, difficulty of task, specialty needed for task, duration of task [31]
 - Performance feedback [19]

Note that for some of the specific examples mentioned above, the influence on task allocation is only clear when you see them in relation to other factors. For example, the location of an actor is only of influence if we know its relation to e.g. the location of the task (since we can then calculate this distance the operator will need to travel to get to the task). In this same manner, a very important influence on task allocation is the cost caused by the reallocation of tasks [5]. These so called coordination costs can be derived by comparing the current task allocation to the new task allocation.

In most studies using multiple factors, all of the factors and their interrelations are brought together in some sort of weighing function.¹ The weight assigned to each factor specifies its importance (relative to the other factors) to the capability of the actor to do a task.

Recent years have shown a surge of studies about automated dynamic task allocation in mixed human-robot teams.

2.5 Adaptive Automation

Traditionally, tasks in mixed human-robot teams are allocated to either a human or a robot. Using capability comparison allocation, these allocations are often based on static lists of human capabilities versus robot capabilities (MABA-MABA²) like the famous Fitts list [36]. This way of allocating tasks has two main problems [26], which we will discuss below.

Firstly, allocating a task either fully to a human (robot has no autonomy) or fully to a robot (robot has full autonomy) is overly coarse. Many different levels of autonomy in between these two extremes are possible, where the human and robot share control over the task [37]. Furthermore, a level of autonomy for a robot can be decided on separately for each subtask [6, 38].

¹For an example, see the weighing function in [30, 35].

²What *men are better at* and what *machines are better at*.

Secondly, static task allocation is insufficient for dynamic and complex environments, as capabilities needed for a task could change due to the changing environment (as before said in Section 2.4). To address these issues, adaptive task allocation was proposed. Adaptive automation is the process in which a robot operates under adaptive task allocation.

Adaptive automation in the context of single human-single robot teams has received a lot of attention in the last few years. Numerous studies have shown the positive effects of adaptive automation in single human-single robot teams. These positive effects of adaptive automation (level of autonomy of robot changes automatically) were shown to exist when comparing it to either adaptable automation (level of autonomy of robot changes when human invokes this) or static automation (level of autonomy of robot does not change) at different levels of autonomy.

The positive effects of adaptive automation include improved performance [7, 39–44], enhanced situation awareness [7, 45] and reduced cognitive workload [7, 45].¹ Adaptive automation outside single human-single robot context has received much less attention. A few studies have looked at adaptive automation in the context of single human-multiple robot teams [46–48]. In these studies however, only the level of autonomy of a single robot [46] or of a system separate from the robots [47, 48] on a single task was adapted. To our knowledge, no studies have looked into adaptive automation in multiple human-single robot teams.

In the majority of the research on adaptive automation, measures are taken on the individual level instead of team level. For example, performance measurements often compass only operator performance, excluding automation performance and group level factors while these also influence team performance.

Different techniques for triggering reallocation are possible [41]. Techniques can be either performance-based [7, 43, 44], psychophysiological measure-based [39, 45], operator cognition-based [44, 49], environment-based [42, 50] or hybrid techniques [40, 41].² Performance based techniques are used quite often, using either performance on the main or a secondary task. The downside of performance-based measurements is that not all tasks allow for real-time performance measurement [21]. Techniques using psychophysiological measures are not suitable for many domains, as the measuring devices often restrict the subject’s movements. Environment based techniques in isolation fail to capture changing states of team members that could point to a need for task allocations. As De Greef [41] argues, hybrid techniques are more robust as multiple independent factors can be used. If multiple independent factors (as opposed to a single factor) point to a need for task reallocation, then this need is more likely to be real or urgent. Only a limited amount of studies so far have used hybrid techniques [41].

Most of the studies referenced in this section used one task with a varying level of autonomy (with usually only two, but sometimes up to five different levels). The trigger for reallocation signaled either a need for the level of autonomy of the robot to increase on this task (e.g. when the workload of the operator is too high), or to decrease (e.g.

¹The list of studies referenced after each effect is non-extensive.

²The list of studies referenced after each technique is non-extensive.

workload operator too low). In response to this, the level of autonomy of the automation was either increased or decreased one level. This approach cannot be used when multiple tasks can have separate levels of autonomy. In this case, we need a way of deciding on which tasks the level of autonomy should be changed. An overarching model/framework could help in devising ways for deciding on specific levels of autonomy on the different tasks.

3 Framework for Dynamic Task Allocation

In this section, a framework is presented for dynamic task allocation that aims to improve team performance. The framework describes the important concepts that influence team performance and can be used to dynamically allocate tasks. It is designed to work in complex and dynamic environments. The framework should be able to serve as a guidance for devising models that dynamically allocate tasks and hereby improve team performance. It does not make assumptions about the team composition (e.g. human or robot members) and therefore does not assume a team member can only do a single task at a time. As our framework should work in dynamic environments, and teams could include both humans and robots, we use a dynamic form of comparison allocation (as argued in Section 2.5). We do not want to cause extra workload for human team members. To achieve this, we use automated centralized coordination (as argued in Section 2.4).

Some topics are, although important, outside the scope of the research. We do not focus on planning tasks that are expected to start some time in the future, but only on instantaneous assignment of tasks to actors. Furthermore, we do not focus on task decomposition and cooperation, we assume tasks are already decomposed into subtasks that can be executed by a single agent.

This section is structured as follows. Firstly we detail how our proposed framework relates to and builds on previous work in Section 3.1. An overview of the components of the proposed framework is given in Section 3.2, followed by a formal description of the different components and their relations in Section 3.3.

3.1 Relation to Previous Work

A starting point for our framework is the taxonomy and analysis of dynamic task allocation problems in the multi-robot domain by Gerkey and Mataric [51]. Gerkey and Mataric's work is very relevant as it describes the important concepts in dynamic task allocation problems. In this section, firstly Gerkey's¹ taxonomy and its relation to the proposed framework is detailed. After this, Gerkey's analysis of dynamic task allocation problems is explained and we describe how the proposed framework builds on his work.

Gerkey [51] provides a taxonomy that characterizes dynamic task allocation problems in the multi-robot domain along three axes. A visualization of this taxonomy is shown in Figure 3. Firstly Gerkey distinguishes between single-task robots and multi-task robots, describing whether a robot can execute one or more tasks simultaneously. Secondly, he distinguishes between tasks requiring a single robot and tasks requiring multiple robots at a time to be executed: single-robot tasks versus multi-robot tasks. Thirdly, a distinction is made based on time: instantaneous task assignment (we can only allocate currently active tasks) versus time-extended assignment (we can plan tasks that are expected to start some time in the future).

¹From now on, as shorthand, we use only Gerkey's name when referring to Gerkey and Mataric's work on the taxonomy and analysis of dynamic task allocation problems in the multi-robot domain [51].

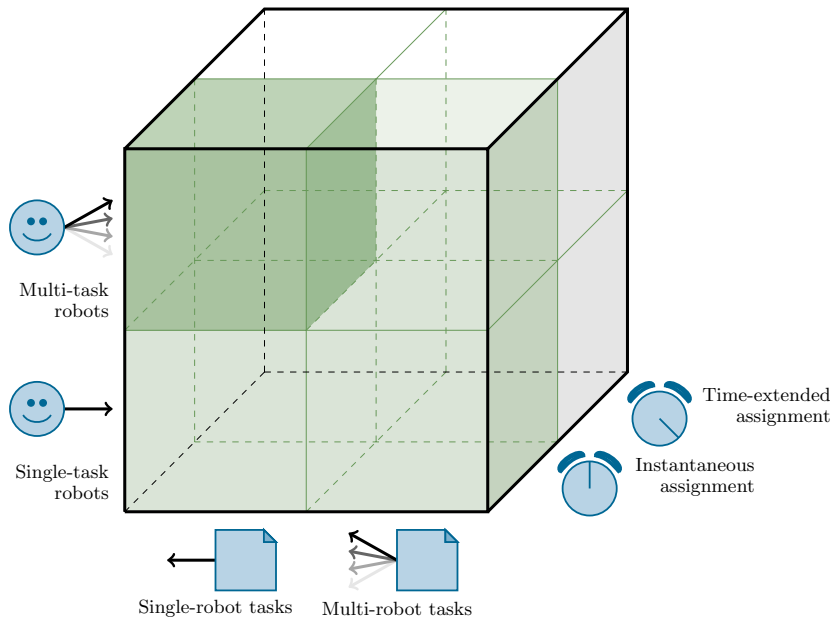


Figure 3: Overview of Gerkey’s taxonomy [51] of dynamic task allocation problems in multi-robot systems. The area highlighted in dark green shows the focus of the proposed framework. The area highlighted in lighter green shows problems that are not focused on, but that can be modeled within the proposed framework. We do not actually exclude the possibility of multi-robot tasks, but assume these tasks are decomposed into multiple single-robot tasks. The same holds for single-task robots, they are not excluded but can be modeled as multi-task actors with a restriction.

If we want to categorize the problems our framework should cover according to Gerkey’s taxonomy, we must first generalize his notions. Instead of talking only about robots, we use the more general term actor. We can now categorize the problems our framework should cover as multiple-task actor, single-actor task, instantaneous assignment problems (as visualized in Figure 3). As we want to include the possibility of having human (or sophisticated robotic) actors in the team, we should not restrict ourselves to single-task actors, as human (and sophisticated robotic) actors can do multiple tasks at a time. Furthermore, single-task actors can be seen as a special case of multi-task actors. Our framework will focus on single-actor tasks; multi-actor tasks are only possible if defined as separate subtasks for each actor. The problem of decomposing multi-actor tasks into single-actor tasks is outside the scope of this research. We restrict ourselves to instantaneous assignment of tasks as time-extended assignment includes scheduling or planning and is outside the scope of this research.

The proposed framework for dynamic task allocation is based on the formal analysis of (multiple-task actor, single-actor task, instantaneous assignment) dynamic task allocation problems proposed by Gerkey [51]. A high level overview of this analysis is seen in Figure 4. Gerkey proposes the following steps to solve the dynamic task allocation problem: Firstly, possible role assignments and their evaluation can be generated from context information. Role assignments are a combination of a robot and a set of tasks

this robot could execute. These role assignments can be evaluated using context information relevant to how well the robot is able to execute the set of tasks. Secondly, we can construct possible task allocations by combining role assignments. A task allocation is a set of role assignments for all robots. The evaluation of a task allocation is a combination of the evaluation of all role assignments it contains. The third step is to choose one of these possible task allocations.

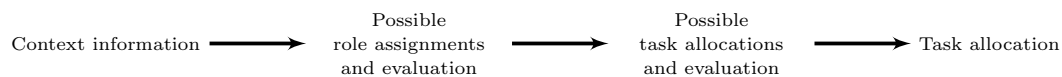
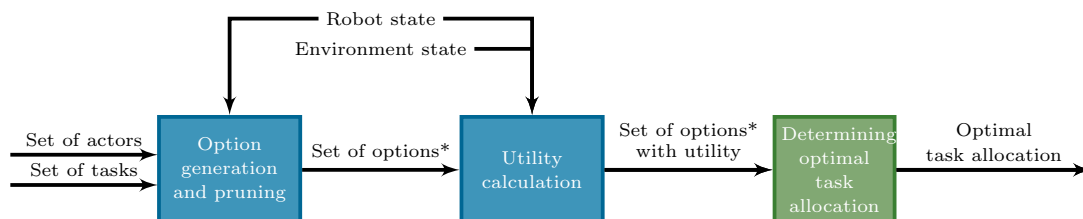


Figure 4: A high level overview of Gerkey’s analysis [51] of dynamic task allocation problems for multi-task robots, single-robot tasks and instantaneous assignment.

How Gerkey details his approach can be seen in the more technical overview in Figure 5. Gerkey argues that dynamic task allocation can be seen as optimizing a utility function. Firstly, all options of allocations of sets of tasks to a robot (*role assignments*) should be generated and this set of options should be pruned. Secondly, a function that assigns a utility value to the options (*evaluation*) should be defined. An optimization algorithm can then be applied, which finds the collection of options which has the highest utility and allocates every task to a robot. Gerkey defines the utility of a task allocation (a set of options/role assignments) as the sum of the utilities of all options/role assignments it contains.



*Options are combinations of an actor and a set of tasks (allocated to that actor), i.e. a role assignment.

Figure 5: Gerkey’s analysis [51] of dynamic task allocation problems for multi-task robots, single-robot tasks and instantaneous assignment. Colored boxes denote processes, arrows represent flow of information. Gerkey focuses on the process of optimization (green box) and less on the processes of option generation and pruning and calculating the utility (blue boxes).

Gerkey defines a utility measure that, based on a single robot and single task (a *robot-task pair*), gives back a utility value. This utility measure gives back the value zero if the robot is unable to execute the task, otherwise it returns the expected quality of the robot executing the task minus the expected cost. Gerkey does not specify the quality and cost functions, but he does state that all relevant aspects of the state of the robots and their environment should be included in the calculations. What the relevant aspects are, or how they should be included in the calculations is not specified.

Gerkey makes some assumptions about interrelations between utilities measures. He assumes that the utility measure of a robot-task pair is not influenced by any other

tasks the robot might be allocated to. This implies the overall utility of a set of tasks allocated to a robot (a *robot-task set pair*) is an addition of all utility measures for the separate robot-task pairs for each task included in the task set. Gerkey assumes that the utility measure of a robot-task pair *is* influenced by which other robots are allocated to the same task. This implies the overall utility of a set of robots allocated to a task *cannot* be seen as an addition of all utility measures for separate robot-task pairs (for each robot in the set of robots).

Gerkey’s analysis has some limitations. These limitations include the assumption that the utility of a robot-task pair is not influenced by other tasks the robot might be doing. Also, Gerkey’s analysis does not include mixed human-robot teams. More importantly, Gerkey casts multi-robot task allocation problems to instances of optimization problems, but fails to describe some important steps that are needed to realize this casting. Firstly, the problem of how to generate feasible options of allocation of tasks to robots is not discussed. Secondly, the utility functions (quality and cost) are underspecified as to what aspects they should take into account and how.

How does the proposed framework relate to Gerkey’s analysis? The structure of Gerkey’s analysis is used as a base for our framework. This includes some sort of utility measure that needs to be optimized. The proposed framework extends Gerkey’s analysis to fit our requirements. We specifically address the issues of option generation and utility calculation. Our framework explicates that the utility measure is built up out of different factors. These factors could be related to the actor, the environment, the task and combinations of these three. Independence of utilities is not assumed, the utility of actor-task pair can depend on which other tasks actor is allocated to.

Once we have dealt with the issues of option generation and utility generation, we can cast the task allocation problem to a known optimization problem like Gerkey suggests. The task allocation problem for multiple-task actor, single-actor task and instantaneous assignment can be cast as an instance of the well-known set partitioning problem (SPP) [51]. It involves splitting the set of tasks into actor-specific task sets.¹

Definition 1. (Set Partitioning Problem (SPP)) Given a finite set E , a family F of acceptable subsets of E , and a utility function $u : F \rightarrow \mathbb{R}_+$, find a maximum-utility family X of elements in F such that X is a partition of E . X is a partition of E if and only if the elements of X are mutually disjoint² ($\forall y, z \in X, y \neq z : y \cap z = \emptyset$) and their union is E ($\bigcup_{x \in X} x = E$). [51]

Although the SPP is strongly NP-hard, it has been studied extensively and many heuristic algorithms that give good approximations have been developed [51]. We can

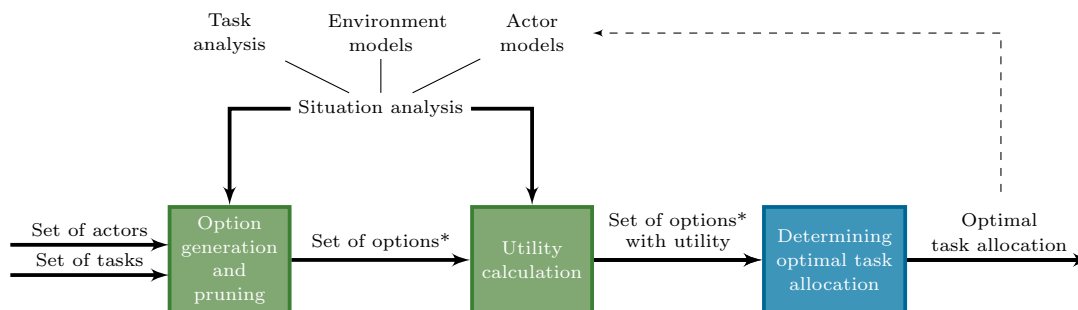
¹The problem of coalition formation is the problem of splitting the set of actors into task-specific coalitions. This is essentially the same problem with the concept of actors and tasks switched. Coalition formation has been extensively studied, for an overview see Gerkey’s discussion for single-task robots, multi-robot task and instantaneous assignment [51].

²If we drop the restriction that all elements of X should be mutually disjoint, we have defined the Set Covering Problem (SCP), of which the task allocation problem with multiple-actor tasks (and multiple-task actors and instantaneous assignment) can be cast as an instance.

cast the task allocation problem for multiple-task actor, single-actor task and instantaneous assignment to an instance of the SPP in the following way: We define E to be the union of the set of actors and the set of tasks, F the set of all feasible actor-task set pairs (sets including a single actor and zero or more tasks that are allocated to this actor) and u a utility estimate function for each actor-task set pair. A more detailed description of this casting can be found in [Appendix 2](#). As we cast the task allocation problem to an instance of the SPP, we introduce the assumption that all tasks should be allocated to an actor, as follows from the definition of a partitioning. In [Section 3.2](#), we relieve this assumption.

3.2 Overview of the Framework

An overview of the components of the proposed framework is shown in [Figure 6](#) and is explained in this section.



*Options are combinations of an actor and a set of tasks (allocated to that actor), i.e. a role assignment.

Figure 6: Overview of our framework. Colored boxes denote processes, arrows represent flow of information. Opposite to Gerkey’s focus (see [Figure 5](#)), we focus on the process of pruning generated options and calculating utility of options (green boxes) and less on the process of optimization (blue box).

The three categories of factors that influence task allocation (individual, environmental en task factors), as described in [Section 2.4](#), are represented by three input concepts in our framework. Firstly individual factors are represented by actor models. Actor models are functions that describe for each actor, their relevant cognitive, emotional and physical ability and state associated with a certain point in time. Abilities are static¹, so will be the same at every point in time. Actor abilities are for example IQ, personality traits and skills. The dynamic counterpart of actor abilities are actor states, for example emotion, location and fatigue. Secondly, environmental factors are represented by environment models. Environment models are functions that describe states and properties of the environment, that are dependent on the specific location, at a certain point in time. Examples of environmental states are resource availability, weather conditions and noise levels. Thirdly, task factors are represented by task analysis. Task analysis contains functions from a specific task to its properties and states at a certain

¹Note that the dividing line between static and dynamic is dependent on the duration of the teamwork the framework is applied to.

point in time. Examples of task properties/state include location, required skills and resource requirement.¹

Some factors influencing task allocation arise from interactions between (two or three of) the three concepts we introduced above (Section 2.4). These interaction factors are represented by the concept of situation analysis in our framework. Some examples are:

- The distance from an actor to a task, a function that is influenced by both actor location (actor state) and task location (task state/property).
- The physical ability of an actor to do a task, a function that is influenced by the physical abilities of the actor and the physical requirements of the task.
- The possibility of executing a task at a certain location, a function that is influenced by the resource requirements of the task and the available resources at the location.

As a very important influence on task allocation is the cost caused by the reallocation of tasks [5](Section 2.4). Therefore our framework includes a feedback loop for the task allocation itself. The current task allocation itself thus is an actor state.

Now that we have discussed influences on task allocation, the next step is to generate possible options of allocations and choose the best option. To find the best allocation of tasks, we need to generate the possible options of actor-task set combinations and use this as input for applying a SPP solving algorithm [51]. As mentioned by Gerkey [51] pruning the set of feasible actor-task set pairs is needed as often the amount of options is very large. Pruning the set of options makes the search for the optimal option² faster. Inspired by other work [35], we distinguish between two different kinds of factors that influence task allocations. The first kind of factors are restrictive, the second kind are preference factors. Restrictive factors put restrictions on the possible actor-task set pairs and thus prune the set of possibilities. For example, an actor might lack the proper sensors to execute a task. This could be the case if our team includes a stationary robot and a patrolling task. Another example is two tasks that cannot be done by the same actor, for example if the two tasks are spatially located far from each other. Preference factors do not prune the set of possible actor-task set pairs, but give some indication as to how well the task set can be executed by the actor. For example, if an actor has been assigned a single, but difficult task, he might do better on this task than if he has also been assigned to do multiple difficult other tasks simultaneously. Another example is that a very strong actor might be better at a task involving carrying heavy boxes, while a physically weaker actor could also be able to carry the boxes, but most probably not as fast as the strong actor.

The next step in our process is defining a utility function that maps remaining possible actor-task set combinations to a utility value. We define this function to be some function (Section 2.4) over all the preference factors. With this utility function

¹Whether these examples are tasks properties (static) or tasks states (dynamic) depends heavily on the specific task.

²Or even an approximation of the optimal option.

and the set of possible actor-task set pairs, we can use a SPP solving algorithm to arrive at the best task allocation for a specific time.

As mentioned in Section 3.1, solving the task allocation problem by casting it to an instance of the SPP introduces the assumption that all tasks need to be allocated to an actor. This excludes the applicability of the framework to scenarios where it might not be possible to allocate all tasks, or scenarios where a team might perform better despite not executing one of the tasks. We relieve this assumption by introducing a placeholder for tasks that are not executed. If we define this placeholder to be a dummy actor and make him a member of the set of actors that comprises our team, tasks can be allocated to him. All tasks allocated to the dummy actor in a certain task allocation are not executed since the set partitioning used to make a task allocation enforces tasks are allocated to precisely one actor. We can now model mandatory tasks by defining a restrictive factor that prunes role assignments that allocate a mandatory task to the dummy actor. Also, the costs of not executing certain tasks can be easily modeled using a preference factor, since the set of tasks that are not executed is explicit as it is the set of tasks assigned to the dummy actor.

3.3 Formalization of the Framework

To be a guidance for devising models, the proposed framework should make explicit what sort of relationships exist between concepts (as stated in Section 2.2). Therefore the following section provides a formalization of the components of the framework, as seen in the previous section (3.2). As an example, we construct a simple formalization of the book reallocation task that was introduced in Section 1. Parts of this example formalization can be found throughout the section, framed in boxes as Example 1 that contains a recap of the book reallocation task.

Imagine two people that need to relocate a lot of books. The books need to be put into boxes per subject, the subject should be written on a label, and then the boxes should be loaded into a car. One of the persons (Albert) is stronger and thus more suited to carry boxes, the other (Ben) will pack and label the boxes. After a while of carrying and packing books, Albert is getting very tired and needs to sit down (change in state of team member). Ben finds some books on geography, which he knows very little about (dynamic environment) and has trouble sorting and labeling them. Dynamic allocation of tasks would now allow Albert (who happens to be a geographer) and Ben (who is well rested) to switch tasks, which will improve team performance (number of books getting boxed, correctly labeled and loaded in to the car).

Example 1: A recap of the book reallocation task.

3.3.1 Environment Models

The first component of our framework we discuss are the environment models. Environments models describe environmental factors that could have an influence on the reallocation of tasks. Some examples of possible relevant environmental concepts are weather conditions, traversability of terrain, geographical location, resource availability and noise. We define a set of environments $\mathcal{E} = \{E_1, E_2, \dots, E_n\}$ and a set of environmental state concepts $\mathcal{S}^{\mathcal{E}} = \{S_1, S_2, \dots, S_m\}$. Environment models are defined as functions from environments at a certain time to values of environmental states: $f_{\mathcal{E}}(S_y, E_x, t)$ ¹ where t stands for time. Note that we do not define the set of environments to be an input factor. Environments are is always related to either the actor or the task (in which environment is the actor or task situated) and environments are thus indirect input via the actor models and/or task analysis.²

For our example, a simple formalization of the book reallocation task, the environment models could like this:

$$\mathcal{E} = \{\text{Utrecht}\}$$

$$\mathcal{S}^{\mathcal{E}} = \{\text{Temperature}\}$$

We model the book reallocation task at two different moments, the start at $t = 0$ and 30 minutes after this at $t = 30$. For both these points in time, the values for the environmental factor ‘Temperature’ are given below:

$$f_{\mathcal{E}}(\text{Temperature}, \text{Utrecht}, 0) = 26^{\circ}\text{C}$$

$$f_{\mathcal{E}}(\text{Temperature}, \text{Utrecht}, 30) = 28^{\circ}\text{C}$$

The task allocation in the book reallocation task is not actually influenced by environmental factors, as there is only one environment. We provide the environmental factor ‘Temperature’ here just as an example of how such a factor would be represented in our framework.

Example 2: Environment models.

3.3.2 Actor Models

Many relevant research defines a collection of states or abilities that an actor has, for example the vector of actor abilities that Shehory and Kraus use in [52]. Much like Shehory and Kraus, we define a set of actor states/abilities. We define actor models as functions from situated actors to values of actor states. Situated actors (A_x) are actors ($Actor_x$) in relation to the environment (E_y) that they are in, we formalize this as a pair: $A_x = \langle Actor_x, E_y \rangle$.

¹Note that the type of output of this function is underdefined, and can be different for each environmental state concept. The environmental state concept (first argument of function) enforces the type of output.

²This is because the problem of determining in which environment (at which location) an actor should do some task is outside of the scope of this research.

To formalize the notion of actor models, we must first formalize some concepts. Firstly, the set of situated actors of which the team is comprised at time t : $\mathcal{A}(t) = \{A_1, A_2, \dots, A_n\}$. Secondly, the set of actor state concepts, which is the set of cognitive, emotional and physical abilities and states we deem relevant of the actors in the team: $\mathcal{S}^{\mathcal{A}} = \{S_1, S_2, \dots, S_m\}$. We can now define functions from situated actors to values of actor states in the following way: $f_{\mathcal{A}}(S_y, A_x, t)$ ¹ where t stands for time. Static properties can be defined as a special case of states, in which case the value of t has no influence on the output of $f_{\mathcal{A}}$. From now on, when we use the concept *actor*, this always refers to a situated actor, thus a pair of an actor and his environment.

¹Note that the type of output of this function is underdefined, and can be different for each actor state concept. The actor state concept (first argument of function) enforces the type of output.

For our example, a simple formalization of the book reallocation task, the actor models could look like the following (we do not use a dummy actor, as we want all tasks to be allocated):

$$\begin{aligned}\mathcal{A}(0) &= \{\langle \text{Albert, Utrecht} \rangle, \langle \text{Ben, Utrecht} \rangle\} \\ \mathcal{A}(30) &= \{\langle \text{Albert, Utrecht} \rangle, \langle \text{Ben, Utrecht} \rangle\} \\ \mathcal{S}^{\mathcal{A}} &= \{\text{Geographical knowledge, Physical condition}\}\end{aligned}$$

The level of an actor's geographical knowledge is modeled as a static property. This is because during the book reallocation task, the level of knowledge will not change.

$$\begin{aligned}f_{\mathcal{A}}(\text{Geographical knowledge}, \langle \text{Albert}, * \rangle, *) &= 0.8 \\ f_{\mathcal{A}}(\text{Geographical knowledge}, \langle \text{Ben}, * \rangle, *) &= 0.1\end{aligned}$$

where the asterisk denotes a wildcard, meaning the environment and the value of t have no influence on the output of $f_{\mathcal{A}}(\text{Geographical knowledge}, \langle \text{Actor}_x, E_y \rangle, t)$.

By the physical condition of an actor, we mean the potential physical energy an actor has to do physical work. The physical condition is modeled as a state. This is because an actor's physical condition might decrease if he is carrying heavy books for some time. Also, environmental factors might matter. For example: if the temperature is very high in the actors environment, his potential physical energy might decrease. The function that calculates the physical condition could look like the following:

$$f_{\mathcal{A}}(\text{Physical condition}, \langle \text{Actor}_x, E_y \rangle, t) = \begin{cases} \begin{cases} f_{\text{fitness}}(\text{Actor}_x) \\ -f_{\text{used_energy}}(\text{Actor}_x, t) \end{cases} & \text{if } f_{\mathcal{E}}(\text{Temperature}, E_y, t) < 35^{\circ}\text{C} \\ \begin{cases} f_{\text{fitness}}(\text{Actor}_x) \\ -f_{\text{used_energy}}(\text{Actor}_x, t) * 0.5 \end{cases} & \text{if } 45^{\circ}\text{C} > f_{\mathcal{E}}(\text{Temperature}, E_y, t) > 35^{\circ}\text{C} \\ 0 & \text{otherwise} \end{cases}$$

where f_{fitness} is some function describing the fitness level of each actor and $f_{\text{used_energy}}$ is some function describing how much physical energy an actor has not yet regained at a certain point in time.

For now, we will not further detail the function, but just give the output values it might produce:

$$\begin{aligned}f_{\mathcal{A}}(\text{Physical condition}, \langle \text{Albert, Utrecht} \rangle, 0) &= 0.9 \\ f_{\mathcal{A}}(\text{Physical condition}, \langle \text{Albert, Utrecht} \rangle, 30) &= 0.4 \\ f_{\mathcal{A}}(\text{Physical condition}, \langle \text{Ben, Utrecht} \rangle, 0) &= 0.6 \\ f_{\mathcal{A}}(\text{Physical condition}, \langle \text{Ben, Utrecht} \rangle, 30) &= 0.6\end{aligned}$$

Example 3: Actor models.

3.3.3 Task Analysis

The concept of task analysis is very similar to that of actor models. Situated tasks (T_x) are tasks ($Task_x$) in relation to the environment (E_y) that they are in, we formalize this as a pair: $T_x = \langle Task_x, E_y \rangle$. We define the set of situated tasks at time t : $\mathcal{T}(t) = \{T_1, T_2, \dots, T_n\}$ ¹ and the set of task state concepts $\mathcal{S}^{\mathcal{T}} = \{S_1, S_2, \dots, S_m\}$ ². Task models are defined as functions from situated tasks to values of task states: $f_{\mathcal{T}}(S_y, T_x, t)$ ³ where t stands for time. From now on, when we use the concept *task*, this always refers to a situated task, thus a pair of a task and its environment.

For the simple formalization of the book reallocation task, the task analysis could look like the following:

$$\begin{aligned} \mathcal{T}(\ast) &= \{\langle \text{Box books, Utrecht} \rangle, \langle \text{Carry boxes, Utrecht} \rangle\} \\ \mathcal{S}^{\mathcal{T}} &= \{\text{Knowledge needed, Physical demand, Incompatible tasks}\} \\ f_{\mathcal{T}}(\text{Knowledge needed}, \langle \text{Box books}, \ast \rangle, 0) &= \{\} \\ f_{\mathcal{T}}(\text{Knowledge needed}, \langle \text{Box books}, \ast \rangle, 30) &= \{\text{Geographical knowledge}\} \\ f_{\mathcal{T}}(\text{Knowledge needed}, \langle \text{Carry boxes}, \ast \rangle, 0) &= \{\} \\ f_{\mathcal{T}}(\text{Knowledge needed}, \langle \text{Carry boxes}, \ast \rangle, 30) &= \{\} \\ f_{\mathcal{T}}(\text{Physical demand}, \langle \text{Box books}, \ast \rangle, \ast) &= 0.1 \\ f_{\mathcal{T}}(\text{Physical demand}, \langle \text{Carry boxes}, \ast \rangle, \ast) &= 0.8 \\ f_{\mathcal{T}}(\text{Incompatible tasks}, \langle \text{Box books}, \ast \rangle, \ast) &= \{\langle \text{Carry boxes}, \ast \rangle\} \\ f_{\mathcal{T}}(\text{Incompatible tasks}, \langle \text{Carry boxes}, \ast \rangle, \ast) &= \{\langle \text{Box books}, \ast \rangle\} \end{aligned}$$

Example 4: Task analysis.

3.3.4 Situation Analysis

Some factors arise from combinations of actor, environment and/or task states, we call these factors situation states. The set of situation state concepts is defined as $\mathcal{S}^{\mathcal{U}} = \{S_1, S_2, \dots, S_m\}$. Situation analysis contains functions from a role assignment (a pair containing an actor and a task set for this actor) to values of situation state concepts: $f_{\mathcal{U}}(S_x, \langle A_y, \check{\mathcal{T}} \rangle, t)$ where t is time, $\check{\mathcal{T}} \subseteq \mathcal{T}(t)$ and $S_x \in \mathcal{S}^{\mathcal{U}}$. Situation analysis functions are often defined using functions from actor models and/or task analysis. Situation analysis functions can be quite straightforward, for example checking whether an agent possesses all capabilities (actor property) needed for a task (task property), as done by Shehory and Kraus [52]. But functions can also be rather complex, for example the fuzzy

¹Note that, when using this notation, it is easy to model multiple tasks that are identical expect for the environment they are in. For example a task such as ‘Explore Area’, set in a building could be modeled as: $\langle \text{Explore Area, Room 1} \rangle, \langle \text{Explore Area, Room 2} \rangle, \dots$

²The set of task state concepts is much like the vector of needed abilities for a task in e.g. [52], only more general.

³Note that the type of output of this function is underdefined, and can be different for each task state concept. The task state concept (first argument of function) enforces the type of output.

logic-based functions used by Tsalatsanis et al. [35]. Note that we defined situation state functions to take a single actor with a task set instead of a single task as arguments, this is because we do not assume independent utilities (as explained in Section 3.1).

Continuing our simple formalization of the book reallocation task, the situation analysis could look like the following:

$$\mathcal{S}^U = \{\text{Physical match, Knowledge match, No incompatible tasks}\}$$

The situation state concept ‘No incompatible tasks’ is true if and only if no two tasks in a given set of tasks are incompatible:

$$f_U(\text{No incompatible tasks}, \langle A_v, \check{T} \rangle, t) = \forall x, y : (T_x \in \check{T} \wedge T_y \in \check{T}) \rightarrow \neg(T_x \in f_{\mathcal{T}}(\text{Incompatible tasks}, T_y, t))$$

Physical match is defined as the actor state ‘physical condition’ minus the sum of the task states ‘physical demand’ with a maximum value of zero.

$$f_U(\text{Physical match}, \langle A_y, \check{T} \rangle, t) = \max(0, f_{\mathcal{A}}(\text{Physical condition}, A_y, t) - \sum_{T_v \in \check{T}} f_{\mathcal{T}}(\text{Physical demand}, T_v, t))$$

Knowledge match is defined as the average value of all actor states concerning relevant knowledge levels, where relevance is defined by the ‘Knowledge needed’ state of the task:

$$f_U(\text{Knowledge match}, \langle A_y, \check{T} \rangle, t) = \frac{\sum_{T_v \in \check{T}} f_{know}(\langle A_y, T_v \rangle, t)}{|\check{T}|}$$

where

$$f_{know}(\langle A_y, T_v \rangle, t) = \begin{cases} 0 & \text{if } n = 0 \\ \frac{\sum_{S_w \in f_{\mathcal{T}}(\text{Knowledge needed}, T_v, t)} f_{\mathcal{A}}(S_w, A_y, t)}{n} & \text{otherwise} \end{cases}$$

where n is the number of relevant knowledge domains:

$$n = |f_{\mathcal{T}}(\text{Knowledge needed}, T_v, t)|.$$

Example 5: Situation analysis.

3.3.5 Option Generation and Weighing

Situation state concepts should be defined for all factors influencing the task allocation. This because we need to estimate the utility of role assignments (pairs of an actor and a set of tasks), instead of estimating the utility of e.g. a single actor or a single task.

From now on, we will refer to situation state concepts as *state concepts*, stressing that for all other relevant (actor/task/environment) concepts a situation state concept exists that maps it to a utility estimation for a role assignment.

As explained in Section 3.2 we distinguish between two types of factors influencing task allocation, thus two types of state concepts. Firstly, the set of all state concepts that put restrictions on possible task allocations, which is a subset of all state concepts: $\mathcal{S}_{\mathcal{R}} \subseteq \mathcal{S}^{\mathcal{U}}$. Secondly, the set of all state concepts that put preference orderings on possible task allocations, these correspond to utility measures and are a subset of the state concepts that are not restrictive: $\mathcal{S}_{\mathcal{P}} \subseteq \mathcal{S}^{\mathcal{U}} \setminus \mathcal{S}_{\mathcal{R}}$

Possible options of actor-task set combinations at time t are generated to be used as input for applying a SPP solving algorithm [51]. These options are role assignments, modeled as pairs including a single actor and a task set for this actor: $O^i = \langle A_x, \check{\mathcal{T}} \rangle$ where $A_x \in \mathcal{A}$ and $\check{\mathcal{T}} \subseteq \mathcal{T}(t)$.

For our book relocating example, all available actor-task set options are given below. This set is the same for $t = 0$ and $t = 30$. Since our environment is always the same in our example, from now on we will write just the name of the actor or task ($Actor_x$ or $Task_x$) in stead of the situated actor or task pair ($A_x = \langle Actor_x, E_y \rangle$ or $T_x = \langle Task_x, E_y \rangle$).

- $O^1 = \langle \text{Albert}, \emptyset \rangle,$
- $O^2 = \langle \text{Albert}, \{\text{Box books}\} \rangle,$
- $O^3 = \langle \text{Albert}, \{\text{Box books}, \text{Carry boxes}\} \rangle,$
- $O^4 = \langle \text{Albert}, \{\text{Carry boxes}\} \rangle,$
- $O^5 = \langle \text{Ben}, \emptyset \rangle,$
- $O^6 = \langle \text{Ben}, \{\text{Box books}\} \rangle,$
- $O^7 = \langle \text{Ben}, \{\text{Box books}, \text{Carry boxes}\} \rangle,$
- $O^8 = \langle \text{Ben}, \{\text{Carry boxes}\} \rangle$

Example 6: Option generation.

We use the restricting state concepts to prune the set of available actor-task set options. Restricting state concepts should always be defined in such a way, that they return false if and only if an option should be pruned (and true otherwise). Only options for which all restricting state concepts return true are not pruned. Options for which the following property holds (at least one of the restricting state concepts returns false) are pruned:

$$\exists S \in \mathcal{S}_{\mathcal{R}} : \neg fu(S, O^i, t)$$

For our book relocating example, the pruning step could be the following:
 $\mathcal{S}_{\mathcal{R}} = \{\text{No incompatible tasks}\}$

The remaining possible actor-task set options after pruning are (this set is the same for $t = 0$ and $t = 30$):

$$\begin{aligned} O^1 &= \langle \text{Albert}, \emptyset \rangle, \\ O^2 &= \langle \text{Albert}, \{\text{Box books}\} \rangle, \\ O^4 &= \langle \text{Albert}, \{\text{Carry boxes}\} \rangle, \\ O^5 &= \langle \text{Ben}, \emptyset \rangle, \\ O^6 &= \langle \text{Ben}, \{\text{Box books}\} \rangle, \\ O^8 &= \langle \text{Ben}, \{\text{Carry boxes}\} \rangle \end{aligned}$$

Example 7: Pruning the options.

For the remaining set of options, we want to know their utility. The utility of an option O^i at time t is some function combining all preference state concepts. An example of this is the work by Feng et al. in [30], in which utility is defined as a weighted sum over experience, ability and individual state. These three concepts are again weighted sums, for example individual state is a weighted sum over personality, emotion and fatigue of an agent.

For our example, the set of preference state concepts is:
 $\mathcal{S}_{\mathcal{P}} = \{\text{Physical match}, \text{Knowledge match}\}$

The utility of an option $O^i = \langle A_x, \check{\mathcal{T}} \rangle$ at time t can be defined as:

$$f_{Utility}^O(\langle A_x, \check{\mathcal{T}} \rangle, t) = \begin{cases} -1 & \text{if } \check{\mathcal{T}} = \emptyset \\ f_{\mathcal{U}}(\text{Physical match}, \langle A_x, \check{\mathcal{T}} \rangle, t) \\ + f_{\mathcal{U}}(\text{Knowledge match}, \langle A_x, \check{\mathcal{T}} \rangle, t) & \text{otherwise} \end{cases}$$

This yields the following utility values:

	Utility at $t = 0$	Utility at $t = 30$
O^1	-1	-1
O^2	0	0.8
O^4	0	-0.4
O^5	-1	-1
O^6	0	0.1
O^8	-0.2	-0.2

Example 8: Calculating the utility of options.

Using this list of options (role assignments) and their utilities, we have to find a task allocation TA (which is a set of role assignments) that:

- Includes exactly one role assignment for each actor.
- Allocates each task to exactly one actor.
- Maximizes the utility. We define the utility of a task allocation as the sum over the utilities of all options included in the task allocation:

$$f_{Utility}^{TA}(TA^i, t) = \sum_{x \in TA^i} f_{Utility}^O(x, t).$$

The problem of finding a task allocation as defined above, can be cast as an instance of the SPP, as described in [Appendix 2](#).

As our example is quite small, we can easily find the maximum utility task allocation without using the SPP, by just exploring all options. All task allocations that include exactly one role assignment for each actor and that allocate each task to exactly one actor are:

$$TA^1 = \{O^2, O^8\} = \{\langle \text{Albert}, \{\text{Box books}\} \rangle, \langle \text{Ben}, \{\text{Carry boxes}\} \rangle\},$$

$$TA^2 = \{O^6, O^4\} = \{\langle \text{Ben}, \{\text{Box books}\} \rangle, \langle \text{Albert}, \{\text{Carry boxes}\} \rangle\}$$

Now we can calculate the maximum-utility task allocation for our book relocating example. Filling in all values we have given before yields that when starting the tasks ($t = 0$), TA^2 has utility 0 and is preferred over TA^1 which has utility -0.2 . Thus, at time 0, Albert should carry the boxes and Ben should put the books in the boxes. After 30 minutes ($t = 30$) however, TA^1 has utility 0.6 and is preferred over TA^2 which has utility -0.3 . Albert and Ben should switch tasks.

Example 9: Finding the maximum-utility task allocation.

4 Model for Adaptive Automation

In this section, a model for adaptive automation is presented, based on the framework presented in the previous section (Section 3, Figure 6). This model is designed to work in complex and dynamic environments. The model is applicable to mixed human-robot teams that include at least one human and one robot, but could include multiple humans and/or multiple robots. This section is structured as follows: Firstly a general description of the proposed model is given in Section 4.1. After this, we detail important concepts and their formalizations in Sections 4.2 to 4.5. How different levels of autonomy are modeled is explained in Section 4.2. Section 4.3 covers the modeling of cognitive task load. Section 4.4 details the concept of coordination costs and finally the utility function is explicated in Section 4.5.

4.1 General Description of Model

Adaptive automation is the process in which a robot that is part of a mixed human-robot team dynamically adapts its level of autonomy on one or multiple tasks, as explained in Section 2.5. Adaptive automation can be seen as a special case of dynamic task allocation. We are not just dynamically allocating tasks, but we are dynamically allocating tasks at a specific level of autonomy. Therefore, we can build a model for adaptive automation, based on the framework presented in the previous section (Section 3, Figure 6). If we are using the framework, the next step in building the model is defining the factors we are including as influence on adaptive automation. As argued in Section 2.3, cognitive task load is a good candidate as it affects performance and is influenced by the tasks an actor has. Specifically, it is likely to be influenced by at which level of autonomy an allocated task is. We will include the predicted cognitive task load of an actor on a set of tasks (a role assignment) as a preference factor in our framework. The cognitive task load can not be the only preference factor however. We also need to have some preference factor that represents coordination costs. Team performance could benefit from an actor switching between different (levels of autonomy of) tasks if it reduces the negative effect on performance of the cognitive state he is in, but only if the costs of switching do not outweigh the cost of the negative effect on performance of the cognitive state [26]. Switching between different levels of autonomy on a task has been shown to have such costs [53].

There are many other factors that could be included in the model. Some examples can be seen in the list of influences on task allocation in Section 2.4. The appropriate level of autonomy for a robotic actor on a task depends for example on how well the robot is trusted to do the task [32]. The current research will focus mainly on cognitive task load and secondary on coordination costs as preference factors that differentiate between a task at different levels of autonomy (e.g. way-point navigation vs. teleoperation). Future research should have no problem extending the model to include other such factors and to include more general preference factors that differentiate between different tasks (e.g. navigation vs. communication).

4.2 Levels of Autonomy

In this section, we will detail how adaptive automation can be seen as a special case of dynamic task allocation. This involves modeling tasks that can be executed at different levels of autonomy in such a way that adaptive automation can be described using the framework presented in the previous section (Section 3, Figure 6). Firstly, we will give a high level description of how we describe adaptive automation in terms of dynamic task allocation in Section 4.2.1. Next, in Section 4.2.2, we will formalize this process.

4.2.1 General Description of Modeling Levels of Autonomy

Based on the task analysis of a team task, possible levels of automation for all tasks and their interrelations need to be defined.¹ Tasks that have multiple possible levels of automation are replaced in the task analysis by a separate version of the task for each different level of autonomy. The separate versions all need to be described in terms of task state concepts. The same tasks at several different levels of autonomy can be modeled as several mutually exclusive subtasks. For example, the task of navigation (controlling the movements of the robot) at three different levels of autonomy is shown in Figure 7. We define three mutually exclusive subtasks of navigation, which correspond to the three levels of autonomy. Firstly, at the lowest level of autonomy of the robot, navigation is called tele-operation. Tele-operation means the human fully controls the robot's every move. An example of navigation at an intermediate level of autonomy is way-point navigation. Way-point navigation implies that a human operator set goal coordinates (way-points) for the robot to move to, but the robot itself computes a path to these goals and follows this on its own. Way-point navigation is further divided into the subtasks of setting way-points and following way-points. The highest level of autonomy implies that the robot also decides itself where it wants to drive (goal), the human has no role in the navigation process at the highest level of autonomy. We now define that only human team members are able to execute the following tasks: tele-operation and setting way-points. Only robotic team members are able to execute the following tasks: following way-points, full autonomy. Mutually exclusive task can be modeled by adding a dummy actor, which (as explained in Section 3.2) is a placeholder for tasks that are not executed. All but one of the mutually exclusive tasks (a single task at different levels of autonomy) should be forcedly allocated to the dummy actor, ensuring a task is only allocated at a single level of autonomy to a real actor. The dummy actor does not execute tasks, so a task is only executed at a single level of autonomy.

Interrelations between tasks might be different at each level of autonomy. The execution of a certain task at a certain level of autonomy might not be possible without another task being at a specific level of autonomy. For example for a robot to fully autonomously navigate it needs to be able to autonomously process dynamic environmental

¹This could be extended to include narrowing of possible levels of autonomy for different tasks by the robot operator, i.e. work agreements, like in [41]. Work agreements are rules a robot operates under, agreed upon before starting a team task, giving the human operator room to restrict which tasks can be done by the robot at which level of autonomy, and possibly when.

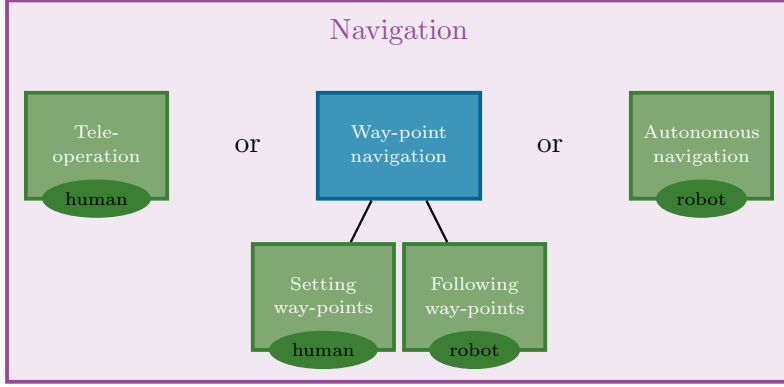


Figure 7: The task of navigation and its different level of autonomy variants, which are mutually exclusive subtasks. Whether a human or robot is needed to execute a task is seen at the bottom in the elliptical label.

data, e.g. detect obstacles, that is relevant for planning.

4.2.2 Formalization of Modeling Levels of Autonomy

In this section, we will formalize the general idea of how we describe adaptive automation in terms of dynamic task allocation that was given above. Firstly, we have a set of actors: $\mathcal{A}'(t) = \{A_1, A_2, \dots, A_n\}$ that comprises the team we want to allocate tasks for. We define the input for the model as this set plus a dummy actor: $\mathcal{A}(t) = \mathcal{A}'(t) \cup \{A_{\text{dummy}}\}$. As explained in Section 3.2 this dummy actor is a placeholder for tasks that are not executed.

Secondly, we have a set of tasks: $\mathcal{T}'(t) = \{T_1, T_2, \dots, T_m\}$ that need to be allocated. For each of these tasks, multiple levels of autonomy for the robot might be possible: $T_x^1, T_x^2, \dots, T_x^k$. For all levels of autonomy that are in between no autonomy and full autonomy (2 to $k-1$), the task execution requires both the human and robot. In these cases, tasks are modeled as two separate subtasks (T_x^y becomes $T_x^{y,h}, T_x^{y,r}$). The resulting set of subtasks each task consists of is: $T_x = \{T_x^1, T_x^{2,h}, T_x^{2,r}, \dots, T_x^{k-1,h}, T_x^{k-1,r}, T_x^k\}$. The input for the model is the set of tasks that includes all these different possibilities for each task $\mathcal{T}(t) = T_1 \cup T_2 \cup \dots \cup T_m$.

To make sure a task is not executed at multiple levels of autonomy, for each task (in $\mathcal{T}'(t)$), we forcedly allocate either all but one of the possible level of autonomy variants of the task or all possible level of autonomy variants of the task to the dummy actor. Note that we leave open the option that the task is not executed at all, meaning it is not allocated at any level of autonomy to a real actor and thus all level of autonomy variants are allocated to the dummy actor. This forced allocation of tasks to the dummy actor is modeled by a restrictive state concept `DummyExecute` ($\in \mathcal{S}_{\mathcal{R}}$). As explained in Section 3.3.5, restrictive state concepts are always defined in such a way, that they return false if and only if an inputted option should be pruned (and true otherwise). We thus define the concept `DummyExecute` to return false if and only if an inputted option is a role assignment for the dummy actor that for some task (in $\mathcal{T}'(t)$) does not contain either

all possible level of autonomy variants or all but one possible level of autonomy variants of that task. This can be described formally by stating that if an role assignment is for the dummy actor, it should contain either all or all but one possible level of autonomy variants of all tasks (in $\mathcal{T}'(t)$):

$$\begin{aligned}
& f_{\mathcal{U}}(\text{DummyExecute}, \langle A_x, \check{\mathcal{T}} \rangle, t) = \\
& (A_x = A_{\text{Dummy}}) \rightarrow \\
& \forall(T_y \in \mathcal{T}'(t)) : \begin{array}{ll} T_y \setminus \check{\mathcal{T}} = \emptyset & \vee \\ T_y \setminus \check{\mathcal{T}} = \{T_y^1\} & \vee \\ T_y \setminus \check{\mathcal{T}} = \{T_y^{k_y}\} & \vee \\ (\exists b : T_y \setminus \check{\mathcal{T}} = \{T_y^{b,h}, T_y^{b,r}\} \wedge 1 < b < k_y) & \end{array}
\end{aligned}$$

where k_y is the highest level of autonomy variant of T_y .¹

We can define additional restricting state concepts to make sure human tasks can only be allocated to humans and likewise for robots. This is done by defining two actor properties, namely one that is only true for human actors ($\text{IsHuman} \in \mathcal{S}^A$) and one that is only true for robotic actors ($\text{IsRobot} \in \mathcal{S}^A$). We further define two task properties, one that defines whether a human actor could execute the task ($\text{IsHumanTask} \in \mathcal{S}^A$), and the other defines whether a robotic actor could execute the task ($\text{IsRobotTask} \in \mathcal{S}^A$). The property IsHumanTask is true for a task T_y^a if and only if either $a = 1$ (the lowest level of autonomy variant of T_y) or $a = b, h$ for some integer b (where $1 < b < k_y$). The property IsRobotTask is true for a task T_y^a if and only if either $a = k_y$ (the highest level of autonomy variant of T_y) or $a = b, r$ for some integer b (where $1 < b < k_y$).

We can now define a restricting state concept $\text{ActorMatch} \in \mathcal{S}_{\mathcal{R}}$ that makes sure role assignments for human actors contain only tasks that can be executed by a human actor and likewise for robotic actors. The concept ActorMatch is defined to be true if and only if an option is a role assignment that only assigns a human actor tasks executable by a human and a robotic actor tasks executable by a robot:

$$\begin{aligned}
& f_{\mathcal{U}}(\text{ActorMatch}, \langle A_x, \check{\mathcal{T}} \rangle, t) = \\
& (f_{\mathcal{A}}(\text{IsHuman}, A_x, t) \rightarrow \forall(T_y \in \check{\mathcal{T}}) : f_{\mathcal{T}}(\text{IsHumanTask}, T_y, t)) \wedge \\
& (f_{\mathcal{A}}(\text{IsRobot}, A_x, t) \rightarrow \forall(T_y \in \check{\mathcal{T}}) : f_{\mathcal{T}}(\text{IsRobotTask}, T_y, t))
\end{aligned}$$

Interrelations between tasks are modeled by the task property CompatibleTaskSet . The task property CompatibleTaskSet ($f_{\mathcal{T}}(\text{CompatibleTaskSet}, T_x^y, t)$) relates a given task at a specific level of autonomy (T_x^y) at a point in time (t) to a set of tasks that are rendered possible in combination with the input task (T_x^y). The output of CompatibleTaskSet is of the form $\{\check{T}_1(\subseteq T_1), \check{T}_2(\subseteq T_2), \dots, \check{T}_m(\subseteq T_m)\}$ where $\{T_1, T_2, \dots, T_m\} = \mathcal{T}'(t)$.

¹This can be formally stated as: $\exists k_y : (T_y^{k_y} \in T_y \wedge \neg \exists k'_y : (T_y^{k'_y} \in T_y \wedge k'_y > k_y))$

Which specific subsets are outputted depends on domain dependent knowledge. `CompatibleTaskSet` describes what tasks are compatible, so we need to know the nature of the tasks. Note that the set \check{T}_x itself (in the `CompatibleTaskSet` of task T_x^y) should always be empty, regardless the nature of the task, as an actor should never have the same task on different levels of autonomy assigned to him at once.¹ We can now define a restricting state concept `CompatibleTasks`² ($\in \mathcal{S}_{\mathcal{R}}$) that ensures a role assignment contains only compatible tasks. The concept `CompatibleTasks` is defined to be true if for every task in a role assignment, all other tasks in the role assignment are contained in its set of compatible tasks. The concept `CompatibleTasks` is also defined to be true if the role assignment is targetted at the dummy actor. As the dummy actor does not actually execute tasks, the compatibility of the set of tasks allocated to the dummy actor is irrelevant.³ The formal definition of the concept `CompatibleTasks` is:

$$f_{\mathcal{U}}(\text{CompatibleTasks}, \langle A_x, \check{T} \rangle, t) =$$

$$(\forall y, z, v, w : (T_y^z \in \check{T} \wedge T_v^w \in \check{T} \wedge \neg(y = v \wedge z = w)) \rightarrow$$

$$(\exists S : T_y^z \in S \in f_{\mathcal{T}}(\text{CompatibleTaskSet}, T_v^w, t)))$$

$$\vee (A_x = A_{\text{Dummy}})$$

In [Appendix 3](#), the effect of the pruning concepts described in this section on the number of possible role assignments is described.

4.3 Cognitive Task Load

In this section, we will describe how we include cognitive task load in our model. Firstly, we will describe the general idea of how cognitive task load can be seen in the context of the model in [Section 4.3.1](#). Next we will formalize these ideas in [Section 4.3.2](#).

¹Actually, the restrictive concept `DummyExecute` together with the demand that all elements of a task allocation are mutually exclusive (i.e. no task is allocated to multiple actors), already enforces that role assignments including the same task at different levels of autonomy cannot possibly occur in a task allocation. Despite this, pruning the set of possible role assignments as much as possible before starting to search for a task allocation is important to keep the problem of finding the optimal task allocation feasible to solve.

²This concept is quite similar to the concept ‘No incompatible tasks’ that was introduced in the example formalization in [Section 3.3.4](#). The concept ‘CompatibleTasks’ is different as it is tailored to suit the formalization of the current model. The same relation holds for the concept ‘Incompatible tasks’ (in formalization example, [Section 3.3.3](#)) versus the new concept ‘CompatibleTaskSet’ (suited to the model).

³Actually, we *need* to be able to allocate task sets including incompatible tasks to the dummy actor, otherwise we cannot model tasks with more than two possible level of automony variants. If we have a task with more than two level of automony variants, the dummy actor should be assigned multiple level of automony variants of the task (so only maximum one variant is allocated to a real actor). As before said, different level of automony variants of a task are always incompatible.

4.3.1 General Description of Modeling Cognitive Task Load

In Section 2.3, we detailed previous work on cognitive task load (CTL). We explained CTL can be described by three metrics [23]. The first metric is the percentage of time that a person is occupied (TO) by his/her tasks. The second metric is the number of task-set switches (TSS), which is the number of times that a person has to switch between different tasks. The third metric is the level of information (LIP) processing that is needed for the current tasks the person is doing. We want to use the predicted CTL level of an actor on a task set to help decide how well this task set is suited to be executed by the actor (relative to other tasks sets). All three metrics of CTL are situation state concepts, combining actor states with task analysis. More specifically, the metrics can be described as some function over the option of allocated tasks of an individual and properties of these tasks (o.a. LIP). Using the three metrics, we can estimate whether the CTL level of an actor will be in a problem region given a set of tasks (role assignment). Task allocations that keep actors out of CTL problem regions should be preferred as this benefits performance.

As explained in Section 2.3, timing is also an important aspect in CTL. The longer a person’s CTL is in a problem region, the more negative the effect on performance will be. Typically, vigilance and underload problems occur only after some time, while overload and cognitive lock-up problems can occur even if the CTL has only been in the problem region for a short time [8].

Cognitive task load as described above only makes sense in the context of human actors, not for robotic actors.¹ For example, robotic actors can not suffer from vigilance problems if they are bored, because generally robots can not be bored.

4.3.2 Formalization of Modeling Cognitive Task Load

We will now formalize how CTL is included in our model. The three CTL metrics are situation state concepts: $TO, TSS, LIP \in \mathcal{S}^U$. The specific functions combining actor states with task analysis into TO, TSS or LIP are not specified here as they could be different for each domain.² We assume all values of problem region boundaries are given, how we name them can be seen in Figure 8.

These values of problem regions are actor and time dependent, therefore we model them as actor states. We can now define the situation state concepts that define whether actors will be in a problem region if they are given a certain set of tasks (functions that map to true (actor in state) or false (actor not in state)):

Vigilance:	$TSS < TSS_{low} \wedge TO > TO_{high} \wedge LIP < LIP_{low}$
Underload:	$TSS < TSS_{low} \wedge TO < TO_{low} \wedge LIP < LIP_{low}$
Overload:	$TSS > TSS_{high} \wedge TO > TO_{high} \wedge LIP > LIP_{high}$

¹Obviously, CTL also does not make sense in the context of dummy actors, as they are not really actors (that execute tasks).

²A good example of how to compute CTL for the urban search and rescue domain is the model made by Colin et al. [34].

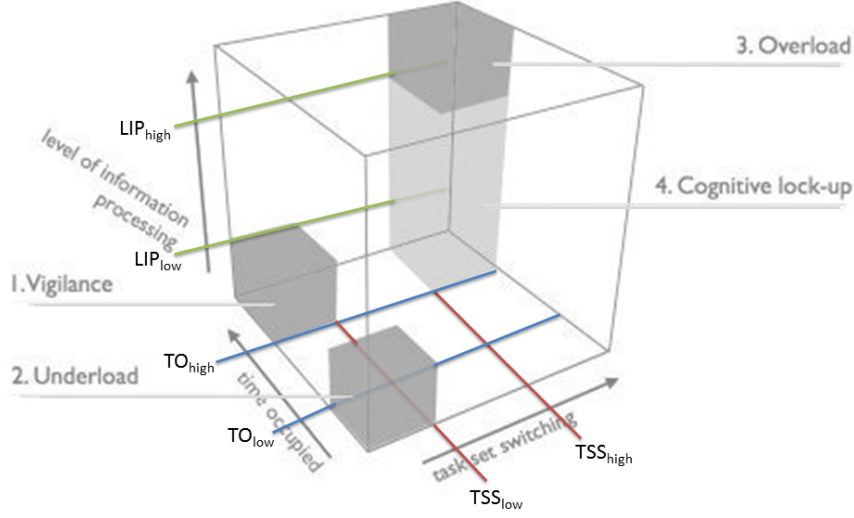


Figure 8: Values of problem region boundaries in the Cognitive Load Space.

Cognitive lock-up: $TSS > TSS_{high} \wedge TO > TO_{high} \wedge LIP < LIP_{high}$

Neutral: $\neg \text{Vigilance} \wedge \neg \text{Underload} \wedge \neg \text{Overload} \wedge \neg \text{Cognitive lock-up}$

Fully formalizing the above definitions of situation state concepts yields the following definitions:

$$\begin{aligned}
 f_U(\text{Vigilance}, \langle A_x, \check{T} \rangle, t) &= f_U(TSS, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(TSS_{low}, A_x, t) \wedge \\
 & f_U(TO, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(TO_{high}, A_x, t) \wedge \\
 & f_U(LIP, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(LIP_{low}, A_x, t) \\
 f_U(\text{Underload}, \langle A_x, \check{T} \rangle, t) &= f_U(TSS, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(TSS_{low}, A_x, t) \wedge \\
 & f_U(TO, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(TO_{low}, A_x, t) \wedge \\
 & f_U(LIP, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(LIP_{low}, A_x, t) \\
 f_U(\text{Overload}, \langle A_x, \check{T} \rangle, t) &= f_U(TSS, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(TSS_{high}, A_x, t) \wedge \\
 & f_U(TO, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(TO_{high}, A_x, t) \wedge \\
 & f_U(LIP, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(LIP_{high}, A_x, t) \\
 f_U(\text{CognitiveLockUp}, \langle A_x, \check{T} \rangle, t) &= f_U(TSS, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(TSS_{high}, A_x, t) \wedge \\
 & f_U(TO, \langle A_x, \check{T} \rangle, t) > f_{\mathcal{A}}(TO_{high}, A_x, t) \wedge \\
 & f_U(LIP, \langle A_x, \check{T} \rangle, t) < f_{\mathcal{A}}(LIP_{high}, A_x, t) \\
 f_U(\text{Neutral}, \langle A_x, \check{T} \rangle, t) &= \neg f_U(\text{Vigilance}, \langle A_x, \check{T} \rangle, t) \wedge \\
 & \neg f_U(\text{Underload}, \langle A_x, \check{T} \rangle, t) \wedge \\
 & \neg f_U(\text{Overload}, \langle A_x, \check{T} \rangle, t) \wedge \\
 & \neg f_U(\text{CognitiveLockUp}, \langle A_x, \check{T} \rangle, t)
 \end{aligned}$$

We can now define the preference concept $CTL(\in \mathcal{S}^P)$ that gives a preference value for a role assignment based on the CTL of the actor the role assignment is for. If a

role assignment causes an actor to be in one of the CTL problem regions, we give the role assignment a low preference. As the negative effect on performance is slightly less harmful for cognitive lock-up than it is for other problem regions, we give this a slightly higher preference value. Role assignments that do not cause the actor to be in a problem region (optimal CTL) are given a high preference. We define CTL preference values to range from 0 (lowest preference) up to 1 (highest preference).

As explained in Section 2.3, we need to take into account how long a person's CTL has been in a problem region. For the problem regions of cognitive lock-up and overload, negative effects on performance can occur even if an actor has only been in them for a short time (maximum effect estimated at five minutes). For the problem regions vigilance and underload, the negative effect slowly increases as the actor has been in the region for a longer time (maximum estimated at fifteen minutes).

The exact values that should be used for the different problem regions and the timing aspects should be tweaked with domain dependent (experimental) knowledge. For now we use an estimation of the values, which is defined below:

$$\text{CTL} = \begin{cases} 1 & \text{if Neutral} \\ 0.2 \quad (300s \text{ in region}) \text{ up to } 0.7 \quad (0s \text{ in region}) & \text{if Cognitive lock-up} \\ 0 \quad (300s \text{ in region}) \text{ up to } 0.5 \quad (0s \text{ in region}) & \text{if Overload} \\ 0 \quad (900s \text{ in region}) \text{ up to } 0.5 \quad (0s \text{ in region}) & \text{otherwise (if Vigilance} \\ & \text{or Underload)} \end{cases}$$

where we assume a linear relation between time and the preference concept CTL within the above time frames.

We can formalize the above, by defining four actor states: *VigilancePast*, *UnderloadPast*, *OverloadPast* and *CognitiveLockUpPast* ($\in S^A$). These actor states describe, given an actor and a time point as input, for how long the actor has been in the problem region corresponding to the name of the actor state concept (in seconds). If an actor is currently not in the problem region corresponding to the name of the actor state concept, the output will be zero.

Note that we have not yet defined that CTL is only applicable to human actors. We define this by letting the output of the preference concept CTL always have the same value for all non-human actors, independent of the set of tasks that is allocated to them.¹ Adding this to the definition and formalizing it yields the following fully defined CTL concept:

¹We let this value be the highest preference value, as this means there is no negative influence of CTL.

$$f_U(\text{CTL}, \langle A_x, \check{T} \rangle, t) =$$

$$\left\{ \begin{array}{ll} 1 & \text{if } \neg f_A(\text{IsHuman}, A_x, t) \\ 1 & \text{if } f_U(\text{Neutral}, \langle A_x, \check{T} \rangle, t) \\ 0.7 - 0.5 * \frac{\min(300, f_A(\text{CognitiveLockUpPast}, A_x, t))}{300} & \text{if } f_U(\text{CognitiveLockUp}, \langle A_x, \check{T} \rangle, t) \\ 0.5 - 0.5 * \frac{\min(300, f_A(\text{OverloadPast}, A_x, t))}{300} & \text{if } f_U(\text{Overload}, \langle A_x, \check{T} \rangle, t) \\ 0.5 - 0.5 * \frac{\min(900, f_A(\text{VigilancePast}, A_x, t))}{900} & \text{if } f_U(\text{Vigilance}, \langle A_x, \check{T} \rangle, t) \\ 0.5 - 0.5 * \frac{\min(900, f_A(\text{UnderloadPast}, A_x, t))}{900} & \text{otherwise} \\ & (\text{if } f_U(\text{Underload}, \langle A_x, \check{T} \rangle, t)) \end{array} \right.$$

where $\min(x, y)$ returns the smaller of its two arguments: $\min(x, y) = x$ if $x < y$, and $\min(x, y) = y$ otherwise.

4.4 Coordination Costs

As explained in Section 4.1, team performance could benefit from an actor switching between different (levels of autonomy of) tasks if it reduces the negative effect on performance of the cognitive state he is in, but only if the costs of switching do not outweigh the cost of the negative effect on performance of the cognitive state [26]. Switching between different levels of autonomy on a task has been shown to have such costs [53]. In this Section we will describe how we will include coordination costs in our model. Firstly we will give a general idea of how coordination costs are modeled in Section 4.4.1. Next, Section 4.4.2 will formalize these ideas.

4.4.1 General Description of Coordination Costs

The coordination costs have to take into account two aspects of switching between tasks. The first aspect is whether the task switch implies an actor gets a task assigned he was not currently assigned to at all or which was currently assigned to him, but at a different level of autonomy. For human actors, there are costs associated to switching between tasks [53]. For robotic actors, these sort of costs do not exist. For a human actor, if a

¹This function and the similar ones below it are linear in the relevant time frame. This is because during the period a human actor remains in the same problem region, the time the human actor has been in this problem region has a linear relation to the time point t it is measured at.

task is assigned that was not currently assigned at all, this has a relatively high cost,¹ as the human actor needs to switch (redirect his attention) to this task. If a task was already assigned to the actor, but with a higher level of autonomy of the robotic actor, the cost is slightly lower. The actor does not need to switch to a new task, but only needs to take up more responsibility in executing it. If a task was already assigned to the actor, but with a lower level of autonomy of the robotic actor, the cost is even lower. Again the actor does not need to switch to a new task, but the robotic actor takes over some of the responsibility in executing the task. Lastly, if a task was already assigned to the actor, and the level of autonomy of this task does not change, there are no costs as nothing changes.

The second aspect that coordination costs have to take into account is how often task reallocations take place. Timing is important in adaptive automation, not only when, but also how often the level of autonomy of a task is changed. Changing the level of autonomy too often could cause extra workload [26]. The frequency at which changing starts causing extra workload likely depends on the nature of a task (domain dependent knowledge) and on the individual actor (e.g. his personality and cognitive state) [54].

4.4.2 Formal Description of Coordination Costs

We will now formalize how coordination costs are included in our model. Firstly we define a function f_{cost} that maps a pair containing a single actor and a single task at a certain time point to a preference value. This preference value represents how preferable the actor-task pair is, based on coordination costs that arise when the task gets allocated to the actor. The higher the coordination costs, the less an actor-task pair is preferred. High coordination costs are thus associated with low preference values and vice versa. The function f_{cost} models the first aspect of coordination costs, namely whether the task was already assigned to the actor and if so, whether the level of autonomy changes.

If a task was already allocated to an actor, and the level of autonomy on this task does not change, there are no coordination costs as there is no switching. If a task was already allocated to an actor, but at a different level of autonomy, there are two options. Firstly, the level of autonomy could go down, this means the human actor gets more responsibilities in executing the task. The cost of lowering the level of autonomy is quite high. Secondly, the level of autonomy could go up, this means the robotic actor takes over some responsibility. The switching costs for the human actor are quite low if the level of autonomy goes up. The cost of allocating a task that was not allocated at all to the actor before is the highest. We define coordination cost preference values to range from 0 (lowest preference, thus highest coordination costs) up to 1 (highest preference, thus lowest coordination costs). The exact values that should be used for the different contexts (changes in task allocation) should be tweaked with domain dependent (experimental) knowledge. For now we use an estimation of the values, which is given below:

¹The cost is very much influenced by how different the tasks are, as argued by Colin et al. [34]. This is outside the scope of the current research.

$$f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) = \begin{cases} 0 & \text{if task was not yet allocated to actor} \\ 0.2 & \text{if task was already allocated to the actor,} \\ & \text{but with a lower level of autonomy} \\ 0.5 & \text{if task was already allocated to the actor,} \\ & \text{but with a higher level of autonomy} \\ 1 & \text{if task was already allocated to the actor,} \\ & \text{with the same level of autonomy} \end{cases}$$

To formalize this function, we need to know the set of tasks that is currently allocated to an actor. To this extent, we define the actor state concept `CurrentTasks`, which returns the current task set of an actor. We can now formalize the cost function described above:

$$f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) = \begin{cases} 0 & \text{if } \neg \exists w : T_y^w \in f_{\mathcal{A}}(\text{CurrentTasks}, A_x, t) \\ 0.2 & \text{if } \exists w : T_y^w \in f_{\mathcal{A}}(\text{CurrentTasks}, A_x, t) \wedge v < w \\ 0.5 & \text{if } \exists w : T_y^w \in f_{\mathcal{A}}(\text{CurrentTasks}, A_x, t) \wedge v > w \\ 1 & \text{otherwise (if } \exists w : T_y^w \in f_{\mathcal{A}}(\text{CurrentTasks}, A_x, t) \wedge v = w \text{)} \end{cases}$$

The second aspect that coordination costs have to take into account is how often task reallocations take place. Changing tasks too often could cause extra workload, so we add a timing factor to the coordination costs. We define a time frame of five minutes after a task has been reallocated. If a task is reallocated again within this time frame, a time penalty is subtracted from the coordination cost preference value. This penalty is high if the task was just reallocated and decreases linearly to zero when the reallocation was five minutes ago. Note that we again make an estimation of the values and how they behave (length of time frame, height of penalty, linearity of function) and this should be tweaked using domain dependent (experimental) knowledge. The function $f_{\text{cost+time}}$ adds the timing factor to the coordination costs and is defined below:

$$f_{\text{cost+time}}(\langle A_x, T_y^v \rangle, t) = \begin{cases} f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) & \text{if last task reallocation was (more} \\ & \text{than) 5 minutes (300 seconds) ago} \\ & \text{or task was already allocated to actor} \\ & \text{at same level of autonomy} \\ \max(0, f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) - \frac{300 - i}{300} * 0.25) & \text{otherwise (where } i \text{ is the number} \\ & \text{of seconds that passed since} \\ & \text{the last task reallocation)} \end{cases}$$

To formalize this function, we need to know how long ago the previous reallocation of a task occurred. To this extent, we define a function $f_{\text{reallocation}}$. This function takes an actor task-pair and a time point as input and returns how many seconds have past since the last reallocation of the inputted task that concerned the inputted actor. In other words, it returns how much time has passed since the last time the task either appeared, disappeared or changed level of autonomy in the task set allocated to the actor. Using

the last reallocation-function we can formalize the definition of the function $f_{\text{cost+time}}$:

$$f_{\text{cost+time}}(\langle A_x, T_y^v \rangle, t) = \begin{cases} f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) & \text{if } f_{\text{reallocation}}(\langle A_x, T_y^v \rangle, t) \\ & \geq 300 \\ & \text{or } f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) = 1 \\ \max(0, f_{\text{cost}}(\langle A_x, T_y^v \rangle, t) - \frac{300 - f_{\text{reallocation}}(\langle A_x, T_y^v \rangle, t)}{300} * 0.25) & \text{otherwise} \end{cases}$$

We can now define the preference concept coordination costs ($\text{CC} \in \mathcal{S}^{\mathcal{P}}$), which describes the preference of a role assignment, based on the coordination costs this role assignment causes. The preference of a role assignment, concerning coordination costs, is some function over the output of the function $f_{\text{cost+time}}$ over all actor task-pairs contained in the role assignment. An actor task-pair is contained in a role assignment, if the role assignment allocates the task to the actor. Generally the average output of the function $f_{\text{cost+time}}$ over all actor task-pairs contained in a role assignment will be a good approximation of the coordination costs of this role assignment. This is because as the more tasks change, the higher the costs are and the lower the preference is. The average might not be a good function for atypical domains however and should be adjusted to some more complex function if necessary.

Note that we have not yet defined that coordination costs are only applicable to human actors. We do this by letting the output of the preference concept CC always have the same value for all non-human actors, independent of the set of tasks that is allocated to them.¹ The definition of the preference concept CC is given below:

$$f_{\mathcal{U}}(\text{CC}, \langle A_x, \check{\mathcal{T}} \rangle, t) = \begin{cases} \frac{\sum_{\forall T_y^v \in \check{\mathcal{T}}} f_{\text{cost+time}}(\langle A_x, T_y^v \rangle, t)}{|\check{\mathcal{T}}|} & \text{if } f_{\mathcal{A}}(\text{IsHuman}, A_x, t) \\ 1 & \text{otherwise (} A_x \text{ is a robotic actor} \\ & \text{or the dummy actor)} \end{cases}$$

4.5 Utility Function

The utility function maps role assignments at a certain point in time to their utility. The utility of a role assignment is some combination of all preference concepts. In this case the set of preference concepts consists of the preference based on CTL and the preference based on coordination costs (CC). As explained in Section 4.1, team performance benefits from an actor switching between different (levels of autonomy of) tasks if the the negative effect on performance of the cognitive state he is in outweighs the costs of switching. The utility of a role assignment should thus be a measure that represents the preference of the role assignment based on CTL minus the coordination

¹We let this value be the highest preference value, as this means there are no coordination costs.

costs. The preference concept CC is high if the coordination costs are low (because this is preferred) and vice versa. Therefore the utility of a role assignment is the addition of the two preference concepts CTL and CC.

For clarity, we define that the lowest utility equals 0 and the highest utility equals 1. To fit this range, we scale the sum of the preference concepts CTL and CC (which also both range from 0 to 1) by dividing it by two. More formally, the utility of a role assignment (an option) $O^i = \langle A_x, \check{T} \rangle$ at time t is the sum of values for all preference state concepts (CTL and CC), divided by two:

$$f_{Utility}^O(O^i, t) = (f_u(\text{CTL}, O^i, t) + f_u(\text{CC}, O^i, t))/2$$

5 Experiment

In this section, we aim to validate the model presented in the previous section (Section) by using it in an experimental setting. Firstly, an example usage of the model is shown, by instantiating it for the urban search and rescue domain in Section 5.1. After this, an experiment that uses the instantiated model is described in Section 5.2. Results of this experiment are described in Section 5.3 and conclusions are detailed in Section 5.4.

5.1 Instantiating the Model for the Urban Search and Rescue Domain

In this section, the model presented in the previous section (Section 5) is instantiated for an urban search and rescue setting. Specifically, we focus on the cooperation between a single robot and its operator. We detail how our model can be instantiated to model how the autonomy level of the robot should adapt to the CTL of its operator in a typical urban search and rescue task. Firstly, a short description of the urban search and rescue domain is given in Section 5.1.1. Secondly, Section 5.1.2 details which levels of autonomy different tasks can have. Thirdly, a model for CTL estimation in the urban search and rescue domain is described in Section 5.1.3.

5.1.1 Urban Search and Rescue

Urban search and rescue (USAR) involves finding and extracting victims from disaster sites. These sites can for example be traffic accidents, earthquake sites, or buildings that are on fire. Finding victims in these situations is very stressful and demanding, as the lay-out of the situation is often unsure and dangerous situations could arise. USAR services try to find victims in these harsh conditions as fast as possible, since survivors could be in need of medical attention. If, for example, a building is damaged by an earthquake or bomb, the disaster site could be very unstable. In these situations, it is very unsafe for rescuers to go in to the building, as walls or ceilings could collapse. To risk as little as possible, robots can be used to help map the site. These robots can find out for example, where the possible dangers are and where victims might be located. To this extent, the NIFTi project¹ was set up. In the NIFTi project, we investigate how robots can be applied for USAR. A focus point in this is the cooperation between robot and human team member, since this is often not yet ideal [55].

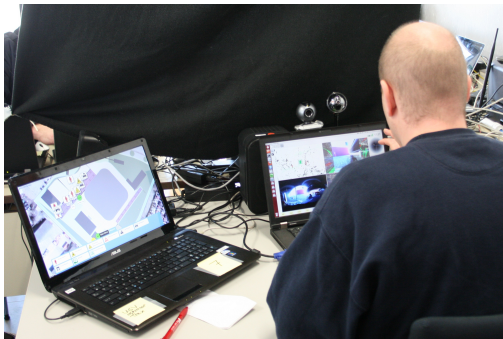
The model presented in Section 5 is instantiated to model cooperation between the unmanned ground vehicle (UGV, see Figure 9a and 9c) developed in the NIFTi project and its operator (Figure 9b). The operator can control the UGV’s movements and see its surroundings via multiple (adjustable) camera views and a colored 3D point cloud (laser mapping) through an operator control unit. The operator and UGV also have access to a tactical display, which is shared with other USAR rescue members. Information about the disaster site is communicated amongst rescuers through this tactical display.

¹www.nifti.eu

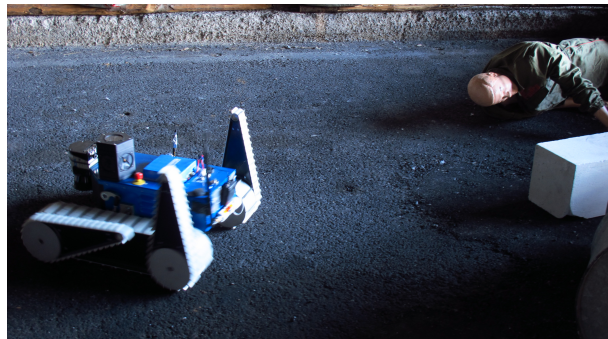
While the UGV is in the field (a disaster site), the operator can stay at a safe location to control it. The UGV is thus not in line-of-sight of the operator.¹



(a) A UGV in scenario



(b) A UGV operator



(c) A UGV with victim

Figure 9: Some pictures taken during NIFTi experiments

The UGV and its operator form a team that gets assigned to tasks during USAR missions. Most of these tasks can be executed at different autonomy levels for the UGV, meaning the UGV has either no, some or full control over task execution. For example, imagine the task ‘uploading relevant information to the tactical display’. The level of

¹For more (detailed) information, see [55].

autonomy of this task concerns who controls which information gets uploaded to the tactical display. If the UGV has no control over this, this means the human decides which information is uploaded. If the UGV has full control, it decides which information is uploaded. Shared control could for example mean that the UGV suggests possible information to upload and the operator can accept or reject this.

As USAR missions are stressful and demanding, and lives could depend on their efficiency and effectiveness, it is important to improve cooperation and team performance as much as possible. To this extent, introducing adaptive automation could be very useful.

The task in USAR is often to explore an area and get an overview of the situation. This often includes locating victims.¹ The task of exploring an area can be reduced to a set of subtasks, as seen in Figure 10.² For the UGV and its operator, the task to explore an area can be divided into three subtasks: driving around, detecting objects and communicating. These three subtasks can be divided into two subtasks each: navigation and obstacle avoidance, obstacle detection and victim detection respectively forwarding information and receiving/processing information. These six (subsub)tasks are not divided into further subtasks for our purpose, so they compose the set of tasks for our model (as seen in the bottom row of boxes (green) in Figure 10). This task set remains constant over the whole mission of exploring the area. For our purpose, detecting and avoiding obstacles can be seen as a single task as both have no meaning without the other. The same holds for detecting victims and forwarding information to others, as the information that needs to be forwarded only contains detected victims. The set of tasks we deal with thus is:

$$\mathcal{T}'(*) = \{\text{Nav}, \text{Obs}, \text{Vic}, \text{Info}\}$$

where the asterisk denotes a wildcard, meaning the value of t has no influence on the set of tasks that need to be allocated (i.e. the set of tasks is constant). The following abbreviations are used:

- Nav : Navigation
- Obs : Obstacle detection and avoidance
- Vic : Victim detection and forwarding information
- Info : Receive and process information from others

5.1.2 Automation Analysis

In this Section, we detail the set of tasks used for the instantiation by introducing the different possible levels of autonomy for each task. After this, we explain which constraints exist between tasks (and actors) and how we model these using pruning rules (restrictive state concepts).

As proposed by Miller [6], we define levels of autonomy for all subtasks separately. For now, we define three possible levels of autonomy for each task: 1, 2 and 3. Level

¹It could also include locating other objects of interest, such as fire or smoke.

²Note that the reduction into subtasks we give here is not the only possible one, but we choose this one to use for the instantiation.

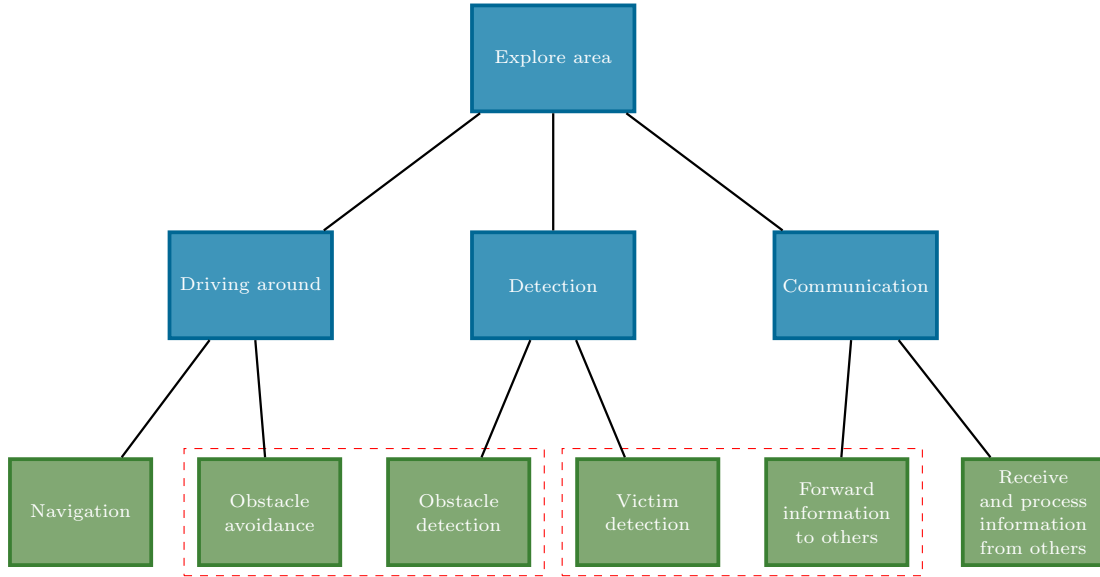


Figure 10: Overview of the different tasks and subtasks in a typical USAR mission, the tasks on the bottom row (green boxes) are used as task set for our task allocation model. The tasks with a dashed box (red) around them are grouped and seen as a single task.

1 means the robot (UGV) has no autonomy, the human controls the robot fully (e.g. tele-operation). Level 2 means there is some autonomy for the robot, but the human makes the higher level decisions (e.g. way-point driving). Level 3 means the robot has full autonomy and makes decisions and executes actions without human interference. An overview of the tasks we define for the different levels of autonomy (LoA) is given in Table 1.

	LoA 1	LoA 2	LoA 3
Nav	Tele-operation	Way-point driving	Full autonomy
Obs	No obstacle detection/avoidance	Robots warns operator for nearby obstacles	Robot avoids obstacles
Vic	No victim detection	Robot suggests possible victims	Robots detects victims and forwards information
Info	No automatic information processing	Robot suggests possible places of interest	Robot decides where to go

Table 1: An overview of the tasks at different levels of autonomy.

Without yet taking constraints into account (e.g. tasks that might be incompatible), this makes the set of tasks that need to be allocated (which again is static):

$$\mathcal{T}(\ast) = \begin{array}{l} \text{Nav} \\ \text{Obs} \\ \text{Vic} \\ \text{Info} \end{array} \cup = \{ \begin{array}{l} \text{Nav}^1, \text{Nav}^{2,h}, \text{Nav}^{2,r}, \text{Nav}^3, \\ \text{Obs}^1, \text{Obs}^{2,h}, \text{Obs}^{2,r}, \text{Obs}^3, \\ \text{Vic}^1, \text{Vic}^{2,h}, \text{Vic}^{2,r}, \text{Vic}^3, \\ \text{Info}^1, \text{Info}^{2,h}, \text{Info}^{2,r}, \text{Info}^3 \end{array} \}$$

Note that if we would not have any pruning, the set of possible role assignments would be enormous even for this small instantiation. As explained in [Appendix 3](#), the number of role assignments without pruning is $(2^{|\mathcal{T}|})^{|\mathcal{A}|}$. In the current instantiation, the number of actors is three (the operator, the robot and the dummy actor) and the number of tasks is 16. This implies there are $(2^{16})^3 = 281474976710656$ distinct role assignments, making it clear that we should have effective pruning. Using the domain independent pruning rules defined in [Section 4.2.2](#) (DummyExecute, ActorMatch and CompatibleTasks), we can reduce the number of options, as explained in [Appendix 3](#). For the current instantiation, there are 481 possible role assignments (256 for the dummy actor, 81 for the other actors each) left after domain independent pruning, which is a huge improvement over the 281474976710656 role assignments that were possible before pruning.

Luckily, there are some more (domain dependent) restricting rules, so we can further prune the set of possible role assignments. Firstly, in the current instantiation all tasks are mandatory. Tasks such as victim detection might not be actively executed the whole time, but it should always be clear who has the responsibility to detect victims allocated to him. With this in mind, we can prune all options in which some task is not executed at any level of autonomy. We can easily do this, by making the restrictive concept DummyExecute (introduced in [Section 4.2.2](#)) more restrictive. The stricter concept DummyExecute is true and only true when a role assignment for the dummy actor contains all but precisely one level of autonomy for each task, which is formally described by the following statement:

$$f_{\mathcal{U}}(\text{DummyExecute}, \langle A_x, \check{\mathcal{T}} \rangle, t) =$$

$$(A_x = A_{\text{Dummy}}) \rightarrow$$

$$\begin{array}{l} \forall (T_y \in \mathcal{T}'(t)) : T_y \setminus \check{\mathcal{T}} = \{T_y^1\} \quad \vee \\ T_y \setminus \check{\mathcal{T}} = \{T_y^{k_y}\} \quad \vee \\ (\exists b : T_y \setminus \check{\mathcal{T}} = \{T_y^{b,h}, T_y^{b,r}\} \wedge 1 < b < k_y) \end{array}$$

where k_y is the highest level of autonomy variant of T_y .^{*}

This reduces the possible role assignment options for the dummy actor from 256^\dagger to 81^\ddagger , meaning there are a total of 243 options left (81 (human) + 81 (robot) + 81 (dummy)).

^{*}This can be formally stated as: $\exists k_y : (T_y^{k_y} \in T_y \wedge \neg \exists k'_y : (T_y^{k'_y} \in T_y \wedge k'_y > k_y))$

[†] $\prod_{T_x \in \mathcal{T}'(t)} (LoA(T_x) + 1)$ where $LoA(T_x)$ returns the number of level of autonomy variants of task T_x (formally defined in [Appendix 3](#)).

[‡] $\prod_{T_x \in \mathcal{T}'(t)} LoA(T_x)$ where $LoA(T_x)$ returns the number of level of autonomy variants of task T_x (formally defined in [Appendix 3](#)).

		Nav		
		LoA 1	LoA 2	LoA 3
Obs	LoA 1	Tele-operation	×	×
	LoA 2	Tele-operation with obstacle warning	×	×
	LoA 3	Tele-operation with obstacle avoidance	Way-point navigation	Full autonomy

Table 2: The relation between different levels of autonomy for navigation (Nav) and obstacle detection/avoidance (Obs).

		Nav		
		LoA 1	LoA 2	LoA 3
Info	LoA 1	Tele-operation	Way-point navigation	×
	LoA 2	Tele-operation with place of interest suggestions	Way-point navigation with place of interest suggestions	×
	LoA 3	×	×	Full autonomy

Table 3: The relation between different levels of autonomy for navigation (Nav) and receiving and processing information from others (Info).

The second restrictive concept we can restrict further using domain knowledge is `CompatibleTasks` (introduced in Section 4.2.2). Setting an autonomy level of one of the tasks could influence which autonomy levels are possible for other tasks. This domain dependent information can be used to prune more options using the restrictive concept `CompatibleTasks`. The relation between different levels of autonomy for navigation (Nav) and obstacle detection/avoidance (Obs) is specified in Table 2. The table shows, for each pair of a level of autonomy variant of navigation and a level of autonomy variant of obstacle detection/avoidance, whether these two tasks are compatible¹. A cross (×) means the tasks are not compatible. If the tasks are compatible, the combination of the two tasks is shortly described. For example, if we set the autonomy level for detecting obstacles very low (LoA 1, top row in table), the robot does not know where obstacles are. Setting the autonomy level for navigation medium or high (LoA 2 or 3) would be a bad idea since the robot has a high risk of crashing into the obstacles it does not detect. If we set the autonomy level for navigation low (LoA 1), the combination with the low autonomy level for detecting obstacles is fine, as the robot is tele-operated.

A same sort of relation exists between different levels of autonomy for navigation (Nav) and receiving and processing information from others (Info), this relation is specified in Table 3. The other tasks do not influence each other in this way.

Using the specified relations in tables 2 & 3, we can define the task property Com-

¹We have define compatible to mean two tasks can be allocated to an actor at the same time, thus occur together in a role assignment.

patibleTaskSet for all tasks:

$$\begin{aligned} f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Nav}^1, *) &= \{\emptyset, \text{Obs}, \text{Vic}, \{\text{Info}^1, \text{Info}^{2,h}, \text{Info}^{2,r}\}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Nav}^{2,*}, *) &= \{\emptyset, \{\text{Obs}^3\}, \text{Vic}, \{\text{Info}^1, \text{Info}^{2,h}, \text{Info}^{2,r}\}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Nav}^3, *) &= \{\emptyset, \{\text{Obs}^3\}, \text{Vic}, \{\text{Info}^3\}\} \end{aligned}$$

$$\begin{aligned} f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Obs}^1, *) &= \{\{\text{Nav}^1\}, \emptyset, \text{Vic}, \text{Info}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Obs}^{2,*}, *) &= \{\{\text{Nav}^1\}, \emptyset, \text{Vic}, \text{Info}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Obs}^3, *) &= \{\text{Nav}, \emptyset, \text{Vic}, \text{Info}\} \end{aligned}$$

$$\begin{aligned} f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Vic}^1, *) &= \{\text{Nav}, \text{Obs}, \emptyset, \text{Info}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Vic}^{2,*}, *) &= \{\text{Nav}, \text{Obs}, \emptyset, \text{Info}\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Vic}^3, *) &= \{\text{Nav}, \text{Obs}, \emptyset, \text{Info}\} \end{aligned}$$

$$\begin{aligned} f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Info}^1, *) &= \{\{\text{Nav}^1, \text{Nav}^{2,h}, \text{Nav}^{2,r}\}, \text{Obs}, \text{Vic}, \emptyset\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Info}^{2,*}, *) &= \{\{\text{Nav}^1, \text{Nav}^{2,h}, \text{Nav}^{2,r}\}, \text{Obs}, \text{Vic}, \emptyset\} \\ f_{\mathcal{T}}(\text{CompatibleTaskSet}, \text{Info}^3, *) &= \{\{\text{Nav}^3\}, \text{Obs}, \text{Vic}, \emptyset\} \end{aligned}$$

Using the restrictive concept `CompatibleTasks`, we can again prune the set of options. This reduces the possible role assignment options for the human actor from 81 to 57 and for the robotic actor from 81 to 36. This implies that there are a total of 174 options left (57 (human) + 36 (robot) + 81 (dummy)).

5.1.3 Cognitive Task Load

A model for estimating the CTL¹ of robot operators in the USAR domain was developed by Colin et al² [34]. Figure 11 shows an overview of the steps Colin’s [34] model takes to estimate the CTL of a robot operator in the USAR domain.

The first step of the model consists of gathering events, for example when the operator starts moving the robot or when the operator stops using the tactical display (Figure 11, top box). Using these events, a task bar concerning the past two minutes³ is generated that shows which tasks the operator is doing and when he is doing them (Figure 11, second box from top). The model, together with an analysis of how tasks interact that is supposed as input, now combines these tasks into a single task bar (Figure 11, third box from top). The model presupposes a task analysis as input. For every task, this analysis describes three properties relevant to the CTL. Firstly, the level of information processing (LIP) an operator needs to execute the task. Secondly, how high the mental occupancy (MO) is, i.e. the proportion of an operator’s attention that is needed for

¹As introduced in Section 2.3.

²From now on, ‘Colin et al.’ is abbreviated to ‘Colin’.

³Or concerning one minute or more than two minutes, depending on the configuration of the model. We configure it to look back two minutes.

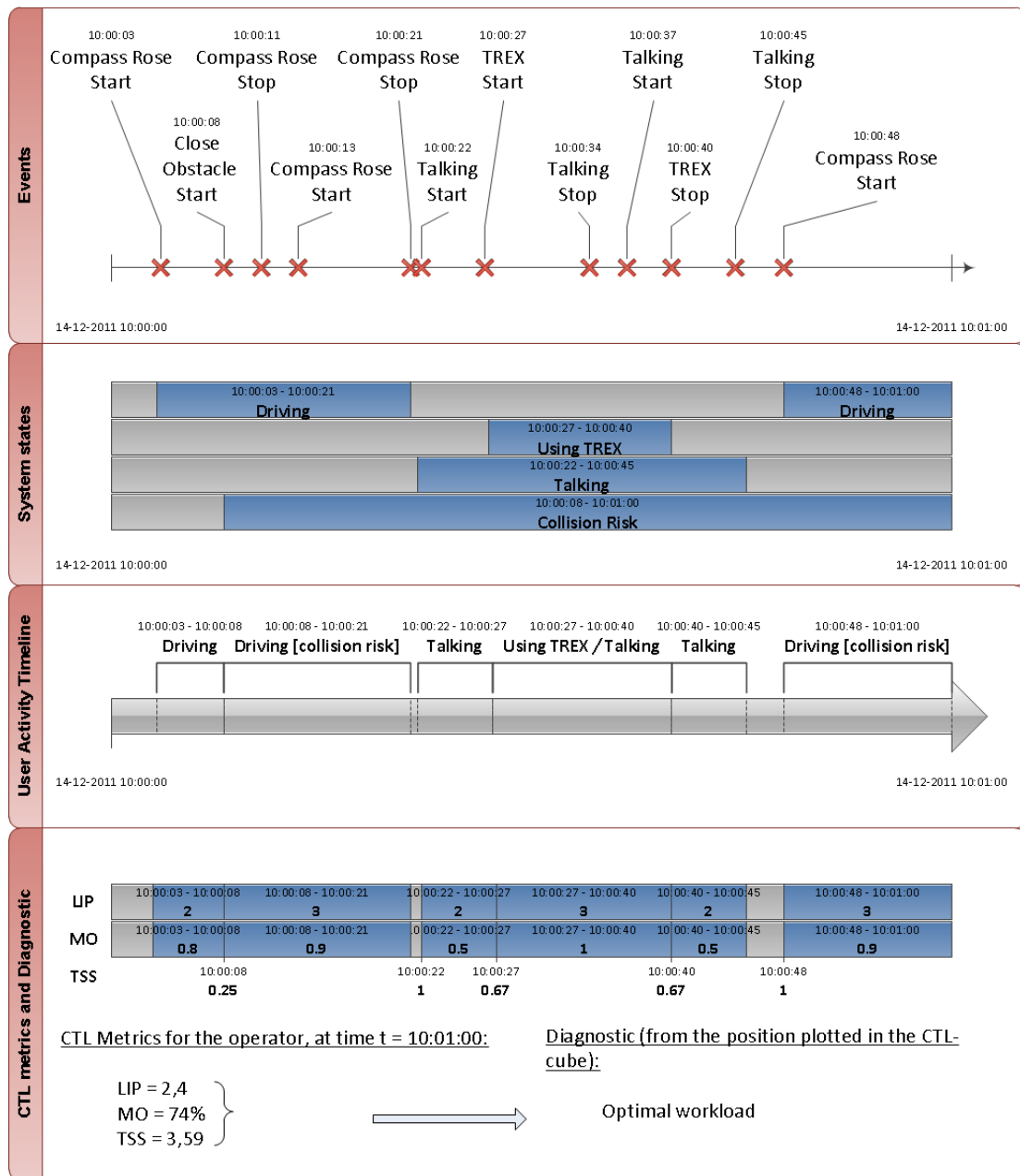


Figure 11: Overview of the steps Colin's [34] model takes to estimate the CTL of a robot operator in the USAR domain. For a detailed description of how the model calculates CTL the reader is referred to [34].

a task. Thirdly, the domains: these are the categories of mental representations an operator needs to execute a task. Domain information is used towards calculating how much workload is associated with switching between tasks (TSS). The task properties

can be made dependent on certain environmental properties, for example whether a robot is near an obstacle (see the modifier ‘Collision Risk’ in Figure 11). Modifiers (term used by Colin) correspond to environmental properties. If a task has an active modifier (e.g. ‘Collision Risk’), this means the task is situated in an environment for which this modifier is active (e.g. a cluttered environment). The task analysis we use for Colin’s model is shown in Table 4. We define two environmental properties that influence the task properties. Firstly, if an environment has much obstacles (*clutter*) avoiding them is a much harder task. Secondly, if there is much information coming in that needs to be processed (*busy*) it is harder to decide where to go. Using the combined task bar, and the input task analysis, the CTL of the operator can be calculated (Figure 11, third box from top). Note that the metric ‘Time occupied’ (TO) is replaced here by ‘Mental occupancy’ (MO) which represents the same metric, but is defined in a way more appropriate to the USAR domain.

range	LIP	MO	Domains
	0 to 3	0 to 1	
Nav ¹	1	0.5	Video feed
Nav ^{2,h}	2	0.8	Video feed, Tactical display
Obs ¹	1	0.5	Video feed
Obs ¹ [<i>clutter</i>]	2	0.8	Video feed, Laser map, Robot shape
Obs ^{2,h}	1	0.3	Video feed
Obs ^{2,h} [<i>clutter</i>]	2	0.6	Video feed, Laser map, Robot shape
Vic ¹	1	0.4	Video feed, Tactical display
Vic ^{2,h}	1	0.2	Video feed, Tactical display
Info ¹	2	0.5	Tactical display
Info ¹ [<i>busy</i>]	2.5	0.7	Tactical display
Info ^{2,h}	2	0.4	Tactical display
Info ^{2,h} [<i>busy</i>]	2.5	0.6	Tactical display

Table 4: The configuration for the CTL model.¹Note that this configuration is an estimation, based on research by Colin et al. [34]. The configuration should be tweaked/validated for each different scenario the model is used for.

We can use Colin’s CTL model to give a prediction of the CTL an operator would have given a certain reallocation of tasks. To do this, we need to predict the temporal behavior of the set of tasks allocated to the operator (when is he executing which tasks) over the upcoming two minutes. We can predict the temporal behavior of a single task, by copying the temporal behavior this task had in the most recent past two minutes it was allocated to the operator.² We can predict the temporal behavior of a set of tasks, by combining all the predictions for the separate tasks. This method of prediction is

¹We have only included tasks that are appropriate for human actors to execute, as CTL is only applicable to humans.

²Using (events gathered from training data as) default temporal behavior to be used if there is no previous allocation of this task to the operator

quite coarse, but as we are only talking about a very short time period, it is likely to be a usable estimate. Further research could refine the prediction of temporal behavior by using a task simulator. This task simulator could simulate how a task is likely to behave temporally, taking into account previous temporal behavior but also the context of the previous occurrence of the task in relation to the future context. For example, the temporal behavior of a task is likely to depend on which other tasks are allocated to the operator at the same time.

If we want to use the CTL model to know whether an operator’s performance is negatively influenced by his CTL, we also need to estimate the values of the problem region boundaries of CTL. As mentioned in Section 4.3.2, these can be dependent on the specific actor and on time. For the current instantiation, we approximate the problem region boundaries with constant values, independent of time and specific actors. These approximated values are obtained by observing human actors and the value of their CTL while executing the task to explore an area (in the set up as described in Section 5.2) and can be seen below:

$$\begin{aligned} LIP_{\text{low}} &= 1, & LIP_{\text{high}} &= 1.8, \\ MO_{\text{low}} &= 0.6, & MO_{\text{high}} &= 0.75, \\ TSS_{\text{low}} &= 4, & TSS_{\text{high}} &= 7 \end{aligned}$$

5.2 Method

In this section, we describe the experiment used to validate the model. This experiment uses the instantiated model as described in the previous section (Section 5.1). Section 5.2.1 describes the design of the experiment, including the hypotheses we aim to validate. Section 5.2.2 describes the participants we used for the experiment and Section 5.2.3 describes the task these participants had to execute. In Section 5.2.4, we cover the materials that were used and the set-up. Section 5.2.5 covers the procedure. Lastly, the measurements are detailed in Section 5.2.6.

5.2.1 Design and Hypotheses

We aim to validate our model with an experiment. The model reallocates tasks as to improve performance of the robot-operator team using the cognitive task load of the operator. A first step towards validating that the model actually improves team performance, is making sure the model works as we expect it to work. The model should reallocate tasks when the cognitive task load of the operator becomes problematic and the task reallocation should be such that this problem is resolved. Our experiment thus aims to validate the following hypotheses about the task reallocation model:

Hypothesis 1: The model reallocates tasks at the right moment.

Hypothesis 2: The model chooses appropriate reallocations.

These two hypotheses are very general and not easily validated using objective data. Therefore we add a third, more specific hypothesis:

Hypothesis 3: The real shift in CTL corresponds to the predicted shift in CTL.

With each task reallocation, the model aims to change the CTL of the participant and hereby relieve the problem state he/she is in. The model chooses a new task allocation based on the shift in CTL it predicts to arise from switching to this new task allocation. With this hypothesis, we aim to validate that the predicted shift in CTL is actually achieved by the task reallocation.¹ This third hypothesis can be validated using objective data. Hypothesis three is a subhypothesis of hypothesis 2: if the predicted shift in CTL corresponds to the real shift in CTL, this helps towards choosing appropriate task allocations.

5.2.2 Participants

A total of 15 runs of the experiment was conducted, each with a different participant. The data of two participants was not usable for analysis, so it was excluded. Data of thirteen participants was used for analysis. These thirteen participants included twelve males and one female, aged 21 to 38 (average 25,54; standard deviation 4,86). Ten participants had at least a higher vocational education². Participants indicated to spend an average of 0 to 40 hours a week playing video games (average 7,88; standard deviation 10,68) and an average of 0 to 10 hours a week driving a vehicle (average 2,04; standard deviation 2,80). Most participants had a little experience with remote controlled vehicles/non-autonomous robots (range from 1 (no experience) to 5 (much experience), average response 2,31; standard deviation 0,82). Most participants had no experience with autonomous robots, some participants had much experience with autonomous robots (range from 1 (no experience) to 5 (much experience), average response 1,96; standard deviation 1,42).

5.2.3 Task

The participant is given the role of robot operator and is asked to execute a typical USAR task (as explained in Section 5.1). The task is to explore an office building after an earthquake. He is told to cooperate with a robot to map the situation in the building. If any victims or large obstacles are seen, these should be added to a tactical display. The participant is told to hurry, as victims might be in need of medical attention and it is not safe for medics to go inside the building until the situation is fully mapped. Participants get 15 minutes to explore the building. A timer counting down from 15 minutes to zero is displayed to remind participants of the time pressure. To motivate participants, (they are told that) a small present is rewarded to the participant that performs best on the task.

¹Note that we could also just check whether the task reallocation relieved the problem state. However, this can be overly coarse. For example, the model could aim to relieve underload by increasing all three metrics with a task reallocation. If only one of the three metrics is actually increased by the task reallocation, the problem state will still be resolved, but the model does not work entirely as expected.

²HBO

The participant is told to do the tasks that are allocated to him by the task allocation model. He is also explained that the task allocation can be changed at any time during the experiment, namely if the model thinks he is too busy or not busy enough to function optimally. The subtasks and their possible level of autonomy variants that used are those described in Section 5.1 except for two small adjustments. Firstly, the task of detecting and avoiding obstacles at level of autonomy two (robot warns for obstacles) was excluded due to technical difficulties. Secondly, the combination of navigation at level of autonomy one (tele-operation) with victim detection at level of autonomy two (robot suggests possible victims) was excluded as it turned out to result in the exact same behavior as the combination of navigation at level of autonomy one (tele-operation) with victim detection at level of autonomy one (no victim detection). The (Dutch) description of the tasks, as given to participants, can be found in [Appendix 4](#).

5.2.4 Materials and Set-up

To execute the experiment, several materials are needed. Firstly, an implemented version of the instantiated model as described in Section 5.1 is needed. Secondly, an autonomous robot is needed that can execute the tasks as we described them in Section 5.1.2. Thirdly, an environment for the task is needed, namely an office building that has been damaged by an earthquake. Lastly, a tactical display is needed through which information about possible places of interest/dangers/victims can be communicated. The remainder of this section describes which materials were used for the experiment.

Implementation of Model The model was implemented in C++ using Microsoft Visual C++ 2010 Express. A high level pseudo code description of how the model was implemented is given in [Appendix 5](#).¹ As input, the model needs to know which tasks are executed at what time. We use a human observer to keep track of this. A screenshot of the interface this observer uses to keep track of the tasks the operator is executing is shown in [Figure 12](#). Buttons can be pressed to indicate that a task is started or stopped by the operator. The output of the interface serves as input for constructing a timeline that indicates which tasks are executed at what time. This timeline is the input for the CTL model which uses it to calculate the current CTL of the robot operator. The current CTL of the operator is then used as trigger for the task reallocation model. If the current CTL is in a problem region, the task reallocation model is run. If the tasks are reallocated, the new task allocation is communicated to the robot and its operator. The task reallocation model also receives the timeline with tasks as input and uses it to predict future temporal behavior of tasks.

¹Some parts of the model were not implemented as we did not need them for this scenario. We did not implement the pruning process. Pruning was done manually, carefully following the rules of the model. As the set of actors and tasks was small and constant, we manually inputted the pruned set of options. This also relieved the need to use a dummy actor. Furthermore, we did not implement a set partitioning problem optimization algorithm, as a brute force solution (explore all options and choose the best) was sufficient for this small set of actors.

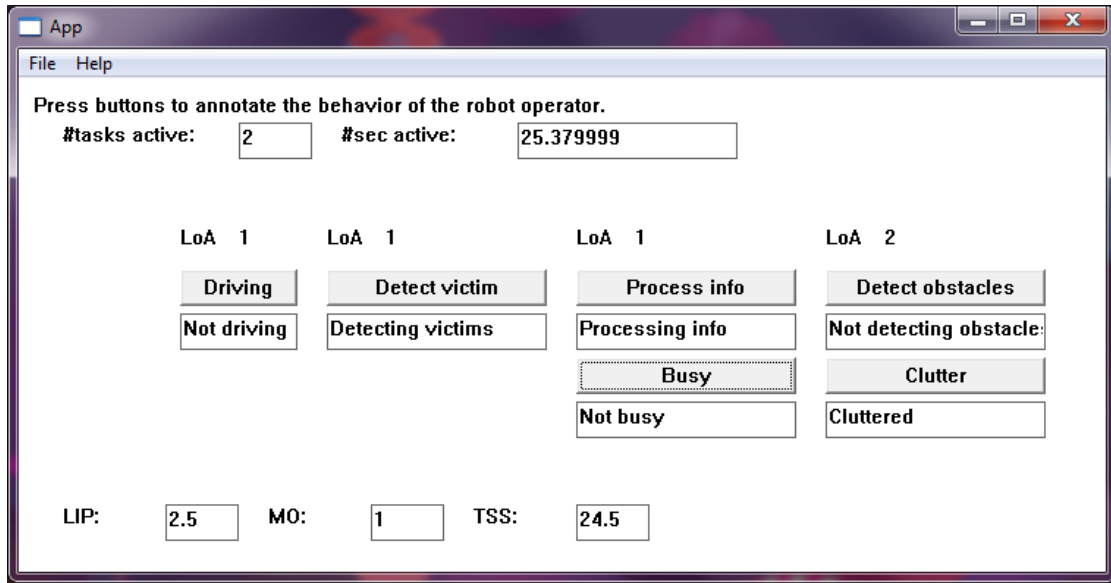


Figure 12: A screenshot of the interface used to keep track of the tasks the operator is executing. This serves as input to construct a timeline, which is used to calculate the current CTL of the operator and to predict future temporal behavior of tasks. The current CTL of the operator is shown at the bottom of the screen, the current level of autonomy (LoA) at which tasks are allocated to the operator is shown just above the buttons. In the top of the screen the number of active tasks and the number of seconds the interface is running is shown.

Environment To simulate a USAR mission, USARsim¹ is used. Using USARsim we create a virtual environment that represents a office building after an earthquake. We can create a virtual robot that can be navigated through the environment. Some screenshots of the USARsim environment are seen in Figure 13. Three similar, but slightly different virtual environments were created plus a simple training environment.

Tactical Display As tactical display, the TrexCOP application (Trex) [56] is used. The Trex system can be used to map information about the environment, for example so that other rescue workers know where victims/dangers and obstacles are located. A screenshot of the TrexCOP application is seen in Figure 14.

¹<http://usarsim.sourceforge.net/>



(a) The robot in a hallway



(b) A victim



(c) Another victim

Figure 13: Some screenshots of the USARsim environment.

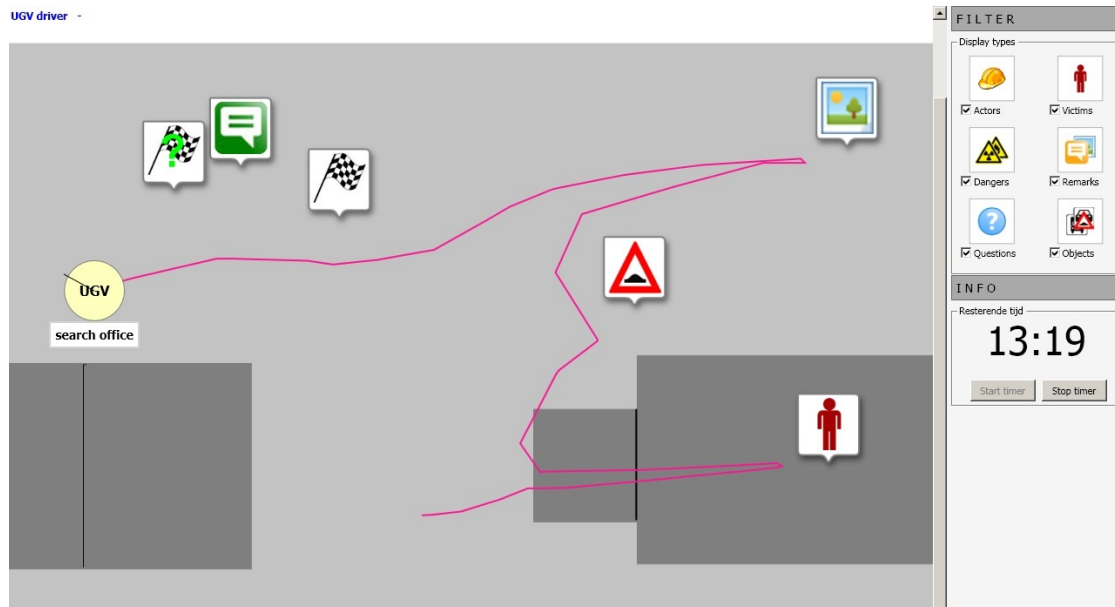


Figure 14: A screenshot of the TrexCOP application. The robot is seen on the left, depicted by the yellow circle labeled “UGV - search office”.

Autonomous Robot To simulate the autonomy of the robot, a second human operator (Wizard of Oz) which is located in a separate room controls the robot when tasks are allocated to the robot. The first robot operator (test subject) is convinced that the robot is capable of autonomous action and does not know the robot autonomy is actually faked by a second human operator.

Set-up The set up of the experiment can be seen in Figure 15. The robot operator (participant) has two computer screens, one showing camera images coming from the robot (USARsim client), the other showing the tactical display (TrexCOP). The robot itself (thus the Wizard of Oz) also has access to two screens, one showing camera images from the robot (USARsim client), the other showing the tactical display (TrexCOP). The observer watches the robot operator (participant) and keeps track of the tasks he is executing. If tasks are reallocated, the new allocation is forwarded to the USARsim server and shown to the participant and the Wizard of Oz via their USARsim client.

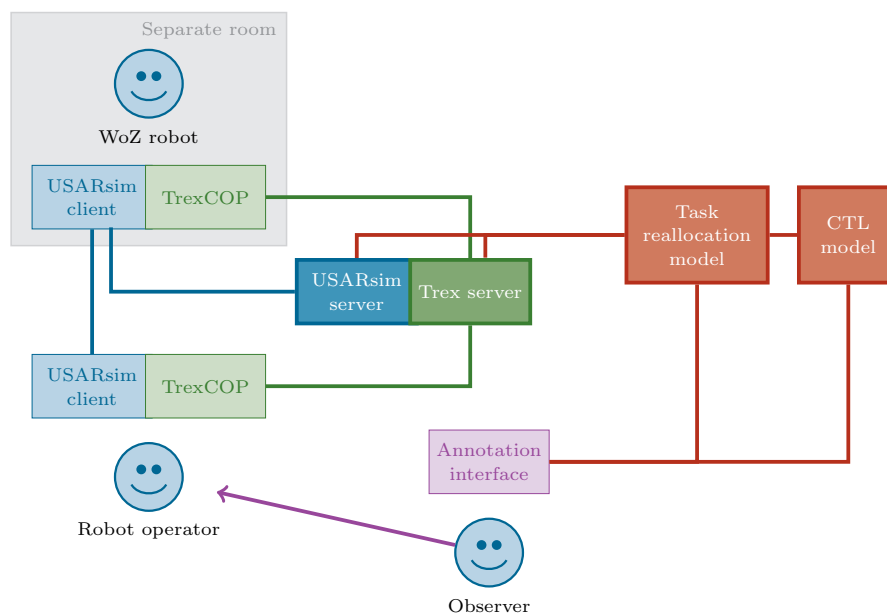


Figure 15: Overview of the experimental set up. In the lower left corner the robot operator is shown and in the upper left corner the second robot operator (Wizard of Oz) is shown. The third person that is shown in the figure is the observer.

5.2.5 Procedure

Every run of the experiment followed the same procedure. Firstly, the participant was asked to sign a consent form and fill in a questionnaire containing some general questions. This (Dutch) questionnaire can be seen in Appendix 6. After this, the task was explained and the participant was allowed some training with the USARsim and Trex environment using a small training level. After the training, there was a 5 minute break and time for

the participant to ask any questions he might have about the task. When the participant understood everything, the first USARsim level was started. The participant was given 15 minutes to find as many victims in this level as possible. After these fifteen minutes, a five minute break was given. This process was repeated with the same task for two similar, but slightly different USARsim levels. Each participant thus completed three fifteen minutes runs of the ‘explore an office building’-task. Finally, after the third level participants were asked to complete another questionnaire. This (Dutch) questionnaire is about their experiences during the task and can be seen in [Appendix 7](#).

5.2.6 Measurements

To test the hypotheses we stated in Section 5.2.1, we take several measures during the experiment. Hypothesis 1 and 2 are validated using a subjective measure, namely a questionnaire. A questionnaire is given to the robot operator after the experiment that contains questions concerning how well the model reallocated tasks, this (Dutch) questionnaire can be seen in [Appendix 7](#). The first six questions are aimed at validating hypothesis 1. Questions 7 to 12 are aimed at validating hypothesis 2. Each question asks participants to what extent they agree with some statement. Some of these statements are positively formulated, for example statement 1: “Whenever I felt too busy, tasks were reallocated”. For these statements a high answer (extent to which participants agree with the statement) corresponds to a validation of the hypothesis related to that statement.¹ To ensure participants carefully read all statements, not all statements were positively formulated. In this way, we prevent participants from being able to fill in a similar answer six times in a row. Some statements were negatively formulated, for example statement 2: “Tasks were reallocated too often.”. For these statements a *low* answer (extent to which participants agree with the statement) corresponds to a validation of the hypothesis related to that statement.

Hypotheses 3 is validated using more objective data which is saved output of the CTL model and task reallocation model. To validate hypothesis 3 we check whether the real CTL (saved output of CTL model) reacts to the tasks reallocations as expected based on predicted CTL (saved output of the task reallocation model). We do this by checking if the difference between the predicted CTL for the old task allocation and the predicted CTL for the new task allocation is the same as the difference between the real CTL before the reallocation and the real CTL after the reallocation. For example, if the task reallocation model reallocates tasks as to lower the level of information processing of the operator, we should see the real level of information processing of the operator decrease after the tasks are reallocated.

¹In this example this means if participants strongly agree with the statement “Whenever I felt too busy, tasks were reallocated.” this is an indication that they think the model reallocates tasks at the right moment.

5.3 Results

In this section, we describe the results of the experiment. Firstly, some general results are detailed in Section 5.3.1. After this, the results concerning the validation of the hypotheses (as described in Section 5.2.1) are given. Section 5.3.2 covers hypothesis 1, Section 5.3.3 covers hypothesis 2 and finally hypothesis 3 is covered by Section 5.3.4. A graphic representation of the CTL of the participants and the reaction of the model, separately for each run of the experiment can be found in Appendix 8.

5.3.1 General Results

Data of thirteen participants was complete enough to be used for analysis. Twelve participants completed all three levels, one participant completed only the first level. Data of 37 sessions could thus be analyzed. A total of 42 task reallocations occurred, an average of 1,14 (standard deviation 0,78) per session. Task reallocations that occurred during the last two minutes of a session are not included in this total and are not used for analysis. This is because we need at least two minutes of cognitive task load data with the new task allocation to be able to analyze whether the task reallocation had an effect on the cognitive task load.¹ At the beginning of Appendix 8 an overview is given as to which task allocations occurred during the missions and what cognitive states participants were in.

Seven general statements were given to participants after the experiment. Participants were asked to indicate how much they agree with these statements. Answers could range from 1 (Strongly disagree) to 5 (Strongly agree). The statements² and responses to them will be discussed below.

“Sometimes I felt too busy.” The average response to this statement was 2,38 (standard deviation 1,15). This implies that participants did not feel very busy. This result complies with CTL data, which indicates that overload was hardly experienced (< 1% of the time). Cognitive lock-up was experienced 15% of the time and this likely explains why (some) participants felt busy sometimes.

“Sometimes I felt as if I could do more tasks.” The average response to this statement was 3,08 (standard deviation 1,33). This implies participants did not often feel they could do more tasks. This result complies with CTL data as underload was only experienced 2% of the time (and vigilance was never experienced).

“I think I did not have enough training to react appropriately to the task reallocations.” The average response to this statement was 2,08 (standard deviation 1,21). So most participants felt they have had enough training (approximately a half

¹Cognitive task load is calculated over a time-frame of two minutes so it needs some time to react to the reallocated tasks.

²All statements/questions were given in Dutch and are translated. Original formulations can be found in Appendix 7, question 13 to 22.

hour) to react appropriately to the model. Overall, the training indeed seemed quite sufficient.

“I think the task reallocation model could have a positive influence on task execution if I would have more time to train/practice.” The average response to this statement was 3,00 (standard deviation 1,57). So, participants responded mostly neutral to this statement.

“I trusted the robot and his ability to execute the tasks assigned to him.” The average response to this statement was 3,54 (standard deviation 1,45). So, participants mostly trusted the robot.¹

“I felt I was better at mosts tasks than the robot.” The average response to this statement was 3,08 (standard deviation 1,38). So, on average participants felt their capabilities were similar to the robot’s. As the task allocation model does not take capabilities into account, the aim was to let the robot be as capable as the participant (such that capability differences do not influence results).

“I was convinced the robot was really autonomous (not controlled by a human).” The average response to this statement was 4,00 (standard deviation 1,28). So, most participants believed the robot was autonomous (when the task allocation allowed it to be autonomous). All participants were surprised to find out afterwards that the robot was actually controlled by a human sitting in the adjacent room.

Two open ended question were given to participants.² Firstly, participants were asked to state three positive aspects of the task reallocation model or the experiment in general. About half of the responses were about the experiment in general and conveyed that people thought the training was clear and useful, liked executing the task, thought controlling the robot was easy/fun and thought the multiple screen display was very useful. The other half of the responses mostly conveyed that people liked the idea of automating some of the tasks. Some of the most positive comments for the current research were that participants said the model releases stress in stressful situations, the model recognized when it was a good moment to automate the task of detecting obstacles and the model found a good task allocation which was constant over all three levels.

Secondly, participants were asked to state three negative aspects of the task reallocation model or the experiment in general. Some of the most common responses to this question are summarized below:

¹When participants did not trust the robot, automation was often accompanied by participants constantly checking whether the robot was doing what is was supposed to do. This causes extra workload and diminishes the positive effect of automation. Luckily, only a small number of participants showed this behavior.

²There was actually a third open ended question that asked participants whether they had any other comments, but hardly any participant had any comments left after the first two open ended questions.

- Some participants found way-point driving to be frustrating. Most of these participants disliked that they could not turn the camera during way-point driving. Some participants disliked that they could not add direction (for the robot to face) to way-points or that they could only set one way-point at a time.
- Most participants indicated that some sort of alarm should be added to indicate when the task allocation changes. Participants often did not notice when tasks were reallocated. This was solved by letting the experimenter pay extra attention to this and vocally attending participants to a change in task allocation.¹
- Some participants wanted control over the task reallocations or wanted the model to take their preferences into account. These participants really disliked that they could not decide/influence what tasks are automated and when.

5.3.2 Hypothesis 1: The Model Reallocates Tasks at the Right Moment

Six statements about timing were given to participants after the experiment. An overview of the average response (and standard deviation) to each statement is given in Table 5.

	Average response	Standard deviation
1. Whenever I felt too busy, tasks were reallocated.	1,62	0,74
2. Tasks were reallocated too often.	1,62	1,15
3. I understood why tasks were reallocated when they were.	1,92	1,21
4. Task execution would have improved if I could decide when tasks were reallocated myself.	3,77	1,37
5. Sometimes I felt bored, but tasks were not reallocated.	2,85	1,35
6. The moment at which tasks were reallocated never surprised me.	2,62	1,15

Table 5: Responses to statements concerning timing. Answers could range from 1 (Strongly disagree) to 5 (Strongly agree).

Cronbach’s alpha was used to check the internal consistency of these six statements. Firstly, responses to negatively formulated statements such as “Tasks were reallocated too often” (2,4 and 5 in Table 5) were negated² before using them for analysis. Cronbach’s alpha over all responses to the six statements was 0,607. This is quite low, but as the concept of timing is rather broad and we use only six statements we should not expect very high Cronbach’s alpha levels. Cronbach’s alpha can be improved by removing statement 1 (+0,081) or removing statement 6 (+0,098).³ As both these statements

¹Participants most likely still indicate this as an issue (although it was solved) as their task description stated that they should watch the task allocation themselves.

²By negating we mean answer 1 was mapped to 5, 2 to 4, 3 to 3, 4 to 2 and 5 to 1.

³Removing both statements 1 and 6 would yield a Cronbach’s alpha of 0,797 and cannot be further improved.

seem clear and integral to the hypotheses (and the effect of removing them on Cronbach’s alpha is not very large¹) we decide to not exclude these statements for analysis.

To validate hypothesis 1, the average response over all six statements about timing was calculated, using the negated response where necessary. The value obtained in this way describes to what extent participants think the model reallocated tasks at the right moment. Averaged out over all participants, this value is 2,65 (standard deviation 0,68). The 95% confidence interval ranges from 2,30 to 3,01. Participants are thus quite neutral (a bit towards the negative) about the timing of the model. We can not validate, based on this data, that the model reallocates tasks at the right moment. Conversely, we can also not say the timing of the model was fully off, as participants were quite neutral. This is quite a good result when taking into account that participants did not feel very busy/bored at all.

5.3.3 Hypothesis 2: The Model Chooses Appropriate Reallocations

Six statements about the appropriateness of reallocations were given to participants after the experiment. An overview of the average response (and standard deviation) to each statement is given in Table 6.

	Average response	Standard deviation
1. Whenever I felt too busy, the level of autonomy of the robot <i>increased</i> on some tasks.	1,77	0,58
2. If there would have been no task reallocations, task execution would have gone worse.	1,85	0,86
3. The task reallocations helped improve task execution.	2,77	1,48
4. Sometimes the level of autonomy of the robot <i>decreased</i> on some tasks, while I was not bored.	1,69	0,91
5. The task would have gone better if I could decide what tasks to reallocate myself.	3,85	1,35
6. I always understood why a specific reallocation of tasks was chosen.	1,96	0,97

Table 6: Responses to statements concerning appropriateness of reallocations. Answers could range from 1 (Strongly disagree) to 5 (Strongly agree).

Cronbachs alpha was used to check the internal consistency of these six statements. Responses to negatively formulated statements such as “I would have performed better if I could decide what tasks to reallocate myself” (4 and 5 in Table 6) were negated before using them to calculate the average over all statements. Cronbach’s alpha over all responses to the six statements was 0,510 which is quite poor, meaning the internal

¹The effect is very large for removing both statements, but removing one third of the statements seems too rigorous when no clear reason to do so besides internal consistency can be found.

consistency of the six statements is low. Removing statement 4 improves Cronbach's alpha to the more acceptable 0,694. Closer inspection reveals that statement 4 should indeed be excluded from analysis as the level of autonomy of the robot did not decrease in any of the sessions¹. Cronbach's alpha can be further improved,² but this is not done since the remaining five statements seem clear and integral to the hypotheses (and either the effect of removing them on Cronbach's alpha is not very large or half of the statements is removed).

To validate hypothesis 2, the average response over statements 1,2,3,5 & 6 about appropriateness of the reallocations was calculated, using the negated response where necessary. The value obtained in this way describes to what extent participants think the model chose appropriate task reallocations. Averaged out over all participant, this value is 1,71 (standard deviation 0,39). The 95% confidence interval ranges from 1,71 to 2,49. Participants are thus quite negative about the appropriateness of the reallocations. We cannot validate, based on this data, that the model chooses appropriate reallocations. Conversely, we *can* say participants think the model *does not* choose appropriate reallocations.

5.3.4 Hypothesis 3: The Real Shift in CTL Corresponds to the Predicted Shift in CTL

For each task reallocation, the real shift in CTL was compared to the predicted shift in CTL. The shift might be different for all CTL metrics. Therefore, the comparison was done separately for the three metrics. We do this by checking if the difference between the predicted CTL for the old task allocation and the predicted CTL for the new task allocation is the same as the difference between the average real CTL in the two minutes before the reallocation and the average real CTL in the two minutes after the reallocation. This difference is calculated by subtracting the value for the new task allocation from the value for the old task allocation. Task reallocations that occurred in the last two minutes of a session were excluded (since the average real CTL in the two minutes after the reallocation can not be computed).

Firstly we checked whether the predicted CTL for the old task allocation corresponds to the average real CTL in the two minutes before the reallocation and whether the predicted CTL for the new task allocation corresponds to the average real CTL in the two minutes after the reallocation by calculating correlation coefficients. This was done separately for all three metrics and should give high correlation coefficients as the measures are straightforwardly corresponding³. We check the correlation coefficients

¹This is probably because sessions always started with all tasks at level of autonomy 1 and they only lasted 15 minutes, so participants were unlikely to experience underload.

²Removing statement 4 and 1 yields an Cronbach's alpha of 0,726; removing statement 2, 4 and 1 yields (the highest possible) Cronbach's alpha of 0,797.

³They are not the same however, as the predicted variants are about a single two minute time-frame, while the real variants are the average over two minutes. Each single estimation this average is calculated over is again based on a two minute time-frame. One of these time-frames is the same as the two minute time-frame the predication was based on. Another one has 115 seconds overlap, another one has 110 seconds overlap and so on until the last time-frame which had no overlap.

here, as the existence of these correlations is a precondition for existence of a correlation between the real and predicted shifts in CTL. Scatter plots of the results are shown in Figures 16, 17 and 18. All predicted metrics are indeed significantly ($p < 0,05$) correlated to their real counterparts, correlation coefficients range from 0,45 to 0,79. The best result is seen for mental occupancy (MO).

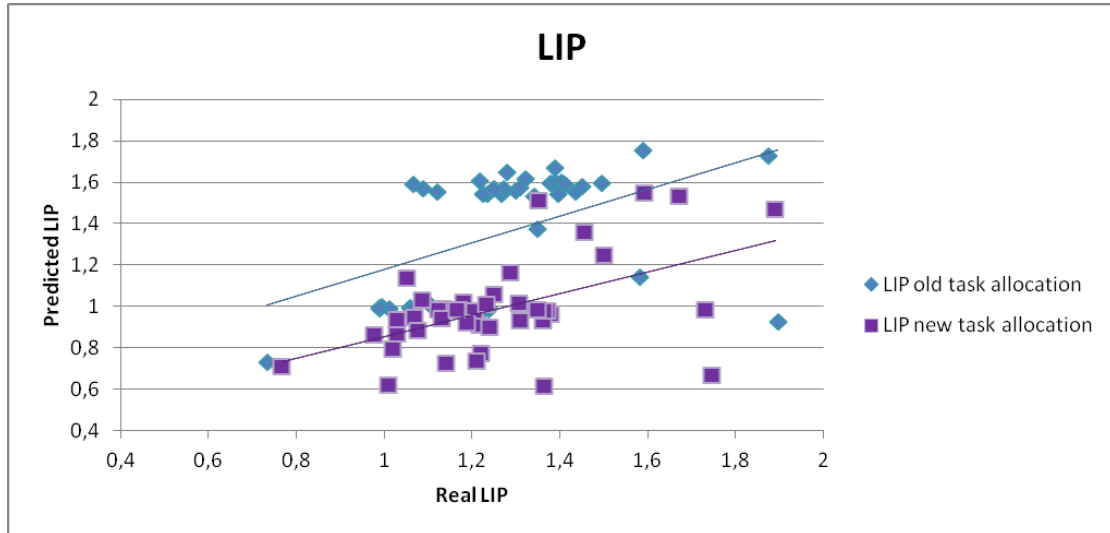


Figure 16: Scatter plot showing the relation between real and predicted LIP values. Each data point corresponds to the moment of a task reallocation. Both the relation between the real and predicted LIP values of the old task allocation (blue) and of the new task allocation (purple) of this reallocation are shown. The correlation coefficient for the old task allocation real and predicted LIP values is 0,47 ($p < 0,01$); the correlation coefficient for the new task allocation real and predicted LIP values is 0,52 ($p < 0,01$).

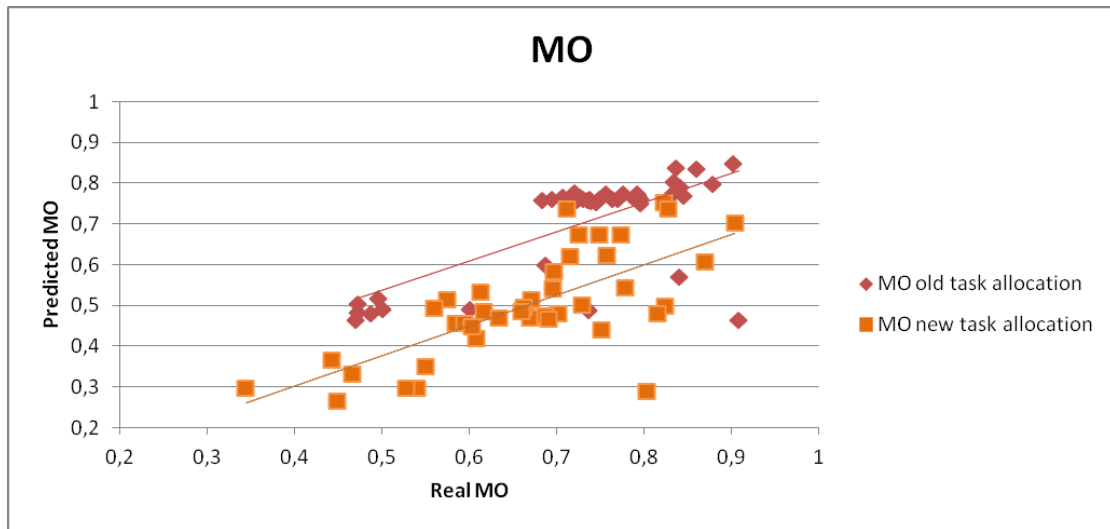


Figure 17: Scatter plot showing the relation between real and predicted MO values. Each data point corresponds to the moment of a task reallocation. Both the relation between the real and predicted MO values of the old task allocation (red) and of the new task allocation (orange) of this reallocation are shown. The correlation coefficient for the old task allocation real and predicted MO values is 0,69 ($p < 0,01$); the correlation coefficient for the new task allocation real and predicted MO values is 0,71 ($p < 0,01$)

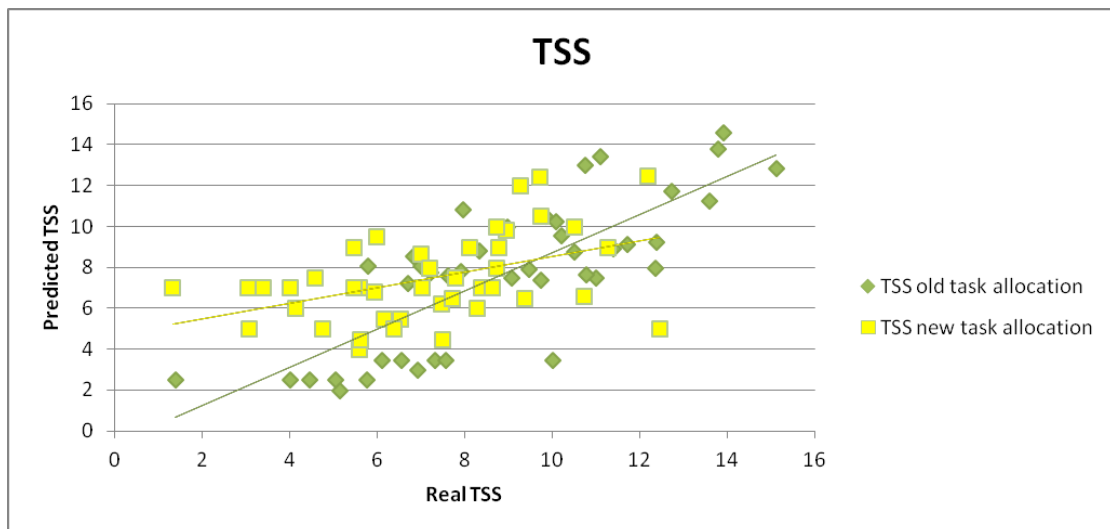


Figure 18: Scatter plot showing the relation between real and predicted TSS values. Each data point corresponds to the moment of a task reallocation. Both the relation between the real and predicted TSS values of the old task allocation (green) and of the new task allocation (yellow) of this reallocation are shown. The correlation coefficient for the old task allocation real and predicted TSS values is 0,79 ($p < 0,01$); the correlation coefficient for the new task allocation real and predicted TSS values is 0,45 ($p < 0,01$).

To validate hypothesis three, we check whether the predicted shift in CTL corresponds to the real shift in CTL obtained by the task reallocations. We do this separately for each metric. These shifts in CTL are not straightforwardly corresponding. Even if the predicted and real CTL levels would be highly correlated, it does not necessarily follow that the shifts are also highly correlated as they cover exactly the hardest part of the CTL to be predicted: the change that occurs as a result of the reallocated task(s). If the predicted and real CTL levels are not correlated, it *does* follow that the shifts are very unlikely to be correlated. As we have found lower correlation coefficients for LIP and TSS than for MO, the shifts in LIP and TSS are likely less correlated than the shifts for MO. Scatter plots of the results are shown in Figures 19, 20 and 21. The correlation coefficients are 0,32 ($p < 0,05$) for LIP; 0,43 ($p < 0,01$) for MO and 0,29 ($p = 0,06$) for TSS. We indeed see that the correlation coefficient for MO is the highest. The correlations for LIP and MO are significant ($p < 0,05$), the correlation for TSS is not. Based on this data, we cannot validate hypothesis 3. We *can* validate that the LIP and MO respond to the task reallocations as the model predicts.

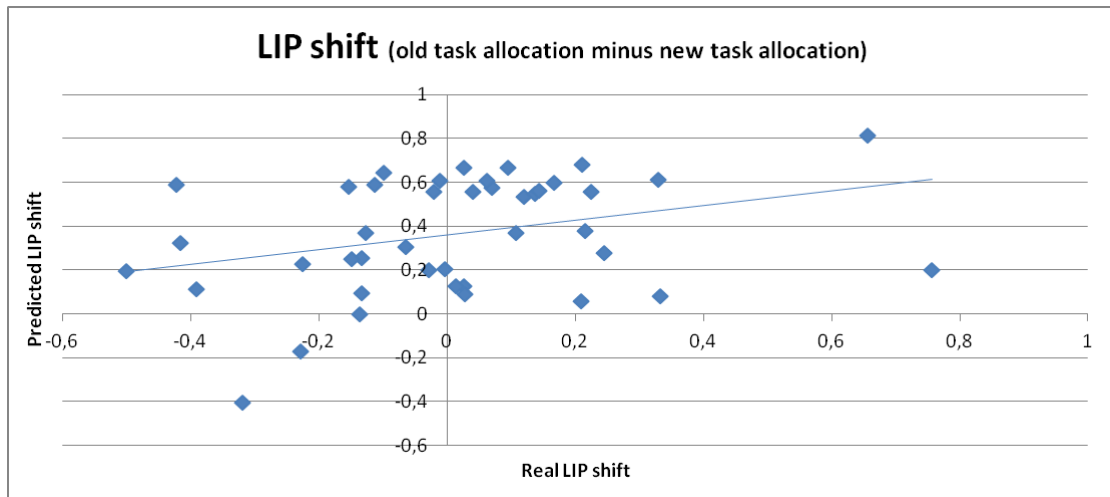


Figure 19: Scatter plot showing the relation between the real and predicted shift in LIP caused by a task reallocation. Each data point corresponds to the moment of a task reallocation. The correlation coefficient for the real and predicted shifts in LIP is 0,32 ($p < 0,05$).



Figure 20: Scatter plot showing the relation between the real and predicted shift in MO caused by a task reallocation. Each data point corresponds to the moment of a task reallocation. The correlation coefficient for the real and predicted shifts in MO is 0,43 ($p < 0,01$).

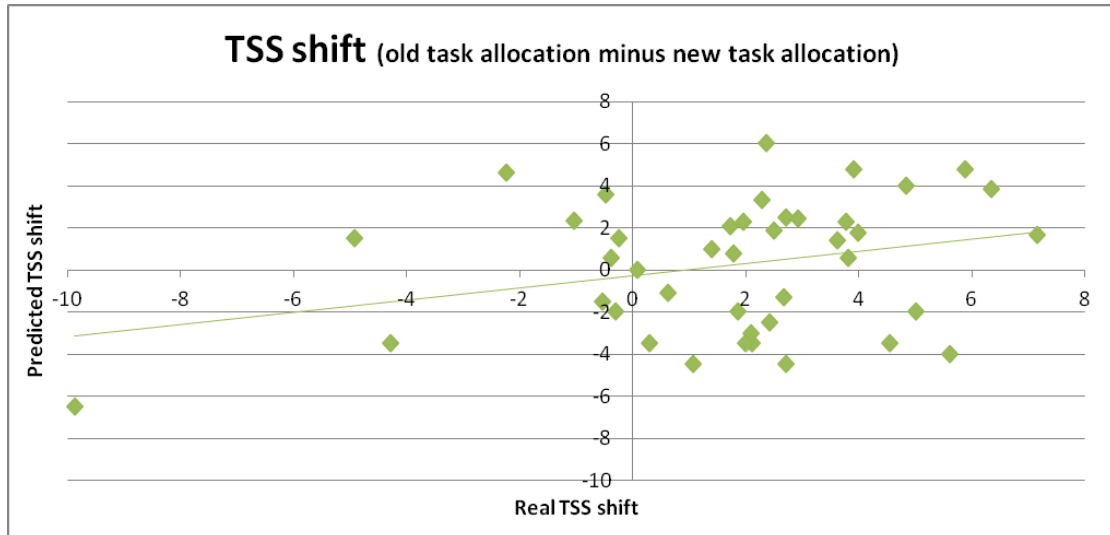


Figure 21: Scatter plot showing the relation between the real and predicted shift in TSS caused by a task reallocation. Each data point corresponds to the moment of a task reallocation. The correlation coefficient for the real and predicted shifts in TSS is 0,29 ($p = 0,06$).

5.4 Conclusion (Experiment)

In conclusion, the experiment did not result in conclusive evidence that validates that the model reallocates tasks at the right moment and chooses appropriate reallocations (that benefit the cognitive task load of the participant). However, we did not find conclusive evidence that the model does *not* work as it should either and some results were promising. Mainly promising was that we found both mental occupancy and level of information processing to react to the task reallocations as intended (predicted) by the model. Furthermore, a lot of important and useful lessons were learned during the experiment. Some lessons are very specific and may only be useful for the specific instantiation of the model used in the current experiment. Other lessons are very important and are steps forward in developing the model in general. An overview of all lessons learned during the experiment is given below.

- To thoroughly test the full functionality of the model, we need to elicit all possible different CTL states. In the current experiment, we did not see enough variation in CTL. Even though we tried to make participants as busy as possible, overload states were hardly experienced. It is very hard to simulate urban search and rescue tasks with their high cognitive demands, especially in an virtual environment. Underload was also hardly experienced, longer term usage of the model and different starting task allocations are needed to elicit this.

Important to note is that the scarceness of overload and underload states was visible in both in the CTL states calculated by the model and in the responses of participants to the questionnaire. This compliance is a strong indicator that the model was working with the correct input (CTL estimation and problem region boundaries).

- Boundaries of problem regions should be configured separately for each user as it is very different per user what their optimal cognitive task load is. Also, upper and lower boundaries of the neutral zones of metrics should not be too close as this might cause unusual behavior.
- Coordination costs should not only be separately calculated per task, but should also include a general switching factor. This general switching factor should discourage the model to reallocate tasks shortly after each other, even if the two reallocations do not involve the same task. Behavior caused by not doing this is seen in [Appendix 8](#) (participant 1, level 2 and participant 10, level 3).
- Preference based on CTL should be more fine-grained. For example, relieving the problematic CTL state underload (all three metrics too low) by increasing all three metrics into their neutral zone should be preferred over relieving underload by increasing one metric to its neutral zone while decreasing the two other metrics. This makes it more likely that the underload is indeed resolved if the future temporal behavior of the tasks differs (a bit) from what was predicted. Behavior caused by not doing this is seen in [Appendix 8](#) (participant 3, level 3 and participant 4, level 2).

6 Discussion and Future Research

In this section we discuss whether the approach taken in this research was suitable for answering the main research question and its limitations. Furthermore, we detail important challenges that are not addressed in the current research and could be the subject of future research.

The following sections detail discussion points concerning each of the three subquestions (as stated in Section 1). Firstly, discussion points concerning the general framework for dynamic task allocation are detailed in Section 6.1. Secondly, we detail some discussion points concerning the model for adaptive automation in Section 6.2. Thirdly, discussion points concerning the experiment are detailed in Section 6.3.

6.1 Framework for Dynamic Task Allocation

We aimed at answering the main research question by building a general framework describing the important concepts that influence team performance and can be used to dynamically allocate tasks. This framework shows how to use dynamic task allocation to improve team performance. It does this on a very high level, by showing what concepts are important and how they can be formalized. For practical application general concepts do not suffice. Specific instantiations of these general concepts are needed. Every context (task and team) calls for different factors influencing the task allocation, or at least a different balance in how important each of these factors are. How to choose what specific factors are needed, is not apparent from the framework as it is not a process that can be generalized. However, the framework serves as a guideline for designing models and using it allows for easier identification and formalization of important factors. We showed this by designing a fully instantiated model based on the framework.

6.2 Model for Adaptive Automation triggered by Cognitive Task Load

We aimed at showing the applicability of the framework by designing a model for adaptive automation, triggered by cognitive task load. The concepts of the framework were easily applicable to the design of the model. Furthermore, using the framework as a guideline allowed for easy implementation and leaves room for expansion. For example, the current model only takes into account two factors when deciding on a task allocation, namely cognitive task load and coordination costs. It is quite straightforward however to imagine how multiple other factors could be included. More models should be built based on the framework to ensure its applicability is robust and flexible enough to deal with different factors. As the model did not limit itself to specific tasks and/or team compositions, flexibility on this point is likely to be sufficiently represented in the framework. Some points of discussion concerning the model are detailed below.

6.2.1 Trust in Model

To ensure human actors have no problems working with a dynamic task allocation model, reallocation surprises and trust issues need to be avoided [26]. During the experiment,

participants found it hard to trust the model and to have no control over the task allocation themselves. This trust issue caused negativity about the task reallocations. The participants were mostly young, male and university student, which might have had some influence on the results. Young male students, when compared to professionals, might have a much harder time accepting that they cannot decide for themselves what to do. Accepting losing some control might be easier for urban search and rescuers, as they are highly trained firefighters that are well adjusted towards hierarchical structures and obeying orders. Even so, the problem likely remains (even if diminished) and should be considered.

Making work agreements, like in [41], could also help improve trust. Work agreements are rules a robot operates under, agreed upon before starting a team task, giving the human operator room to restrict which tasks can be done by the robot(s) (at which level of autonomy), and possibly when. Work agreements can also give insight into what tasks actors can expect to be reallocated and possibly also why and when specific reallocations occur.

To further give actors insight and even some influence, we could adapt the level of automation of the task reallocation model itself. A hybrid approach might be most suitable, in which the level of autonomy of the task allocation model itself reacts to the cognitive task load of an actor. We could imagine having adaptive automation for high workloads and adaptable automation for lower workloads [32], meaning the model could be used to decide for high workloads and to suggest for low workloads. This gives an actor influence on the task allocation, only when he is not too busy to think about it. Furthermore, the model could show to the actor how it values the options it suggests. In this way, the actor can get more insight into how the model chooses a task reallocation which again benefits trust [54].

Future research using different participants, work agreements and hybrid models is needed to investigate the effects on trust in the model. Furthermore, future research is needed that specifically addresses how participant's trust in the model effects how useful the model is in terms of improving performance and how useful the model is perceived to be by the participant.

6.2.2 Factors in Choosing a Task Allocation

CTL is a very important factor in choosing a task allocation, but it is likely beneficial to include other factors. Furthermore, CTL as factor has some limitations itself. In this section, we discuss firstly the limitations of CTL and secondly the additional factors that might be needed.

Limits of CTL The CTL model used here is limited by only being able to reason with a specific sort of tasks. Namely, it needs tasks for which it is possible to automatically detect whether they are active or not. In the current experiment this was not a problem,¹

¹Note that we did not actually use automatic detection of task activity in the experiment, but a human observer. Research by Colin et al. [34] however shows for a very similar task that this process

but one could imagine a lot of domains where this would be a problem. For example, if an actor seemingly does nothing for some time, we cannot assume directly that he/she is indeed doing nothing that affects CTL. Thinking is a hard behavior to automatically detect, while it has a large influence on CTL. The seemingly vacant actor could really be doing nothing, then he might be in underload and we need to assign some tasks to him/her. Conversely, the actor might be extremely busy thinking about complex problems in which case we surely do not want to assign more tasks to him/her. Some sort of additional measure might be necessary to help estimate how busy someone is. For example, a biochemical measure such as skin conductance can be used to complement the CTL estimations. Skin conductance is shown to increase with cognitive load [57]. Including a measure for cognitive task load that covers less visible tasks would make the CTL estimation applicable to a much wider range of tasks. Additional research into CTL is needed to figure out the most robust way of measuring it.

Beyond CTL Two factors that are important when making task allocation decisions, besides CTL, were identified during the experiment and are discussed here. Firstly, it is very important to take the capability of an actor to do a task into account. This can be done by either making a static factor, using estimations made beforehand, or by using a real-time estimation of how good someone is doing. A static capability factor could be used to take into account capability differences of the different actors. These differences were minimized for the current experiment, but in real life situations they exist and are important to consider. A real-time estimation of an actors capability can also be very useful, imagine as example the task of obstacle avoidance. It would have been very beneficial for this experiment to include some estimation of how well a participant avoids obstacles, for example by counting the number of times he/she bumps into an obstacle. When the CTL of the participant becomes problematic, the estimation can be used to decide whether to automate the obstacle avoidance task (if the participant often bumps into obstacles) or some other task (if the participant hardly bumps into obstacles).

A second important additional factor is which tasks actors personally prefer to execute. Taking this into account could greatly benefit actors trust towards the model and reduce reluctance to accept its decisions, indirectly benefiting performance. It can also directly benefit performance as an actor is probably more likely to execute a task he likes doing well. How much the preference of an actor needs to be taken into account, relative to other factors, is not clear and might be a domain specific decision. The preference for tasks an actor might have, can again be estimated in different ways. We can include a static factor that needs to be configured beforehand (by asking the actor(s) which tasks they prefer) or we could estimate the preference real-time. This real-time estimation can for example be done if we implement a hybrid model that gives the actor some options of possible task allocations and lets the actor decide a task allocation from these options. Which task allocation is selected by the actor can give information about which tasks he prefers to execute, so the model can learn the preference of the actor without having to ask him/her beforehand.

can be automated.

Research is needed to investigate how to include actors' preference and capability as factors. Different strategies should be tested, both static and real-time estimation. This research should explore the effects on performance of including actors' preference and capability into the decision on which tasks to reallocate. Also, how much actors' preference and capability need to be taken into account, relative to the other preference factors (e.g. CTL) should be explored.

6.2.3 Configuration

The exact moment of a task allocation relies on the configuration of the CTL model. In the experiment we conducted, this was not configured separately for each participant and was thus likely always a bit off. Participants' opinion about the timing of the task allocation model will likely benefit from personalizing configuration of cognitive task load problem region boundaries.¹ Future research is needed to figure out how to incorporate personal configuration and to explore the effects.

Configuration poses an additional challenge. Results of experiments using task allocation models are often very hard to generalize as they highly depend on how all user and task dependent values are configured and this is very different for each domain. Furthermore, configuration takes a lot of time and effort. It is integral to the applicability of task allocation models that future research will invest a lot of effort into automatizing as much of the configuration process as possible. Ideally, models will need to become self-learning, adapting themselves to novel tasks and actors when needed.

6.2.4 Representation and Notification

The task allocation model as discussed focuses on how to determine the most suitable task allocation. We did not address how we should communicate this task allocation to the actors using the model. More research is needed to investigate how to keep all actors aware of which tasks are allocated to them. Furthermore, when tasks are reallocated, actors should be notified. It is very important that the representation of the task allocation and the notifications of changes are very clear as an unclear task allocation could hinder actors in executing their tasks. Future research into task reallocation models should include representation and notification design as it is an important part of designing usable models.

6.3 Experiment Urban Search and Rescue

We aimed at validating the effects of our model in an experimental setting. While the main research question is about improving performance, we chose to perform an experiment that checks whether the model works as expected in terms of improving the CTL of the operator, not whether the performance of the team actually improves. As explained in Section 2.3, the relation between CTL and performance is quite well

¹I.e. testing in advance what CTL levels are problematic for each different participant and configuring this in the model.

established by earlier research. If we can validate that our model improves the CTL of an operator, we can thus reasonably expect it will also help optimize performance. Experimentally validating whether the model works as expected in terms of relieving problematic CTL states is thus an important first step towards validating that the model improves performance.

The experiment did not use the full functionality of the model and further experiments are needed to generalize experiment results to conclusions about the effects of the model. For instance, experiments with the model reallocating tasks for teams with multiple humans and/or multiple robots and for different domains are needed. Until further experiments are conducted, the results of the current experiment cannot be seen as validation of the model's effects. However, as we did not make any assumptions that severely simplified validating the model's effects in our experimental design, the positive results obtained *can* be seen as an indication that we are moving in the right direction.

Some points of discussion concerning the experiment and directions for future research are detailed below.

6.3.1 Error in Prediction

Important to note is that an error was found in the code of the model after the experiment. It is not hard to fix this error, but as it was not found until after the experiment, the whole experiment was run with this error. The error consisted of the model failing to take into account the temporal behavior of modifiers when predicting the temporal behavior of tasks. More precisely, if a modifier was active/inactive at the exact time of the call to the model, it was predicted to be active/inactive during the whole future time frame. Depending on the exact situation, this causes under- or overestimation of all three metrics of the predicted CTL.¹ It is not clear to what extent this error has influenced the model's behavior, but it is clear that its predictions were further off than they should be. This has a negative influence on all results, but the magnitude of this influence is unclear and future research is needed. The most influence was likely on the prediction of task set switching. We can see in the results for validating hypothesis 3 (5.3.4) that task set switching is the only CTL metric for which a significant correlation between real and predicted shift could not be found. This observation indicates that it is likely that fixing the error will result in finding a significant correlation for all three metrics thereby validating hypothesis 3. This means the task reallocations likely do what they aim to do, relieve problematic CTL states, if we resolve the modifier prediction error. However, we can not be sure about this until further experiments are conducted using the model without the error.

¹In Table 4, an overview of the task analysis can be found. Tasks and their properties (that contribute to the CTL estimation) are shown for both the task in combination with an active modifier (modifier showed behind name of task in brackets, e.g. Obs¹[*clutter*]) and in combination with an inactive modifier (no modifier showed behind name of task, e.g. Obs¹). The effect of a modifier can thus be seen by comparing the two task properties (with active and inactive modifier).

6.3.2 Lack of Training and Frustration

Although participants indicated they thought training was sufficient, some task switches seemed quite hard for some participants and could likely have benefited from more training. The best example of this is switching from tele-operation to way-point driving. Way-point driving can save time, but only if participants actually do other tasks while the robot is driving towards the way-point. Some participants forgot to first set a new way-point before doing another task (for example filling in a victim report) which nullifies the time benefit. Furthermore, some participants were frustrated by way-point driving, mostly because they were unable to turn the camera. This can and should be improved to ensure easier and more efficient way-point driving. As the behavior of the model is very much influenced by the task analysis (what effect some task has on CTL) it is very important that the task analysis is correct. Frustration and lack of training increase the mental occupancy and likely the level of information processing a task requires and future research should make sure to avoid it.¹

6.3.3 Variation in CTL

Participants were mostly in a neutral CTL state during the experiment, hardly experiencing problem states (neither busy nor underloaded).

The fact that participants did not feel very busy is puzzling. The task was clearly too much work to do in the given time. For example, 32 victims had to be found in each level while participants found only 10 to 26. Furthermore, most participants seemed very motivated to do the task fast as they wanted to get a high score (as the highest scorer was promised a prize and a list of scores to beat was put up). The lack of feeling too busy (while they actually were too busy to do all tasks) most likely influenced participants' opinion on the model greatly. For example, participants mostly disagreed with statements as "Whenever I felt too busy, tasks were reallocated." as they hardly ever felt too busy. Future research should put even more effort into trying to get participants to experience business as it is integral to how useful the model is perceived to be. This could be done for example by adding even more tasks that have to be executed simultaneously or by making the scenario more realistic.²

The fact that participant did not feel underload/vigilance was expected as sessions always started with all tasks fully allocated to the participant (no autonomy of the robot) and lasted only 15 minutes. Future research should either have longer experiment sessions or vary the starting task allocation if they want to trigger more underload/vigilance.³

¹Or at least take it into account.

²It will probably be very beneficial to test the model during a non-virtual experiment.

³Which is necessary if the full functionality of the model is to be tested.

6.3.4 Short Term Interaction

The focus of this experiment was on short term interaction, while the model is likely also of use for longer term interaction. The model is built to deal with dynamic circumstances. These dynamics are not only short term dynamics (such as cluttered rooms versus empty rooms) but also (and maybe more importantly) longer term dynamics. These longer term dynamics for example include exhaustion/alertness and visibility (day/night). Future research is needed to investigate whether the model is indeed also of use for long term interaction.

6.3.5 Measuring performance

Ultimately the most important measure of how well the model works is team performance. How useful or good participants perceive the model to be is not a perfect measure of how useful the model actually is in terms of improving performance. Some difficult issues need to be tackled when setting up an experiment to test whether the model improves team performance. Firstly, it would need to be verified that the CTL as calculated corresponds to the real CTL of the robot operator. This does not only involve verifying Colin's model,¹ but also the specific configuration that is used. Secondly, if we would want to measure performance, we would need a baseline measurement of performance. It is very hard to imagine what this would be. If we go for adaptable levels of automation, the robot operator can decide when the robot takes over tasks. This is a task in itself and could cause extra workload. If we go for static levels of automation, it is not clear which levels to use. Future research is needed to tackle these issues and set up a experiment that tests whether the model improves team performance.

¹Or possibly another CTL model that is used.

7 Conclusion

The main research question of this thesis is:

How can we use dynamic allocation of tasks to improve team performance?

In this section we will firstly detail conclusions and contributions to the current state of affairs for each of the three subquestions we formulated in the introduction (Section 1). After this, the conclusion for the main research question will be given.

Subquestions

1. Can we build a general framework describing the important concepts that influence team performance, that can be used to dynamically allocate tasks?

A high-level framework for dynamic task allocation, aimed at improving team performance in mixed human-robot teams was presented. The framework describes the important concepts that influence team performance and can be used to dynamically allocate tasks. The framework applies to a wide array of problems, including heterogeneous teams that might (but do not necessarily) include multiple human actors and multiple (sophisticated) robots or agents, a variety of tasks that might change over time and complex and dynamic environments. The framework provides a formally described base for designing applied models that dynamically allocate tasks. Using the framework makes models easier to design, extend and generalize.

2. Can we apply the framework to design a model for adaptive automation, triggered by cognitive task load?

We were able to use the framework as a base for designing a model for adaptive automation triggered by cognitive task load. The framework was general and flexible enough to cover all aspects needed to formalize the model, mainly cognitive task load (as a preference factor) and adaptive automation (as dynamic task allocation). Furthermore, using the framework allows for easy extension of the model where needed. We concluded (using results from the experiment) that reallocating tasks purely on cognitive task load is not optimal for improving performance. Some factors are also important, such as capability (who does what best), preference (what does the user like doing best), and trust or perceived capability (does the user think the robot can do the tasks). Cognitive task load is a very important factor, but to optimize performance, we need to take into account more factors. As the model is based on the framework, the model can be quite easily extended to include other factors, both static and dynamic.

The model addresses a wider range of problems than most current adaptive automation research as it focuses on multiple tasks each with their own variable level of autonomy. Also, the model is able to deal with teams that are larger than just a single robot or system and its operator.

3. Can we validate the effects of our model in an experimental setting?

As shown, we were able to design an experiment using the model, to measure the effects of the resulting adaptive automation. The model was instantiated for a single human agent cooperating with a single robot in the urban search and rescue domain. A small experiment was conducted aimed at testing the model. The experiment did not result in conclusive evidence that the model worked as it should, but encouraging results were found. Two of the three cognitive task load metrics (both the level of information processing and the mental occupancy) of participants could be managed using the model. Furthermore, important focus points for improving the model and furthering research on adaptive automation in general were identified.

Main Research Question This research was aimed at finding out how to use dynamic task allocation to improve team performance in mixed human-robot teams. We have shown how to use dynamic task allocation to improve team performance. The most important contribution we made is providing a general high-level framework that can be used as a foundation for future research into dynamic task allocation, giving a guideline to designing dynamic task allocation models and making it easier to compare, draw inspiration from and extend studies that use the framework. We showed the framework's usability by building a model for adaptive automation and we showed its practical application by instantiating it for an experimental setting.

References

- [1] R. A. Guzzo, E. Salas, and I. L. Goldstein, *Team Effectiveness and Decision Making in Organizations*. Jossey-Bass San Francisco, 1995.
- [2] M. T. Brannick, E. Salas, and C. Prince, *Team Performance Assessment and Measurement: Theory, Methods, and Applications*, ser. Series in Applied Psychology. Lawrence Erlbaum Associates, 1997.
- [3] J. L. Burke, R. R. Murphy, E. Rogers, V. J. Lumelsky, and J. Scholtz, “Final report for the DARPA/NSF interdisciplinary study on human-robot interaction,” *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, no. 2, pp. 103–112, 2004.
- [4] K. S. Barber, A. Goel, and C. E. Martin, “Dynamic adaptive autonomy in multi-agent systems,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 2, pp. 129–147, 2000.
- [5] C. M. Barnes, J. R. Hollenbeck, D. T. Wagner, D. S. DeRue, J. D. Nahrgang, and K. M. Schwind, “Harmful help: The costs of backing-up behavior in teams,” *Journal of Applied Psychology*, vol. 93, no. 3, p. 529, 2008.
- [6] C. A. Miller and R. Parasuraman, “Beyond levels of automation: An architecture for more flexible human-automation collaboration,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 47, no. 1, pp. 182–186, 2003.
- [7] G. L. Calhoun, H. A. Ruff, S. Spriggs, and C. Murray, “Tailored performance-based adaptive levels of automation,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 56, no. 1, pp. 413–417, 2012.
- [8] M. A. Neerincx, “Cognitive task load design: Model, methods and examples,” in *Handbook of Cognitive Task Design*. Mahwah, NJ: Lawrence Erlbaum Associates, 2003, pp. 283–305.
- [9] M. A. Neerincx, S. Kennedie, M. Grootjen, and F. Grootjen, “Modeling the cognitive task load and performance of naval operators,” in *Foundations of Augmented Cognition. Neuroergonomics and Operational Neuroscience*, ser. Lecture Notes in Computer Science, D. Schmorrow, I. Estabrooke, and M. Grootjen, Eds. Springer Berlin / Heidelberg, 2009, vol. 5638, pp. 260–269.
- [10] S. Kennedie, “Performance prediction with cognitive task load and emotional state: Preliminary research for manned missions to mars,” Master’s thesis, Radboud University Nijmegen, 2009.
- [11] D. B. Kaber, E. Onal, and M. R. Endsley, “Design of automation for telerobots and the effect on performance, operator situation awareness, and subjective workload,” *Human Factors and Ergonomics in Manufacturing*, vol. 10, no. 4, pp. 409–430, 2000.

- [12] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, “Common metrics for human-robot interaction,” in *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*. ACM, 2006, pp. 33–40.
- [13] M. Wooldridge, *An Introduction to Multiagent Systems*. Wiley, 2008.
- [14] K. P. Sycara, “Multiagent systems,” *AI magazine*, vol. 19, no. 2, p. 79, 1998.
- [15] P. R. Cohen and H. J. Levesque, “Teamwork,” *Nous*, vol. 25, no. 4, pp. 487–512, 1991.
- [16] M. Tambe, “Towards flexible teamwork,” *Journal of Artificial Intelligence Research*, vol. 7, pp. 83–124, 1997.
- [17] J. E. McGrath, *Social Psychology: A Brief Introduction*. Holt, Rinehart and Winston, 1964.
- [18] J. R. Hackman, “The design of work teams,” *Handbook of Organizational Behavior*, vol. 315, p. 342, 1987.
- [19] D. R. Ilgen, J. R. Hollenbeck, M. Johnson, and D. Jundt, “Teams in organizations: From input-process-output models to IMO models,” *Annual Review of Psychology*, vol. 56, pp. 517–543, 2005.
- [20] S. W. Kozlowski, R. P. DeShon, A. M. Schmidt, B. A. Chambers, and K. R. Milner, “Developing adaptive teams: Training strategies, learning processes, and performance adaptability,” Michigan State University, Tech. Rep., 2001.
- [21] E. Salas, N. J. Cooke, and M. A. Rosen, “On teams, teamwork, and team performance: Discoveries and developments,” *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 50, no. 3, pp. 540–547, 2008.
- [22] G. Matthew, D. R. Davies, S. J. Westerman, and R. B. Stammers, *Human Performance: Cognition, Stress, and Individual Differences*. Psychology Press, 2000.
- [23] M. A. Neerincx and N. J. P. Besouw, “Cognitive task load: A function of time occupied, level of information processing and task-set switches,” *Engineering Psychology and Cognitive Ergonomics, Volume Six: Industrial Ergonomics, HCI, and Applied Cognitive Psychology*, vol. 6, pp. 247–254, 2001.
- [24] J. Rasmussen, *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, ser. North-Holland series in system science and engineering. North-Holland, 1986.
- [25] M. A. Neerincx, J. A. Veltman, M. Grootjen, and J. v. Veenendaal, “A model for cognitive task load prediction: Validation and application,” in *Proceedings of the 15th Triennial Congress of the International Ergonomics Association*. Seoul, Korea: IEA2003, 2003.

- [26] T. Inagaki, “Adaptive automation: Sharing and trading of control,” in *Handbook of Cognitive Task Design*. Mahwah, NJ: Lawrence Erlbaum Associates, 2003, pp. 147–169.
- [27] J. Duggan, J. Byrne, and G. J. Lyons, “A task allocation optimizer for software construction,” *Software, IEEE*, vol. 21, no. 3, pp. 76–82, 2004.
- [28] M. R. Barrick, G. L. Stewart, M. J. Neubert, and M. K. Mount, “Relating member ability and personality to work-team processes and team effectiveness,” *Journal of Applied Psychology*, vol. 83, no. 3, p. 377, 1998.
- [29] J.-W. Streefkerk, M. Esch-Bussemakers, and M. A. Neerincx, “Context-aware team task allocation to support mobile police surveillance,” in *Foundations of Augmented Cognition. Neuroergonomics and Operational Neuroscience*, ser. Lecture Notes in Computer Science, D. Schmorow, I. Estabrooke, and M. Grootjen, Eds. Springer Berlin Heidelberg, 2009, vol. 5638, pp. 88–97.
- [30] L. Feng, J.-W. Hu, Y.-B. Zha, and Q.-J. Yin, “Dynamic task decomposition and allocation in CGF,” in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 5, 2010, pp. 624–628.
- [31] A. Vafaeinezhad, A. Alesheikh, M. Hamrah, R. Nourjou, and R. Shad, “Using GIS to develop an efficient spatio-temporal task allocation algorithm to human groups in an entirely dynamic environment case study: Earthquake rescue teams,” in *Computational Science and Its Applications ICCSA*, ser. Lecture Notes in Computer Science, O. Gervasi, D. Taniar, B. Murgante, A. LaganÃ , Y. Mun, and M. Gavrilova, Eds. Springer Berlin Heidelberg, 2009, vol. 5592, pp. 66–78.
- [32] K. v. Dongen and P. P. v. Maanen, “Designing for dynamic task allocation,” in *Proceedings of the Seventh International NDM Conference*, 2005.
- [33] P. A. Hancock, “Environmental stressors,” in *Sustained Attention in Human Performance*. John Wiley & Sons Chichester, 1984, pp. 103–142.
- [34] T. R. Colin, T. Mioch, N. J. J. M. Smets, and M. A. Neerincx, “Estimating an operator’s cognitive state in real time: a user modeling approach,” in *Ro-Man 2012, 21th IEEE International Symposium on Robot and Human Interactive Communication*, 2012.
- [35] A. Tsalatsanis, A. Yalcin, and K. P. Valavanis, “Dynamic task allocation in cooperative robot teams,” *Robotica*, vol. 30, pp. 721–730, 2012.
- [36] P. M. Fitts, M. S. Viteles, N. L. Barr, D. R. Brimhall, G. Finch, E. Gardner, W. F. Grether, W. E. Kellum, and S. S. Stevens, “Human engineering for an effective air-navigation and traffic-control system,” Ohio State University Research Foundation Columbus, Tech. Rep., 1951.

- [37] T. B. Sheridan and W. L. Verplank, “Human and computer control of undersea tele-operators,” Massachusetts Institute of Technology Cambridge Man-Machine Systems Lab, Tech. Rep., 1978.
- [38] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, “A model for types and levels of human interaction with automation,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 30, no. 3, pp. 286–297, 2000.
- [39] L. J. Prinzel III, F. G. Freeman, M. W. Scerbo, P. J. Mikulka, and A. T. Pope, “Effects of a psychophysiological system for adaptive automation on performance, workload, and the event-related potential P300 component,” *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 45, no. 4, pp. 601–614, 2003.
- [40] C.-H. Ting, M. Mahfouf, A. Nassef, D. A. Linkens, G. Panoutsos, P. Nickel, A. C. Roberts, and G. Hockey, “Real-time adaptive automation system based on identification of operator functional state in simulated process control operations,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 2, pp. 251–262, 2010.
- [41] T. E. d. Greef, H. F. R. Arciszewski, and M. A. Neerincx, “Adaptive automation based on an object-oriented task model: Implementation and evaluation in a realistic C2 environment,” *Journal of Cognitive Engineering and Decision Making*, vol. 4, no. 2, pp. 152–182, 2010.
- [42] K. Cosenzo, J. Chen, L. Reinerman-Jones, M. Barnes, and D. Nicholson, “Adaptive automation effects on operator performance during a reconnaissance mission with an unmanned ground vehicle,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 54, no. 25, pp. 2135–2139, 2010.
- [43] D. B. Kaber and J. M. Riley, “Adaptive automation of a dynamic control task based on secondary task workload measurement,” *International Journal of Cognitive Ergonomics*, vol. 3, no. 3, pp. 169–187, 1999.
- [44] R. Parasuraman, M. Mouloua, R. Molloy, and B. Hilburn, “Adaptive function allocation reduces performance costs of static automation,” in *The Adaptive Function Allocation for Intelligent Cockpits (AFAIC) Program: Interim Research Guidelines for the Application of Adaptive Automation*, 1993, p. 37.
- [45] N. R. Bailey, M. W. Scerbo, F. G. Freeman, P. J. Mikulka, and L. A. Scott, “Comparison of a brain-based adaptive system and a manual adaptable system for invoking automation,” *Human Factors: The Journal of the Human Factors and Ergonomics Society*, vol. 48, no. 4, pp. 693–709, 2006.
- [46] R. Parasuraman, K. A. Cosenzo, and E. d. Visser, “Adaptive automation for human supervision of multiple uninhabited vehicles: Effects on change detection, situation

- awareness, and mental workload,” *Military Psychology*, vol. 21, no. 2, pp. 270–297, 2009.
- [47] H. Saqer, E. d. Visser, A. Emfield, T. Shaw, and R. Parasuraman, “Adaptive automation to improve human performance in supervision of multiple uninhabited aerial vehicles: Individual markers of performance,” in *Proceedings of the Human Factors and Ergonomics Society*, 2011, pp. 890–893.
- [48] B. Kidwell, G. L. Calhoun, H. A. Ruff, and R. Parasuraman, “Adaptable and adaptive automation for supervisory control of multiple autonomous vehicles,” in *Proceedings of the Human Factors and Ergonomics Society*, 2012, pp. 428–432.
- [49] B. Hilburn, R. Molloy, D. Wong, and R. Parasuraman, “Operator versus computer control of adaptive automation,” in *The Adaptive Function Allocation for Intelligent Cockpits (AFAIC) Program: Interim Research Guidelines for the Application of Adaptive Automation*. DTIC Document, 1993, pp. 31–36.
- [50] N. Moray, T. Inagaki, and M. Itoh, “Adaptive automation, trust, and self-confidence in fault management of time-critical tasks,” *Journal of Experimental Psychology: Applied*, vol. 6, no. 1, p. 44, 2000.
- [51] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.
- [52] O. Shehory and S. Kraus, “Methods for task allocation via agent coalition formation,” *Artificial Intelligence*, vol. 101, no. 1, pp. 165–200, 1998.
- [53] L. Reinerman-Jones, G. Taylor, K. Sprouse, D. Barber, and I. Hudson, “Adaptive automation as a task switching and task congruence challenge,” *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 55, no. 1, pp. 197–201, 2011.
- [54] M. W. Scerbo, “Theoretical perspectives on adaptive automation,” in *Automation and Human Performance: Theory and Applications*, R. Parasuraman and M. Mouloua, Eds. CRC Press, 1996, pp. 37–63.
- [55] G. J. M. Kruijff, M. Janíček, S. Keshavdas, B. Larochelle, H. Zender, N. J. J. M. Smets, T. Mioch, M. A. Neerinx *et al.*, “Experience in system design for human-robot teaming in urban search & rescue,” in *Proceedings of 8th International Conference on Field and Service Robotics*, ser. STAR. Springer Verlag, 2012.
- [56] J. v. Diggelen, M. Grootjen, E. M. Ubink, M. van Zomeren, and N. J. J. M. Smets, “Content-based design and implementation of ambient intelligence applications,” in *Ambient Intelligence-Software and Applications*. Springer, 2013, pp. 1–8.

- [57] Y. Shi, N. Ruiz, R. Taib, E. Choi, and F. Chen, “Galvanic skin response (GSR) as an index of cognitive load,” in *CHI '07 extended abstracts on Human factors in computing systems*, ser. CHI EA '07. ACM, 2007, pp. 2651–2656.

Appendix 1

Glossary

Teams

Team: A group of two or more actors that set out to achieve a common goal. To achieve this goal, several different tasks need to be done or roles need to be fulfilled.

Team performance: A measure of how well the common goal of a team is achieved.

Mixed human-robot team: A team that consists of both one or more human actors and one or more robotic actors.

Single human-single robot team: A team that consists of one human actor and one robotic actor.

Dummy actor: A placeholder (actor) for tasks that are not executed.

Allocating Tasks

Task allocation: An agreement that specifies for each task a team needs to do, which actor is responsible for the execution of that task.

Dynamic task allocation: A task allocation that can change during task execution.

Comparison allocation: A procedure that allocates tasks based on capabilities of actors.

Role assignment: A combination of a single actor with the set of all tasks allocated to that actor.

Actor-task pair: A combination of a single actor with a single task that is allocated to that actor. This does not imply the actor has no other tasks allocated to him.

Actor-task set pair: A role assignment: a combination of a single actor with the set of all tasks allocated to that actor.

Utility

Utility: An evaluation measure of a task allocation (or role assignment or actor-task pair). The utility describes the expected quality minus the costs of the execution of the tasks by the actors that are assigned to do them.

Restrictive factor/restrictive state concept: A concept that describes whether role assignments are possible or impossible. This could be, for example, a concept that restricts stationary robots to get assigned to patrolling tasks.

Preference factor/preference state concept: A concept that puts a preference ordering on role assignments. This could be, for example, a concept that describes allocating a patrolling task to a fast moving robot should be preferred over allocating it to a slow moving robot. The preference ordering produced can be used as a factor in deciding on a task allocation.

Coordination costs: The costs introduced by switching between different tasks or between different level of autonomy variants of a task. In the current study, coordination costs mostly focus on the second aspect. The first aspect is taken into account by the CTL metric task set switching.

Autonomy

Level of autonomy (LoA): The level of autonomy of an actor on a task is the degree of freedom or responsibility the actor has in executing that task. In this thesis, the level of autonomy usually refers to the robot's level of autonomy.

Adaptive automation: The process in which robots, which can be assigned different levels of autonomy on tasks, operate under *automated* dynamic task allocation in a mixed human-robot team.

Adaptable automation: The process in which human team members can invoke a change in a robot's level of autonomy (instead of this being automated).

Static automation: The process in which a robot's level of autonomy does not change during task execution.

Cognitive Task Load

Cognitive task load (CTL): A measure of how task characteristics are of influence on individual performance and mental effort of a human actor. CTL consists of three metrics: level of information processing, time occupied (or mental occupancy) and task set switching.

Level of information processing (LIP): A measure of how mentally challenging tasks are for an actor, based on the skill-rule-knowledge framework described by Rasmussen [24].

Time occupied (TO): The percentage of time an actor is occupied by his tasks.

Mental occupancy (MO): The percentage of mental resources that are occupied by an actor's tasks.

Task set switching (TSS): The costs associated with switching between different tasks.

Shift in CTL (or one of the metrics): The change in CTL that occurs in result to a task reallocation.

Problem region: A CTL level that has a negative influence on performance. For example overload (all three metrics too high) or underload (all three metrics too low).

Problem state: The state an actor is in when his CTL level is in a problem region.

Modifier: A state of the environment that influences the contribution of some task to CTL. For example, driving through a cluttered environment is a more difficult task than driving through an environment without obstacles.

Temporal behavior of task: When a task is allocated to an actor, this does not imply he is constantly executing it. The temporal behavior of a task describes at what intervals an actor is actively executing a task, while the task is allocated to him.

Urban Search and Rescue

Urban Search and Rescue (USAR): A highly trained team of rescuers that specializes in finding and extracting victims from disaster sites. These sites can for example be large traffic accidents, earthquake sites, or buildings that are on fire.

Appendix 2

Casting the Task Allocation Problem to an Instance of the Set Partitioning Problem

We aim to cast the task allocation problem for multiple-task actor, single-actor task and instantaneous assignment (MT-SA-IA task allocation problem) to an instance of the Set Partitioning Problem (SPP). To do this, we firstly have to define both problems.

The MT-SA-IA task allocation problem can be defined in the following way:

Definition 2. (MT-SA-IA task allocation problem) Given the following input:

- A finite set of actors $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$
- A finite set of tasks $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ (which is disjoint from the set of actors $\mathcal{A} \cap \mathcal{T} = \emptyset$)
- A set of possible role assignments, which are pairs of an actor and a set of tasks that this actor could execute $RA = \{\langle a^1, \check{\mathcal{T}}^1 \rangle, \langle a^2, \check{\mathcal{T}}^2 \rangle, \dots, \langle a^m, \check{\mathcal{T}}^m \rangle\}$ where $\forall_{i=1}^m : a^i \in \mathcal{A} \wedge \check{\mathcal{T}}^i \subseteq \mathcal{T}$
- A utility function from role assignments to a utility value $f : RA \rightarrow \mathbb{R}_+$

Find a task allocation $TA \subseteq RA$ that includes exactly one role assignment for each actor $\forall A_i \in \mathcal{A} : \exists! \langle x, y \rangle \in RA : A_i = x$, allocates each task to exactly one actor $\forall T_i \in \mathcal{T} : \exists! \langle x, y \rangle \in RA : T_i \in y$ and has the maximum possible utility $\neg \exists TA' \subseteq RA : \sum_{ra \in TA'} f(ra) > \sum_{ra \in TA} f(ra)$.

The SPP is defined below:

Definition 3. (Set Partitioning Problem (SPP)) Given a finite set E , a family F of acceptable subsets of E , and a utility function $u : F \rightarrow \mathbb{R}_+$, find a maximum-utility family X of elements in F such that X is a partition of E . X is a partition of E if and only if the elements of X are mutually disjoint ($\forall y, z \in X, y \neq z : y \cap z = \emptyset$) and their union is E ($\bigcup_{x \in X} x = E$).

We can cast the input of the MT-SA-IA task allocation problem to the input for the SPP in the following way:

We define E to be the union of the set of actors and the set of tasks $E = \mathcal{A} \cup \mathcal{T}$. We define F to be the set of all feasible actor-task set pairs RA , where we change the notation of a role assignment from a pair to a set which is the union of both pair elements (actor and tasks): $\forall x, y : \langle x, y \rangle \in RA \leftrightarrow \{x\} \cup y \in F$. Note that, as required, the set F now only contains (certain) subsets of the set E . Furthermore, every set contained in F includes precisely a single actor (as follows from the definition of RA combined with

the fact that $\mathcal{A} \cap \mathcal{T} = \emptyset$). Lastly, we define u to be a the utility estimate function f for each role assignment. Since u takes sets as input, and f pairs, we define u to be $u(X) = \sum_{x \in X} \bigcap_{\mathcal{A}} f(\langle x, X \setminus x \rangle)$.

Solving the SSP for the above input, yields a maximum-utility family X of elements in F such that X is a partition of E . We aim to prove that X corresponds to a task allocation TA in the way in it defined above as desired result of the MT-SA-IA task allocation problem. To do this, we first construct X_{TA} from X and then prove X_{TA} is a task allocation TA .

Constructing X_{TA} from X :

We define X_{TA} as follows: $\forall x \in \mathcal{A} : \{x\} \cup y \in X \wedge x \notin y \leftrightarrow \langle x, y \rangle \in X_{TA}$. From the definition of the SPP, it follows that every element in X is also an element of F ($X \subseteq F$). Every element of F contains precisely one actor, as mentioned above. Thus, every element of X also contains precisely one actor. From this we can conclude that $\forall x \in \mathcal{A} : \{x\} \cup y \in X$ it holds that y does not contains actors. Since every element of X is also an element of F and every element of F is a subset of E , every element of X is also a subset of E . As E contains only actors and tasks (and these are two disjoint sets) y can only contain tasks, thus $y \subseteq \mathcal{T}$. The pairs in X_{TA} thus correspond to role assignments, pairs of an actor and a set of tasks that this actor could execute.

Proof that X_{TA} is a task allocation TA :

1. Since X is a partition of E , it holds that $\bigcup_{x \in X} x = E$. As $E = \mathcal{A} \cup \mathcal{T}$, the set of actors is a subset of E ($\mathcal{A} \subseteq E$). Taking these two facts together yields that $\bigcup_{x \in X} x \supseteq \mathcal{A}$, meaning for each actor, X contains at least one element that contains this actor. From the definition of X_{TA} it follows that if X contains an element that contains an actor, X_{TA} has an element that is a pair which first element is that actor. Thus, for each actor, X_{TA} has an element that is a pair which first element is that actor. Since pairs in X_{TA} correspond to role assignments, we conclude that X_{TA} includes a role assignment for each actor.
2. Since X is a partition of E , it holds that all its elements are mutually disjoint ($\forall y, z \in X, y \neq z : y \cap z = \emptyset$). Since every two elements of X are mutually disjoint, no two elements of X include the same actor. Following from the definition of X_{TA} , if no two elements of X contain the same actor, no two pairs in X_{TA} will contain the same actor as first element. Since pairs correspond to role assignments, X_{TA} does not include multiple role assignments for one actor.
3. Since some role assignment in X_{TA} exists for all actors (1) and no actor appears in multiple role assignments (2) we can conclude that each actor has exactly one role assignment in X_{TA} .
4. Since X is a partition of E , it holds that $\bigcup_{x \in X} x = E$. As $E = \mathcal{A} \cup \mathcal{T}$, the set of tasks is a subset of E ($\mathcal{T} \subseteq E$). Taking these two facts together yields that $\bigcup_{x \in X} x \supseteq \mathcal{T}$, meaning for each task, X contains at least one element that contains this task. From the definition of X_{TA} it follows that if X contains an element that

contains a task (and this element also contains an actor, which they all do), X_{TA} has an element that is a pair whose second element contains that task. Thus, for each task, X_{TA} has an element that is a pair whose second element contains that task. Since pairs in X_{TA} correspond to role assignments, we conclude that each task is included in some role assignment in X_{TA} .

5. Since X is a partition of E , it holds that all its elements are mutually disjoint ($\forall y, z \in X, y \neq z : y \cap z = \emptyset$). Since every two elements of X are mutually disjoint, no two elements of X include the same task. Following from the definition of X_{TA} , if no two elements of X contain the same task, no two pairs in X_{TA} exist in which some task occurs in both pair's second element, since each element of X only includes one actor. Since pairs correspond to role assignments, no task appears in multiple role assignments in X_{TA} .
6. Since all tasks are included in some role assignment in X_{TA} (4) and no task appears in multiple role assignments (5) we can conclude that each task is allocated to exactly one actor in X_{TA} .
7. According to the definition of the SPP, X_{TA} represents the maximum utility (function u) family of F that is a partition of E . This means there is no family $X_{TA'}$ of F that is a partition of E , that has a utility higher than X_{TA} ($\neg \exists X_{TA'} : X_{TA'} \subseteq F : \sum_{ra \in X_{TA'}} u(ra) > \sum_{ra \in X_{TA}} u(ra)$). Since the function u is defined as $u(X) = \sum_{x \in X \cap \mathcal{A}} f(\langle x, X \setminus x \rangle)$, and every element of F (and thus of X_{TA} and $X_{TA'}$) always contains precisely one actor, the function u over a set X always has the same outcome as the function f over the actor task-set pair that corresponds to this set ($\langle x, X \setminus x \rangle$ where $x \in \mathcal{A}$). Thus, X_{TA} has the maximum possible utility according to function f : ($\neg \exists X_{TA'} : (\forall x \in \mathcal{A} : \{x\} \cup y \in TA' \leftrightarrow \langle x, y \rangle \in X_{TA'}) \wedge TA' \subseteq RA : \sum_{ra \in TA'} f(ra) > \sum_{ra \in TA} f(ra)$).
8. Since X_{TA} includes exactly one role assignment for each actor (3), each task is allocated to exactly one actor (6) and it has the maximum possible utility (7), X_{TA} is a task allocation TA . \square

We have now successfully cast the MT-SA-IA task allocation problem to an instance of the SPP.

Appendix 3

Results of Domain Independent Pruning of Possible Role Assignments

As before said (3.3.5), we need to generate all possible role assignments to use as input for applying a SPP solving algorithm. If this set is very large, solving the SPP will be very time-consuming.¹ If our model would not include pruning rules, the set of possible role assignments would be enormous. It would include all distinct pairs of an actor (there are $|\mathcal{A}|$ actors) and a subset of tasks (there are $|\mathcal{T}|$ tasks, so $2^{|\mathcal{T}|}$ subsets of tasks). The number of role assignments without pruning is $(2^{|\mathcal{T}|})^{|\mathcal{A}|}$. Even for a small set of actors and a small set of tasks this number can be very large, making it clear that we should have effective pruning.

We can use the domain independent pruning concepts we defined in Section 4.2.2, namely DummyExecute, ActorMatch and CompatibleTasks. For now, let us assume the most unrestrictive form of CompatibleTasks: all tasks are compatible² except two tasks that are the same, but at different levels of autonomy. We have defined the set of different level of autonomy variants for each task T_x as: $T_x = \{T_x^1, T_x^{2,h}, T_x^{2,r}, \dots, T_x^{k-1,h}, T_x^{k-1,r}, T_x^k\}$, where 1 to k is the level of autonomy and h and r specify whether the task is appropriate for a human respectively a robot to execute. This definition states that the lowest and highest lowest and highest level of autonomy only have a single task (for the human respectively the robot) and all tasks in between have two variants (one for the human and one for the robot). The number of possible levels of autonomy of a task T_x can thus be calculated with the following formula:

$$LoA(T_x) = \begin{cases} |T_x| & \text{if } |T_x| < 3 \\ 2 + \binom{|T_x| - 2}{2} & \text{otherwise} \end{cases}$$

The pruning concept DummyExecute enforces that all tasks are allocated to the dummy actor at either all level of autonomy variants (task is not executed at all) or all but one level of autonomy variants (task is executed at one level of autonomy). This implies that for all tasks T_x , the number of possible subsets of T_x that can be allocated to the dummy actor is precisely the amount of different autonomy levels T_x has (all but one level of autonomy variants are assigned to dummy actor) plus one (all level of autonomy variants are assigned to dummy actor): $LoA(T_x) + 1$. This makes the number of possible role assignments for the dummy actor $\prod_{T_x \in \mathcal{T}'(t)} (LoA(T_x) + 1)$. The pruning concept CompatibleTasks enforces that all other actors besides the dummy actor can only have a tasks allocated to them at maximum one level of autonomy. The concept ActorMatch further enforces that a human actor can only have human tasks (the lowest LoA up to the second highest LoA) and a robotic actor can only have robot tasks (the second lowest LoA up to the highest LoA). This implies that for all tasks T_x , the number

¹Which is a problem as we want our model to work real-time, dynamically allocating tasks in response to current context.

²We have define compatible to mean two tasks can be allocated to an actor at the same time, thus occur together in a role assignment.

of possible subsets of T_x that can be allocated to the all actors besides the dummy actor is precisely the amount of different autonomy levels T_x has minus one (either highest of lowest level of autonomy) plus one (no level of autonomy at all) = $LoA(T_x)$. Thus, the number of possible role assignments for all non-dummy actors is $\prod_{T_x \in \mathcal{T}'(t)} LoA(T_x)$. The total number of possible role assignments after using only domain independent pruning rules is:

$$(\prod_{T_x \in \mathcal{T}'(t)} (LoA(T_x) + 1)) + (\prod_{T_x \in \mathcal{T}'(t)} LoA(T_x)) * |\mathcal{A}'(t)|$$

The above formula has a substantially smaller output than the formula we had before pruning $((2^{|\mathcal{T}'|})^{|\mathcal{A}'|})$, especially for large actor/task sets. Thus, we have effective domain independent pruning.

Appendix 4

Task description for participants

Taakomschrijving

Scenario

Er heeft een aardbeving plaatsgevonden waardoor een kantoorgebouw is ingestort. Je bent een reddingswerker die in het gebouw moet gaan kijken wat de situatie is. Door verder instortingsgevaar van het gebouw mogen er geen mensen naar binnen en wordt er gebruik gemaakt van robots. Mogelijk zijn er slachtoffers in het gebouw. Je hebt beschikking over een tactische display waarin deze slachtoffers moeten worden vastgelegd zodat andere hulpdiensten daar rekening mee kunnen houden. Denk bijvoorbeeld aan voldoende mankracht en vervoer kunnen regelen om de levende slachtoffers zo goed mogelijk te kunnen helpen. Ook is het belangrijk om grote obstakels, die lastig zouden kunnen zijn bij de zoektocht toe te voegen aan de tactische display. Een collega-reddingswerker is met iemand aan het praten die in het gebouw werkt, maar vandaag niet aanwezig was. Deze persoon vertelt soms waar hij denkt dat mensen zich bevinden. Je collega voegt dit dan toe aan de tactische display, eventueel kun je hier naar toe gaan om te kijken. Er wordt 15 minuten gezocht, daarna zullen de slachtoffers die niet gevonden zijn niet meer gered kunnen worden.

Geef nu een seintje aan de proefleider, deze zal het trainingslevel opstarten.

De omgeving

Je werkt met twee schermen, zie een voorbeeld op het plaatje hieronder. Op het linkerscherm zie je de beelden die worden opgenomen door de camera die op de robot zit. Op het rechterscherm zie je de tactische display.



- Vul het formulier zo compleet mogelijk in als daar sprake van is.

- Voeg eventueel commentaar (opmerking) toe.
- Klik "Add" om door te voeren (of 'cancel' om te annuleren).

Je kunt op bestaande datapunten klikken om meer informatie te tonen. Je kunt deze informatie ook aanpassen. Je kunt een datapunt niet verplaatsen, als je dit toch wilt doen zul je het datapunt moeten weghalen en opnieuw aanmaken. Een datapunt verwijderen kan door op "remove" te klikken.


Taak

Je gaat samen met de robot als team op zoek in het gebouw. Je hebt beschikking over een digitale kaart (tactische display) waar op aangegeven kan worden wat er in het kantoor ligt; bijvoorbeeld slachtoffers en obstakels. Het is belangrijk om hierop zo goed mogelijk in te vullen hoe de situatie is zodat je collega-reddingswerkers op een later moment snel en makkelijk de slachtoffers kunnen weghalen. Verplaats je dus in je collega's, zij horen bij jouw team en jullie willen zo veel mogelijk mensen redden. Het belangrijkste is dat duidelijk gemaakt wordt waar de slachtoffers liggen (zowel levende als overleden slachtoffers) en waar grote obstakels zijn.

Er zijn vier subtaken die allemaal tegelijk uitgevoerd moeten worden. Ten eerste navigatie, dit is het rondrijden van de robot. De tweede subtaak is het herkennen en ontwijken van obstakels. De derde subtaak is het herkennen van slachtoffers en deze toevoegen aan de tactische display. De vierde subtaak is het verwerken van de informatie die je collega-reddingswerker toevoegt aan de tactische display. Deze vier taken hebben allemaal drie verschillende varianten, waarin het verschillend is hoeveel van de taak jij zelf op je neemt en hoeveel de robot zelfstandig doet.

De taakverdeling

Er zijn dus vier taken (navigatie, obstakels, slachtoffers, informatie verwerken) die altijd uitgevoerd moeten worden. Van alle vier de taken bestaan drie varianten, er is altijd één variant actief tijdens de missie. Welke variant dit is kunt je zien linksboven in je linker scherm (de gele tekst, zie het plaatje hiernaast). Hier staan onder elkaar de vier actieve varianten van de vier taken, dit is de taakverdeling. Welke variant van een taak actief is, kan veranderen tijdens de missie (verandering van de taakverdeling). Het is dus heel belangrijk dat je er constant op let welke variant van de taken je moet uitvoeren! (Met andere woorden: in welke mate je iedere taak moet uitvoeren.)



Tele-operation
No obstacle detection
No victim detection
No automatic information processing

De taakverdeling wordt in de gaten gehouden door een recentelijk ontwikkeld systeem. Dit systeem reageert op de taaklast die jij ervaart tijdens de missie. Het systeem verandert de taakverdeling zodra het denkt dat jouw taaklast niet optimaal is. Het doel van dit experiment is testen of het systeem goed werkt. Het is dus zeer belangrijk dat je op de taakverdeling let en altijd de variant van de taken uitvoert die op je scherm wordt weergegeven.

Taak – Navigatie

De navigatietaak is het rondrijden van de robot door de omgeving. Hiernaast zie je een plaatje van de robot.

De robot is langzaam en heeft een vaste camera bovenop. De robot heeft moeite met het nemen van obstakels en door de bumpers voor en achter kan de robot ergens achter blijven haken. De robot is makkelijk te besturen maar kan zich vastrijden in een gat of tussen obstakels. Je bestuurt de robot met de linkerjoystick van de gamecontroller. Door de joystick minder ver te bewegen zal de robot minder snel rijden of sturen, om de robot nauwkeurig te besturen.



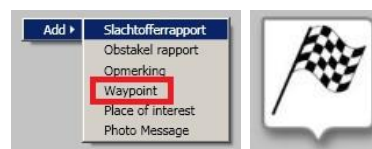
Van de navigatietaak bestaan drie verschillende varianten, deze varianten worden hieronder kort uitgelegd.

1. Tele-operation

Jij bestuurt de robot.

2. Way-point driving

Jij voegt een waypoint (zie hiernaast) toe aan de tactische display op de plek waar je wilt dat de robot naartoe gaat. De robot zal dan zelf naar dit punt toe navigeren. Voeg maar één waypoint tegelijk toe, de robot zal het waypoint verwijderen als hij is aangekomen. (Je kunt een 'place of interest'-icoon plaatsen om zelf te onthouden waar je eventueel later de robot nog naar toe wilt laten gaan.) Zet de waypoints niet al te ver weg (niet helemaal aan de andere kant van de kaart, in een andere kamer is prima en aan te raden). Terwijl de robot naar een waypoint toe rijdt, kun jij je bezig houden met andere dingen zoals informatie verwerken die binnenkomt of slachtofferformulieren invullen.



3. Full autonomy

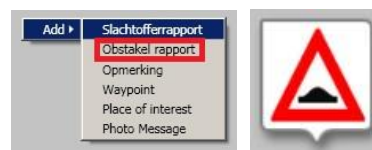
De robot bestuurt zichzelf. Je hoeft geen waypoints te zetten, de robot bepaalt zelf waar hij heen gaat.

Taak – Obstakels

De tweede taak is het herkennen en ontwijken van obstakels. Er zijn weer drie varianten, deze worden hieronder uitgelegd.

1. No obstacle detection

De robot ontwijkt zelf geen obstakels, dus jij zult dit moeten doen. Mocht je grote obstakels zien, voeg deze dan toe aan de tactische display (zie hiernaast).



2. Robots warns for obstacles

Deze taak is door technische problemen niet mogelijk en zal dus niet voorkomen.

3. Robot avoids obstacles

De robot ontwijkt zelf obstakels, jij hoeft dit niet te doen. Jij hoeft geen obstakels toe te voegen aan de tactische display. Let op: het zou kunnen voorkomen dat deze taak voorkomt terwijl jij wel zelf de robot bestuurt (tele-operation). In dit geval kan de robot dus de controle van je overnemen, als hij denkt dat je anders tegen een obstakel op gaat botsen.

Taak – Slachtoffers

De derde taak is het herkennen van slachtoffers en deze toevoegen aan de tactische display. Bij een slachtoffer is altijd een nummer te vinden, zie het plaatje hiernaast. Met behulp van dit nummer kun je in de papieren lijst die je hebt gekregen opzoeken wat er met het slachtoffer aan de hand is. Vul deze informatie in op het slachtofferformulier in de tactische display.



Van deze taak bestaan drie verschillende varianten, deze varianten worden hieronder kort uitgelegd.

1. No victim detection

De robot herkent geen slachtoffers. Jij zult de slachtoffers moeten herkennen en toevoegen aan de tactische display (zie plaatje hiernaast). Vul alle informatie in!



2. Robot proposes possible victims

De robot zal foto's maken van slachtoffers die hij denkt te herkennen. Deze foto's verschijnen op je tactische display op de plek waar ze genomen zijn. Als je op de foto inderdaad een slachtoffer ziet, voeg deze dan toe aan de tactische display en vul alle informatie in. De robot voegt zelf geen slachtoffers toe aan de tactische display, het is dus belangrijk dat jij alle foto's bekijkt en de slachtoffers toevoegt.

3. Robot detects victims

De robot voegt slachtoffers die hij herkent zelf toe aan de tactische display. Jij hoeft geen slachtoffers toe te voegen.

Taak – Informatie verwerken

De laatste taak is het verwerken van de informatie die binnenkomt op je tactische display van je collega-reddingswerker. (Een collega-reddingswerker is met iemand aan het praten die in het gebouw werkt, maar vandaag niet aanwezig was. Deze persoon vertelt soms waar hij denkt dat mensen zich bevinden. Je collega voegt dit dan toe aan de tactische display.) De informatie komt binnen in de vorm van remarks, zie het plaatje hiernaast. Je kunt op een remark klikken om de inhoud te zien. Van de informatietaak bestaan drie verschillende varianten, deze varianten worden hieronder kort uitgelegd.



1. No automatic information processing

Jij leest de binnenkomende informatie en beslist waar de robot naar toe gaat.

2. Robots suggests possible places of interest

De robot verwerkt de binnenkomende informatie en maakt op basis hiervan suggesties van wat hij denkt dat mogelijk interessante locaties zijn om naar toe te gaan. Jij leest de binnenkomende informatie niet. De robot voegt de mogelijk interessante locaties toe aan de tactische display, zie het plaatje hiernaast. Jij kiest op basis van de opties die de robot geeft waar hij naartoe gaat.



3. Robot decides where to go

De robot verwerkt de binnenkomende informatie en beslist op basis hiervan zelf waar hij naar toe gaat. Jij leest de binnenkomende informatie niet.

De missies

Je gaat straks drie keer een missie doen. Deze missies duren elk 15 minuten. Mocht je nog vragen of onduidelijkheden hebben over de taken of iets anders dan kun je nu vragen stellen.

De proefleider zal de schermen opstarten die je nodig hebt voor elke missie, doe dit niet zelf! De proefleider vraagt hierna of je klaar bent om te beginnen. Als dit zo is, wacht je tot de proefleider start zegt. Op start druk je op de "start timer" knop die rechts in je tactische display gaat. De tijd begint nu te lopen en je kunt beginnen met je taak. Wanneer de vijftien minuten om zijn, staat de timer op nul en stop je met de taak. Je hoeft zelf je schermen niet af te sluiten, de proefleider zal dit voor je doen! Na elke missie is er een korte pauze.

Na de drie missies wordt je gevraagd een korte vragenlijst in te vullen. Hierna ben je klaar.

In het kort: Let goed op de taakverdeling en zorg dat jij en de robot zoveel mogelijk van de omgeving verkennen en zoveel mogelijk slachtoffers/grote obstakels toevoegen aan de tactische display. Vergeet niet de slachtofferformulieren goed in te vullen en op de hints te letten die verschijnen (wanneer dit jouw taak is). Je hebt niet veel tijd, dus probeer zo snel mogelijk te werken! Als jij samen met de robot de taak het best uitvoert van alle proefpersonen (in totaal 15) dan win je een leuk cadeautje.

Succes!

Geef de proefleider nu een seintje dat je klaar bent met lezen en stel eventuele vragen die je nog hebt.

Appendix 5

Pseudo Code Description of the Model

```
function MAIN( )
  SET UP ANNOTATING INTERFACE( )
  INITIALIZE MODEL( )
  while true do
    if a button is pressed then
      update current timeline
    end if
    if 5 seconds have passed since this if-loop was last entered then
      shorten current timeline to last 120 seconds
      current CTL ← GET CTL(current timeline)
      if current CTL != neutral then REALLOCATE TASKS( )
      end if
    end if
  end while
end function
```

```
function SET UP ANNOTATING INTERFACE( )
  sets up an interface in which an observer can annotate
  which tasks the UGV operator is doing by pressing a button
end function
```

```
function INITIALIZE MODEL( )
  initializes tasks with task properties
  initializes possible task allocations
end function
```

```
function GET CTL(timeline)
  uses Colin's model to calculate CTL from timeline
end function
```

```

function REALLOCATE TASKS( )
  best utility  $\leftarrow$  0
  for all options in possible task allocations do
    construct predicted timeline for option
    predicted CTL option  $\leftarrow$  GET CTL(constructed timeline for option)
    preference CTL option  $\leftarrow$ 
      GET PREFERENCE CTL(predicted CTL, option)
    preference CC option  $\leftarrow$ 
      GET PREFERENCE CC(option, current allocation)
    utility option  $\leftarrow$  (preference CTL option + preference CC option)/2
    if utility option > best utility then1
      best utility  $\leftarrow$  utility option
      best option  $\leftarrow$  option
    end if
  end for
  reallocate tasks with best option
  save option as current allocation
end function

```

```

function GET PREFERENCE CTL(predicted CTL, option)
  Calculate the preference based on CTL, as described in Section 4.3.2
end function

```

```

function GET PREFERENCE CC(option, current allocation)
  Calculate the preference based on coordination costs, as described in Section 4.4.2
end function

```

¹Note that if there are multiple options with the same highest utility, this function selects the first one it finds. Options were ordered from lowest level of autonomy to highest level of autonomy (firstly on the victim detection task, next on navigation, next on the obstacle task and lastly on the information task).

Appendix 6

Questionnaire - Before Experiment

Vragenlijst vooraf

Proefpersoonnummer: ____ (in te vullen door experimentleider)

Deze vragenlijst bevat een aantal vragen over persoonlijke karakteristieken die van invloed kunnen zijn op uw prestatie tijdens het experiment. De antwoorden worden alleen gebruikt om het resultaat te analyseren.

1. Wat is uw leeftijd?

2. Wat is uw geslacht?

3. Wat is uw hoogst gevolgde opleiding (bv. HBO of WO)?

4. Hoeveel uur per week bestuurt u gemiddeld een voertuig (bv. auto of motor)?

5. Hoeveel uur per week besteedt u gemiddeld aan het spelen van computerspellen (bv. op de PC of een console)?

Z.O.Z.

Bij de volgende vragen wordt u gevraagd hoeveel ervaring u heeft met verschillende systemen. Hierbij staat het antwoord '1' gelijk aan 'geen ervaring' en antwoord '5' gelijk aan 'heel veel ervaring'. Omcirkel het antwoord tussen de '1' en '5' dat voor uw gevoel het best klopt.

6. Hoeveel ervaring heeft u met het besturen van op afstand bestuurbare voertuigen/robots/speelgoed?

(geen ervaring) 1 2 3 4 5 (heel veel ervaring)

7. Hoeveel ervaring heeft u met autonome robots?

(geen ervaring) 1 2 3 4 5 (heel veel ervaring)

Appendix 7

Questionnaire - After Experiment

Vragenlijst achteraf

Proefpersoonnummer: ____ (in te vullen door experimentleider)

Deze vragenlijst gaat over uw persoonlijke ervaring met verschillende aspecten van het experiment. Een aantal beweringen zal u worden voorgelegd. U wordt gevraagd aan te geven in hoeverre u het met deze beweringen eens bent. Dit antwoord mag tussen de '1' en de '5' zitten. Hierbij staat het antwoord '1' gelijk aan '*helemaal mee oneens*' en antwoord '5' gelijk aan '*helemaal mee eens*'. Omcirkel het antwoord tussen de '1' en '5' dat voor uw gevoel het best klopt.

Omdat de beweringen over uw persoonlijke ervaring gaan, zijn er geen goede of slechte antwoorden. In het kader van het onderzoek is het belangrijk dat u de vragen goed leest en eerlijk antwoord geeft.

De volgende beweringen gaan over de timing van de veranderingen in taakverdeling.

1. Elke keer dat ik me te druk begon te voelen, veranderde de taakverdeling.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

2. De taakverdeling werd te vaak veranderd.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

3. Ik begreep waarom de taakverdeling veranderde, wanneer deze veranderde.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

4. De taak zou beter zijn gegaan (bv. meer slachtoffers gevonden en toegevoegd aan de tactische display of een beter overzicht van de situatie gekregen) als ik de momenten waarop de taakverdeling veranderde zelf had kunnen kiezen.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

5. Soms had ik het gevoel dat ik meer taken zou kunnen doen, maar werd de taakverdeling niet veranderd.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

6. Het moment waarop de taakverdeling veranderd werd, verraste me nooit.

(*helemaal mee oneens*) 1 2 3 4 5 (*helemaal mee eens*)

z.o.z.

De volgende beweringen gaan over de kwaliteit van de veranderingen in taakverdeling.

7. Elke keer dat ik me te druk begon te voelen nam de robot één of meerdere taken (deels) over.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

8. Als de taakverdeling constant zou zijn gebleven (geen veranderingen) dan zou de taak slechter zijn gegaan (bv. minder slachtoffers gevonden en toegevoegd aan de tactische display of een slechter overzicht van de situatie gekregen).

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

9. De veranderingen in taakverdeling hielpen om de taak sneller/beter uit te voeren.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

10. Soms moest ik één of meerder taken (deels) overnemen van de robot, terwijl ik mij niet het gevoel had dat ik meer taken zou kunnen doen.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

11. De taak was beter gegaan (bv. meer slachtoffers gevonden en toegevoegd aan de tactische display of een beter overzicht van de situatie gekregen) als ik zelf de taakverdeling had kunnen bepalen.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

12. Ik begreep altijd waarom een bepaalde taakverdeling werd gekozen.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

z.o.z.

Ten slotte volgen er nog wat algemene beweringen en vragen.

13. Ik voelde me soms te druk met mijn taken.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

14. Ik voelde me soms te rustig met mijn taken en had op die momenten het idee dat ik meer zou kunnen doen.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

15. Ik had het gevoel dat ik te weinig had kunnen oefenen om goed met de verschillende taakverdelingen om te kunnen gaan.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

16. Ik denk dat gebruik van dit taakverdelingsmodel een positieve invloed zou hebben op de uitvoering van de taak (omgeving verkennen, slachtoffers vinden) als ik meer had kunnen oefenen.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

17. Ik vertrouwde de robot en ik vertrouwde erop dat hij de taken kon uitvoeren die aan hem waren toebedeeld.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

18. Ik vond dat ik beter was in het uitvoeren van de taken dan de robot.

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

19. Ik geloofde dat de robot echt autonoom was (niet door een mens bestuurd werd).

(helemaal mee **oneens**) 1 2 3 4 5 (helemaal mee **eens**)

z.o.z.

20. Noem drie aspecten die u positief vond aan het taakverdelingsmodel / experiment.

21. Noem drie aspecten die u negatief vond aan het taakverdelingsmodel / experiment.

22. Heeft u verder nog opmerkingen over het taakverdelingsmodel / experiment?

Appendix 8

CTL Data Experiment Runs

In this section we give a detailed account of the CTL level and task reallocations that occurred during the experiment. When we look at all sessions together, we see that the CTL level of participants was mostly neutral (83% of the time).¹ The second most common cognitive state was cognitive lock-up (15%), followed by underload (2%). Overload was hardly experienced (< 1%) and vigilance was not experienced at all.

For clarity, the explanation of the tasks at their different level of autonomy (LoA) variant is repeated in Table 7. This table also includes the percentage of time a task was active at a specific LoA, over all sessions. All sessions started with all tasks at LoA 1.

	LoA 1	LoA 2	LoA 3
Nav	Tele-operation 85%	Way-point driving 15%	Full autonomy 0%
Obs	No obstacle detection/avoidance 50%	Robots warns operator for nearby obstacles 0%	Robot avoids obstacles 50%
Vic	No victim detection 95%	Robot suggests possible victims 1%	Robots detects victims and forwards information 4%
Info	No automatic information processing 94%	Robot suggests possible places of interest 6%	Robot decides where to go 0%

Table 7: An overview of the tasks at different levels of autonomy and the percentage of time (over all sessions) tasks were active at each specific level of autonomy.²

In the following sections, a detailed account of the CTL level and task reallocations that occurred is given separately for each session. For each participant, three graphs are shown that indicate the CTL level of the participant during the three levels played during the experiment. The different metrics are scaled to have overlapping neutral zones, shown by the green background. The red line indicates mental occupancy (MO), the blue line indicates level of information processing (LIP) and the green line indicates task set switching (TSS). The graphs contains a vertical blue line for each moment the CTL is in a problem region (and thus the task reallocation model is called) and a vertical (thick) purple line for when tasks are actually reallocated.³ All levels started with no autonomy for the robot on any of the tasks. The caption below each graph explains which task reallocations occur. Task reallocations less than two minutes before the end

¹As the task reallocation model (and the CTL model) are not active in the first two minutes of each session, only the remaining thirteen minutes are used to calculate these percentages.

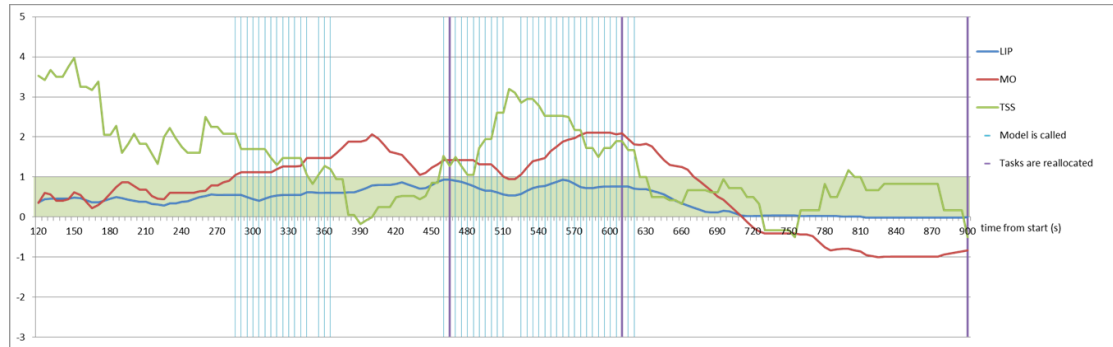
²See footnote 1.

³The vertical purple line at the end (right) of each graph is caused by closing the model and can be ignored.

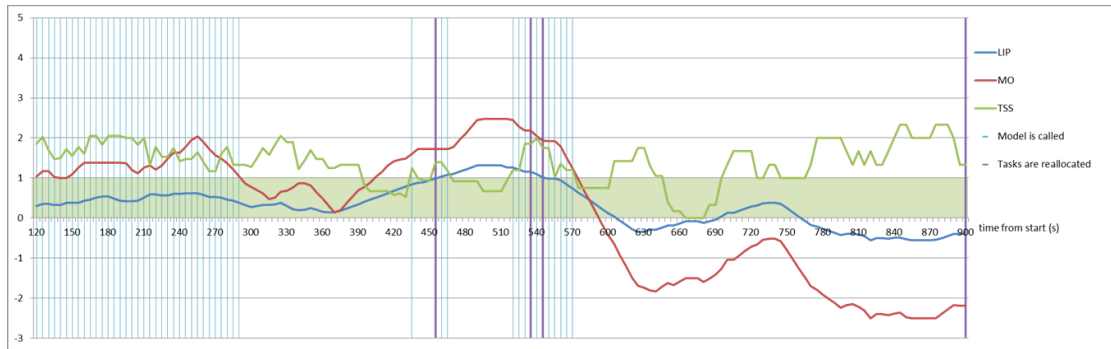
are not mentioned nor used for analysis as there is not enough future CTL data to analyze them with.

Participant 1 The CTL of participant 1 was mostly in a neutral zone (70%), sometimes in cognitive lock-up (28%), scarcely in overload (2%), scarcely in underload (< 1%) and never in vigilance. Tasks were reallocated a total of six times. Noteworthy for participant 1 is that the content of the task switches as well as the timing was quite similar in the three different levels. The only difference is the number of times tasks were reallocated.

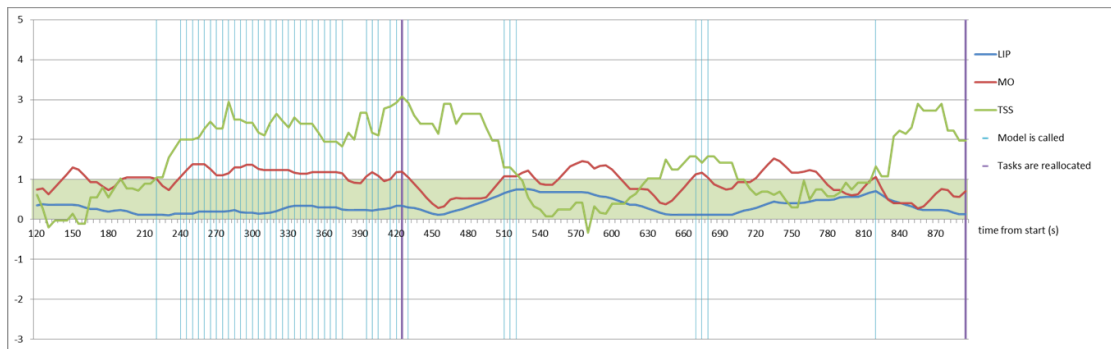
In the graph of level two we see a curious phenomenon. Two task reallocations occur very fast after each other. At the time of the first of these two reallocations, the model considers multiple different reallocations that could solve the cognitive task load problem. It prefers and chooses an option that only changes the level of autonomy on a single task, as this causes less coordination costs. Some seconds later, the cognitive task load did not yet have enough time to benefit from the new allocation and the model is again called. The time penalty included in the coordination costs should now prevent the model from reallocating tasks again (see Section 4.4). This does not happen, as the model only changes the level of autonomy on some other task and we defined the time penalty separately for each task. This problem can be solved by including a general time penalty that discourages a task reallocation to happen right after another task reallocation, no matter what specific tasks are reallocated.



Participant 1, level 1: The first reallocation switches the LoA of Info from 1 to 2. The second reallocation switches the LoA of Obs from 1 to 3.

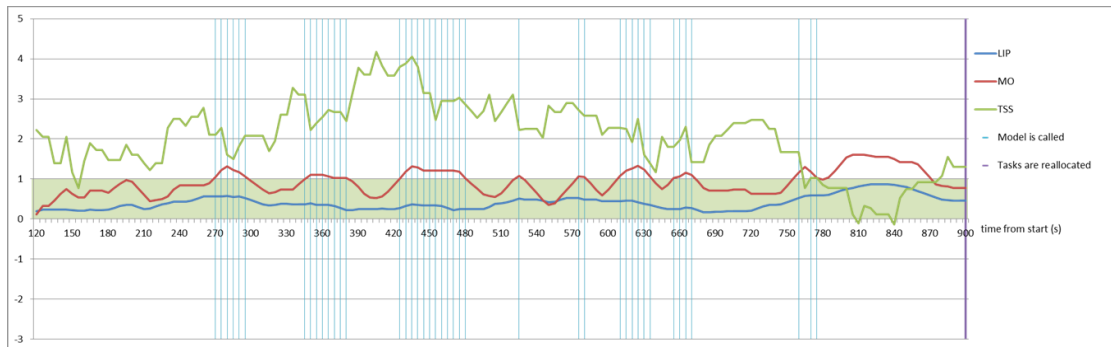


Participant 1, level 2: The first reallocation switches the LoA of Info from 1 to 2. The second reallocation switches the LoA of Obs from 1 to 3. The third reallocation switches the LoA of Nav from 1 to 2.

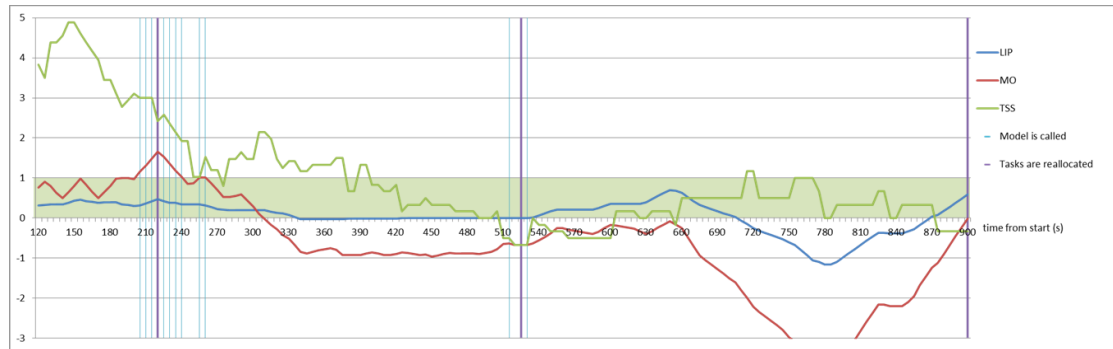


Participant 1, level 3: The first reallocation switches the LoA of Info from 1 to 2.

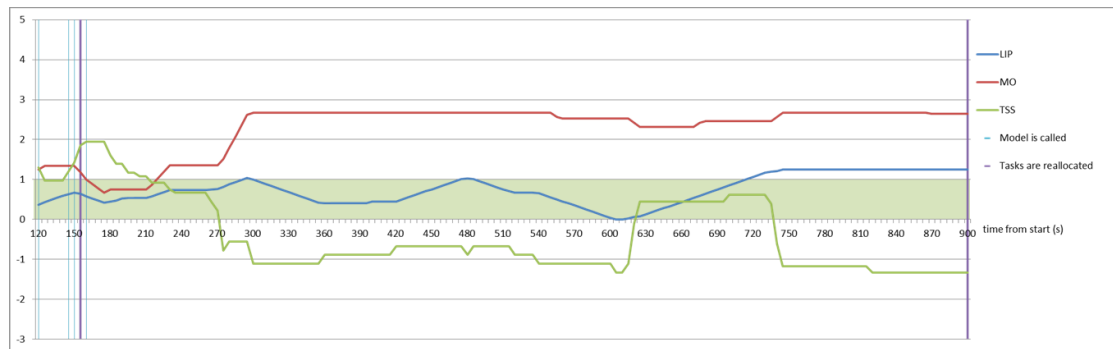
Participant 2 The CTL of participant 2 was mostly in a neutral zone (87%), sometimes in cognitive lock-up (12%), scarcely in underload (1%) and never in overload or vigilance. Tasks were reallocated a total of three times.



Participant 2, level 1: No reallocations occurred.



Participant 2, level 2: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

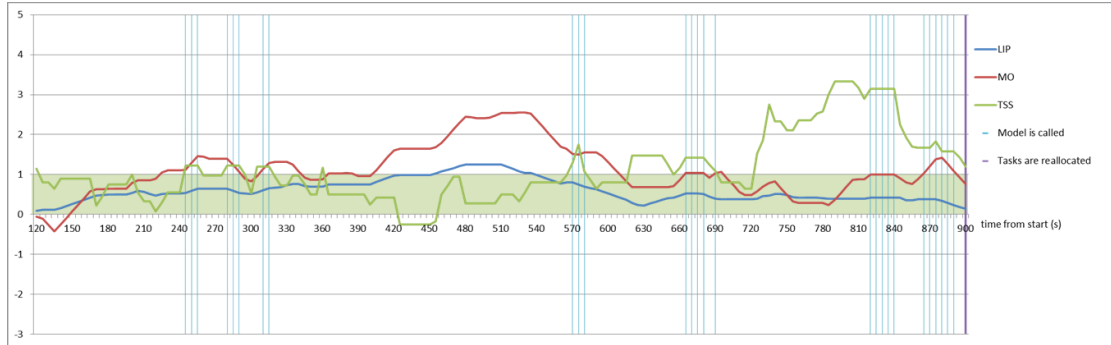


Participant 2, level 3: The first reallocation switches the LoA of Vic from 1 to 3.

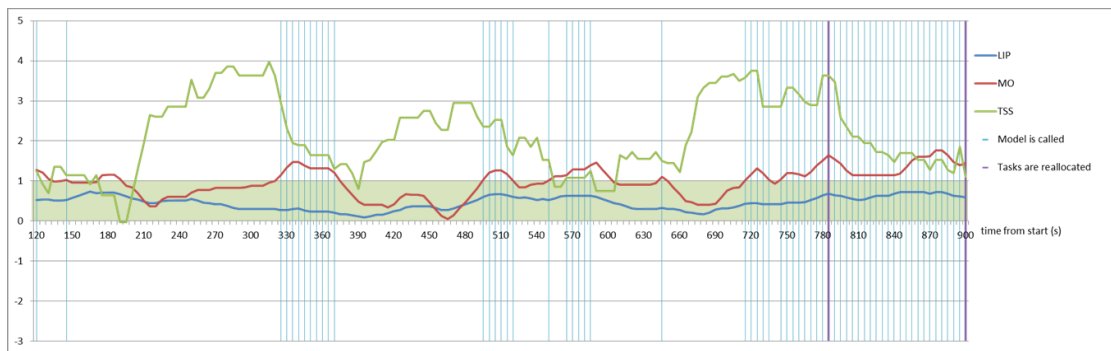
Participant 3 The CTL of participant 3 was mostly in a neutral zone (74%), sometimes in cognitive lock-up (21%), scarcely in underload (5%) and never in overload or vigilance. Tasks were reallocated a total of three times.

Noteworthy for participant 3 is that we can clearly see that the model is not able to resolve the underload situation in level three. Tasks are reallocated but the underload only gets worse. This is partly caused by a failure to take into account modifiers when predicting future task behavior. This is an error in the implementation of the model and can be easily fixed. However, a more important lesson can also be learned. The model chooses to increase the level of autonomy of the robot on a task to solve the underload problem, which is counterintuitive. The new task allocation is expected to resolve the underload situation as the higher level of autonomy will increase TSS just into its neutral zone. The new task allocation is also expected to decrease LIP and MO, but low LIP and MO (no matter how low) combined with neutral TSS does not constitute a problematic CTL. The preference based on CTL as it is described at the moment does not differentiate between multiple task reallocations that are all expected to result in neutral CTL. As coordination costs are lower for switching to a higher level of autonomy for the robot, this option is selected. We can solve this issue by making preference based on CTL more fine-grained. For example, solving underload by increasing all three

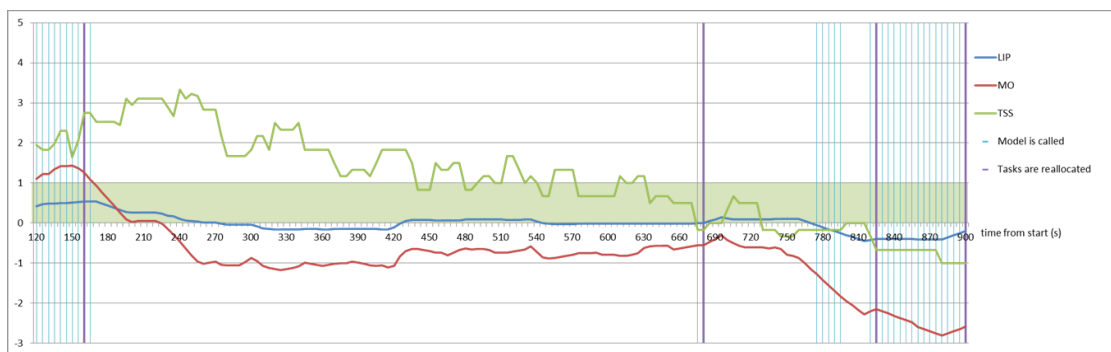
metrics into their neutral zone should be preferred over solving underload by increasing one metric to its neutral zone while decreasing the two other metrics. This makes it more likely that the underload is indeed resolved if the future temporal behavior of the tasks differs (a bit) from what was predicted, making the model more robust.



Participant 3, level 1: No reallocations occurred.

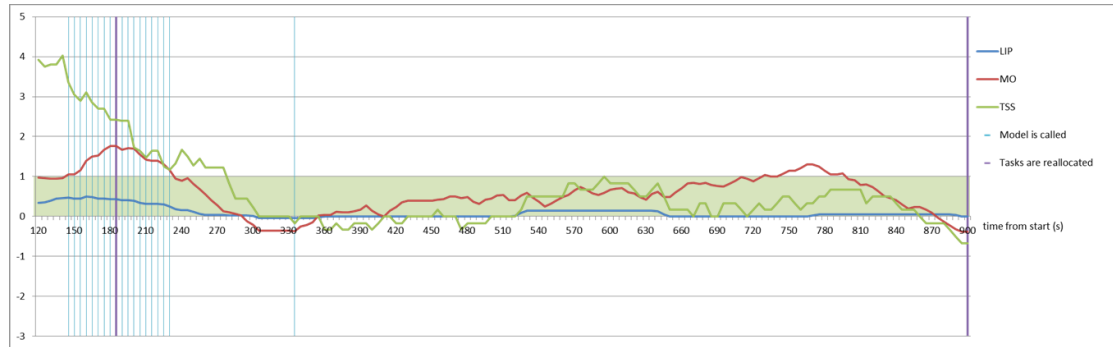


Participant 3, level 2: The first reallocation switches the LoA of Info from 1 to 2.

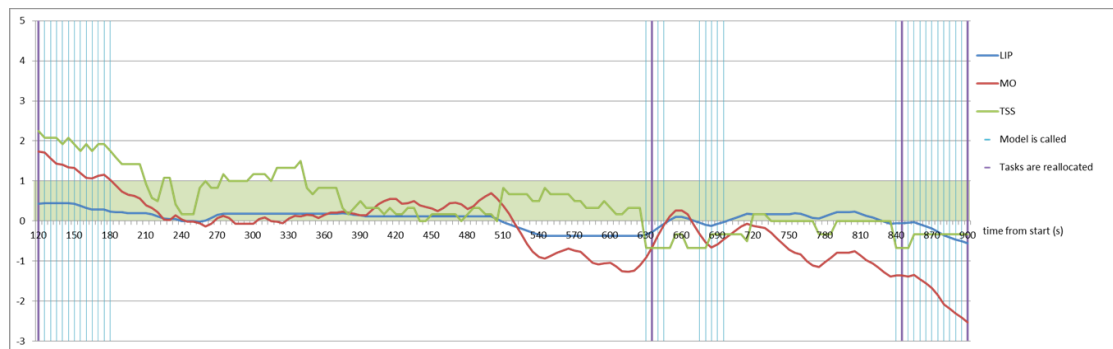


Participant 3, level 3: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Info from 1 to 2.

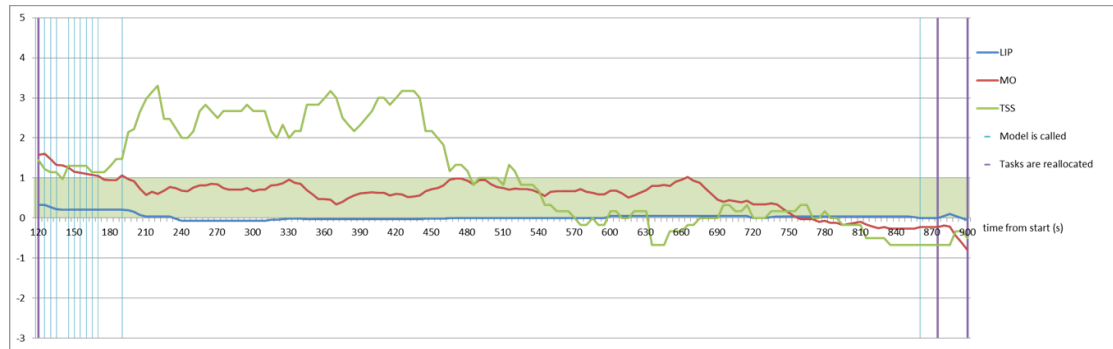
Participant 4 The CTL of participant 4 was mostly in a neutral zone (85%), sometimes in cognitive lock-up (9%), sometimes in underload (6%) and never in overload or vigilance. Tasks were reallocated a total of four times. The first reallocation is the same for all three levels and comparable in terms of timing. This reallocation solves the cognitive lock-up problem. The underload problem at the end of level two is again difficult for the model to solve.



Participant 4, level 1: The first reallocation switches the LoA of Obs from 1 to 3

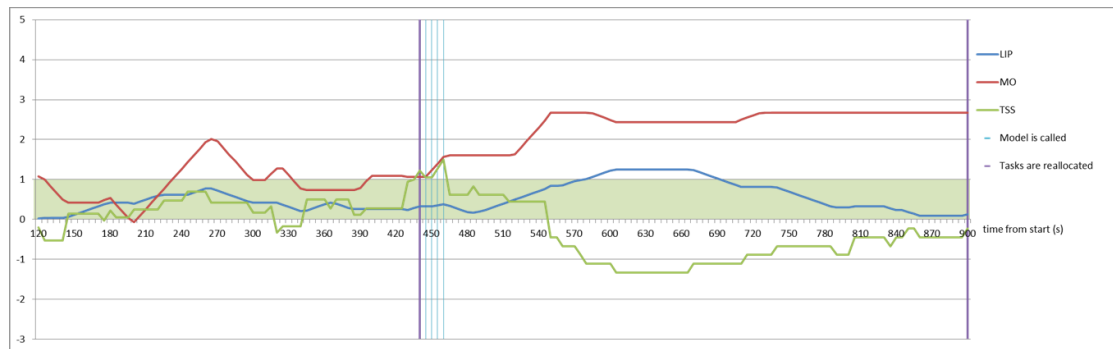


Participant 4, level 2: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.



Participant 4, level 3: The first reallocation switches the LoA of Obs from 1 to 3.

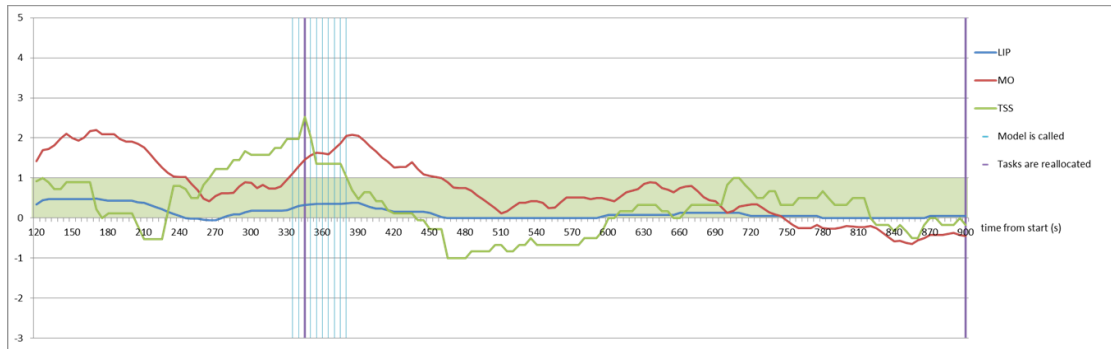
Participant 5 Participant 5 only completed one level due to motion sickness, during this level tasks were reallocated once. The CTL of participant 5 was mostly in a neutral zone (97%), scarcely in cognitive lock-up (3%), and never in overload, underload or vigilance.



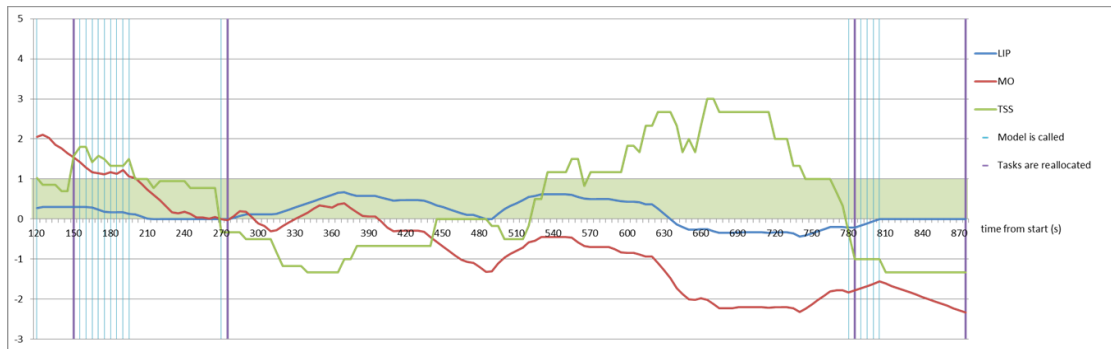
Participant 5, level 1: The first reallocation switches the LoA of Vic from 1 to 3

Participant 6 Participant 6 was not able to properly execute the task as his vision was impaired. His data was not used for analysis.

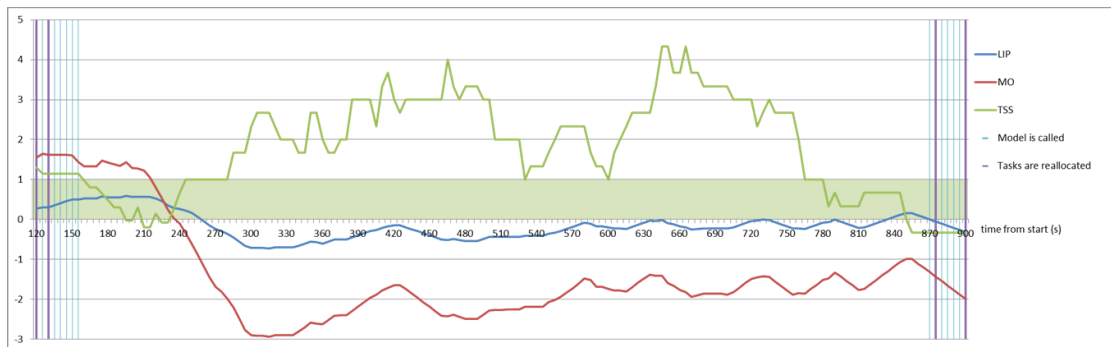
Participant 7 The CTL of participant 7 was mostly in a neutral zone (90%), sometimes in cognitive lock-up (6%), scarcely in underload (3%) and never in overload or vigilance. Tasks were reallocated a total of five times. The first reallocation is the same for all three levels and comparable in terms of timing (except for the first level where it occurred some minutes later). The second reallocation is also the same in the different levels except for the first level (which has only one reallocation). In level two we see the same curious phenomenon (two subsequent task reallocations) as we saw for participant 1 in level two, the same reasoning as explained there applies for this case.



Participant 7, level 1: The first reallocation switches the LoA of Obs from 1 to 3

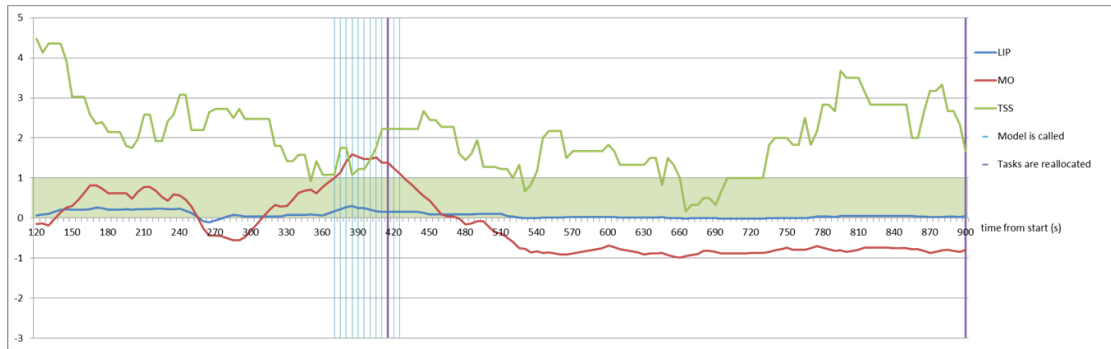


Participant 7, level 2: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

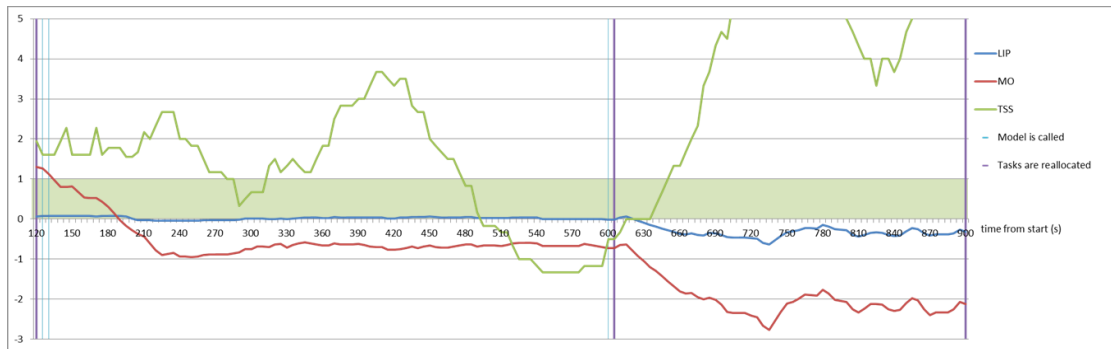


Participant 7, level 3: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

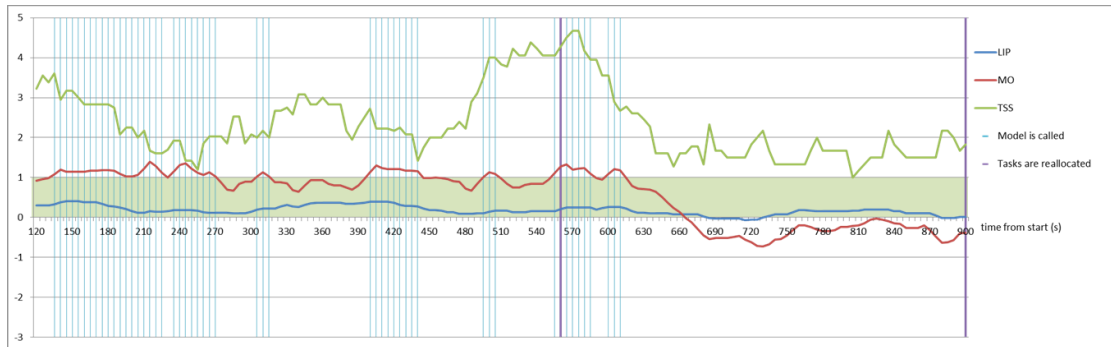
Participant 8 The CTL of participant 8 was mostly in a neutral zone (85%), sometimes in cognitive lock-up (14%), scarcely in underload (< 1%) and never in overload or vigilance. Tasks were reallocated a total of four times. The first reallocation is the same for all three levels and comparable in terms of timing, except in level two, where it occurred much earlier.



Participant 8, level 1: The first reallocation switches the LoA of Obs from 1 to 3



Participant 8, level 2: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

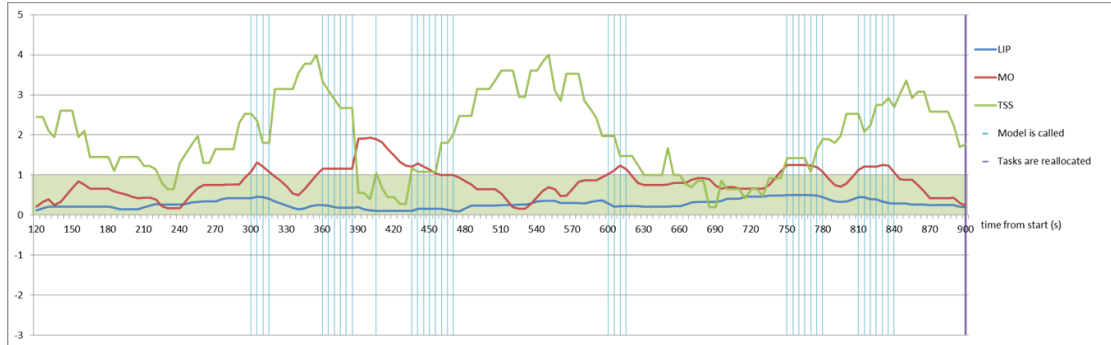


Participant 8, level 3: The first reallocation switches the LoA of Obs from 1 to 3.

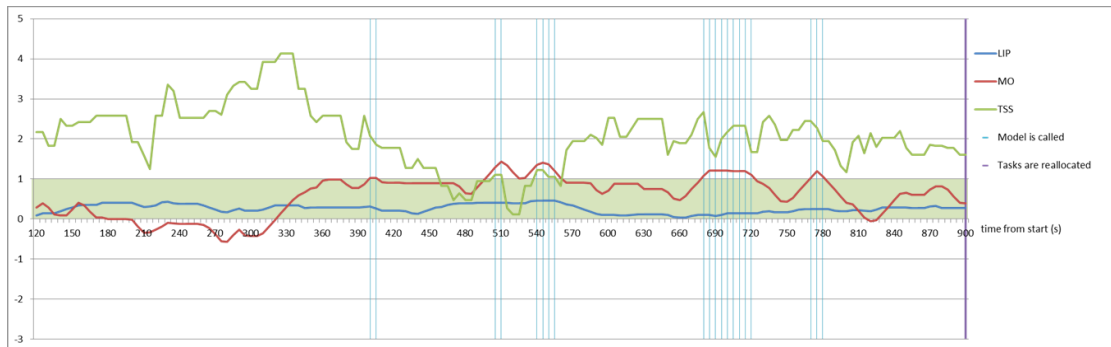
Participant 9 Inspection of the data of participant 9 after the experiment revealed that an error occurred that corrupted the data. The data of participant 9 was not used for analysis.

Participant 10 The CTL of participant 10 was mostly in a neutral zone (84%), sometimes in cognitive lock-up (16%), scarcely in overload (< 1%) and never in underload

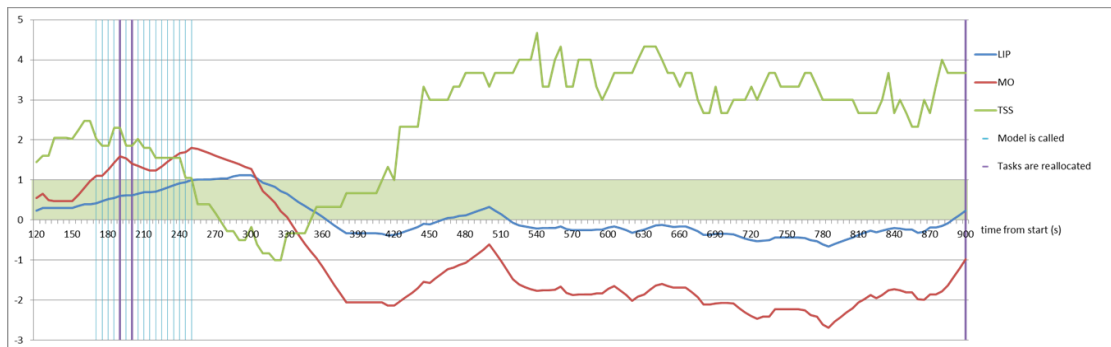
or vigilance. Tasks were reallocated a total of two times. In level three we see the same curious phenomenon (two subsequent task reallocations) as we saw for participant 1 in level two, the same reasoning as explained there applies for this case.



Participant 10, level 1: No reallocations occurred.

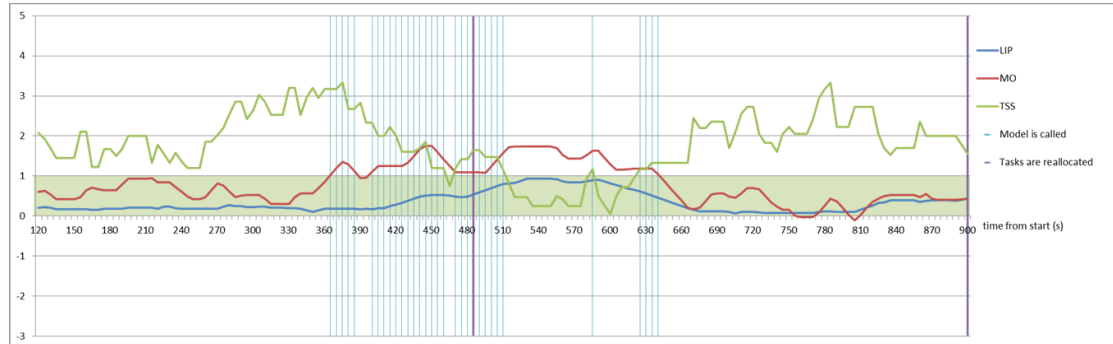


Participant 10, level 2: No reallocations occurred.

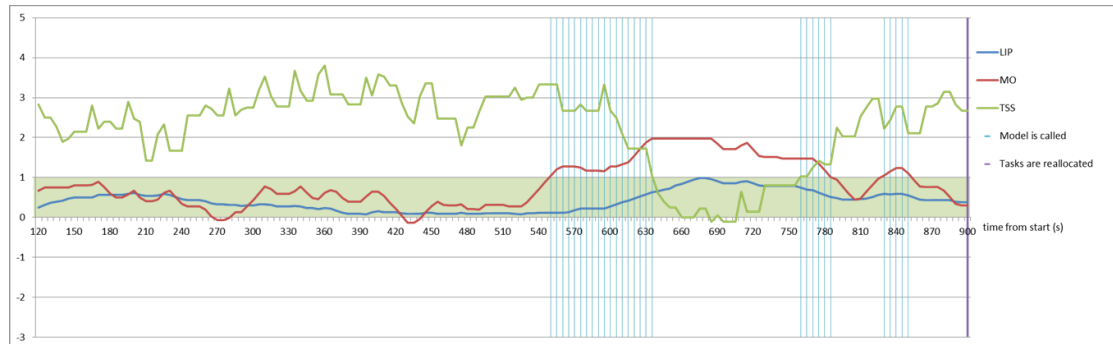


Participant 10, level 3: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

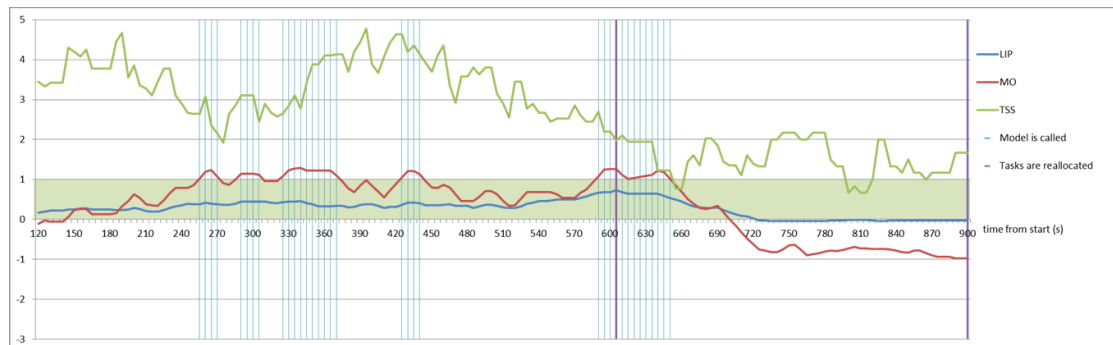
Participant 11 The CTL of participant 11 was mostly in a neutral zone (79%), sometimes in cognitive lock-up (21%) and never in overload, underload or vigilance. Tasks were reallocated a total of two times.



Participant 11, level 1: The first reallocation switches the LoA of Info from 1 to 2.

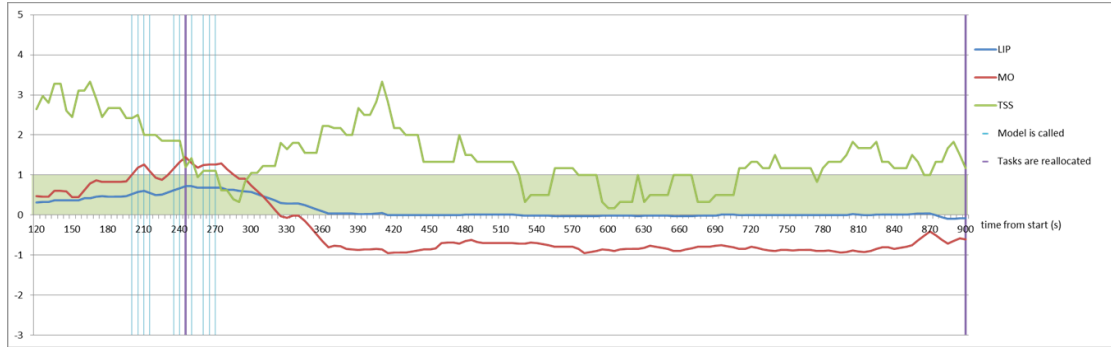


Participant 11, level 2: No reallocations occurred.

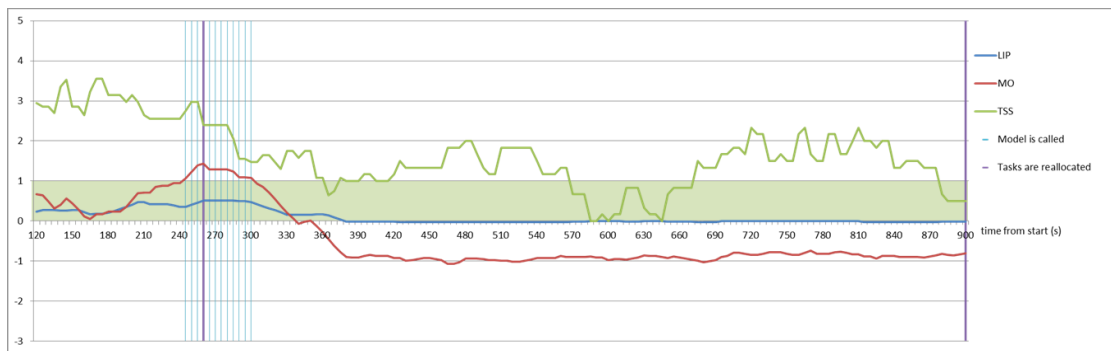


Participant 11, level 3: The first reallocation switches the LoA of Obs from 1 to 3.

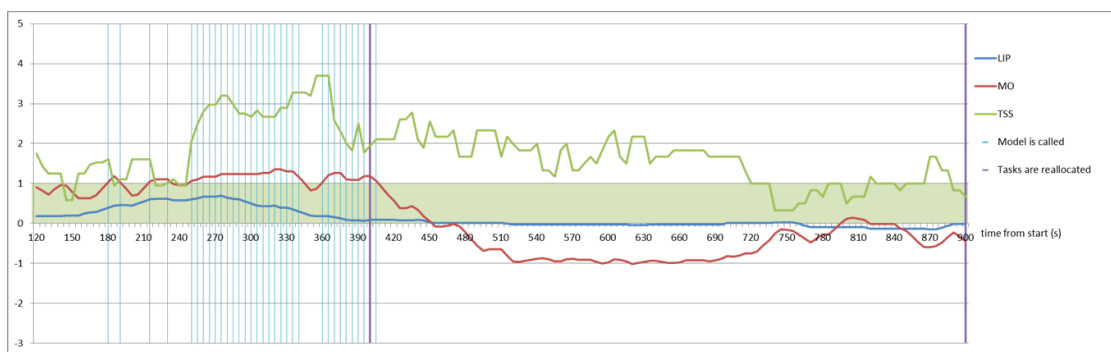
Participant 12 The CTL of participant 12 was mostly in a neutral zone (88%), sometimes in cognitive lock-up (12%) and never in overload, underload or vigilance. Tasks were reallocated a total of three times. The first and only reallocation is the same for all three levels and comparable in terms of timing.



Participant 12, level 1: The first reallocation switches the LoA of Obs from 1 to 3.

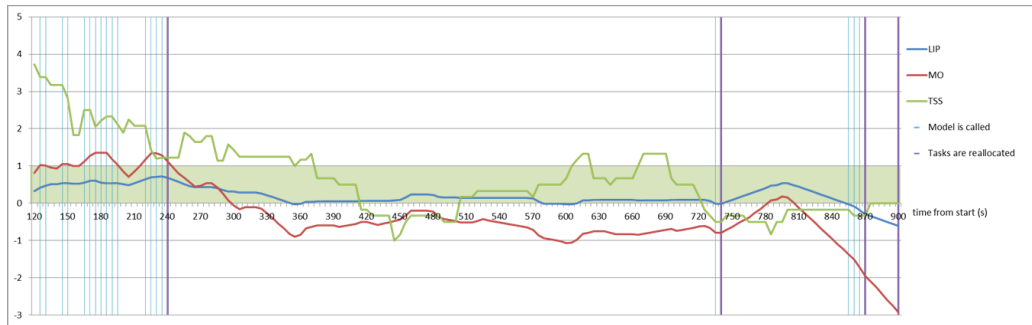


Participant 12, level 2: The first reallocation switches the LoA of Obs from 1 to 3.

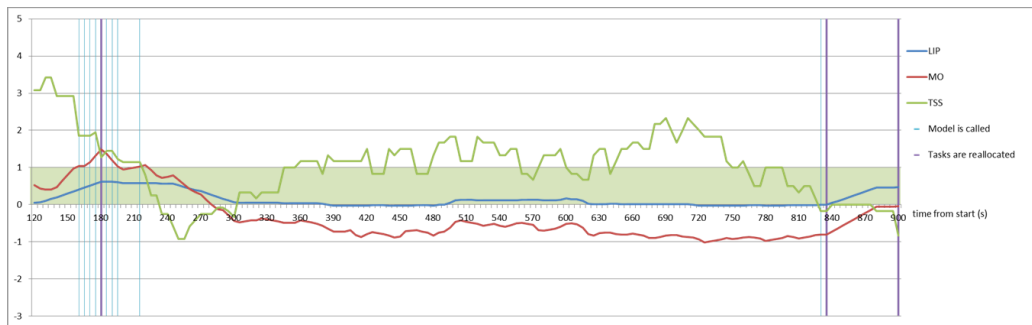


Participant 12, level 3: The first reallocation switches the LoA of Obs from 1 to 3.

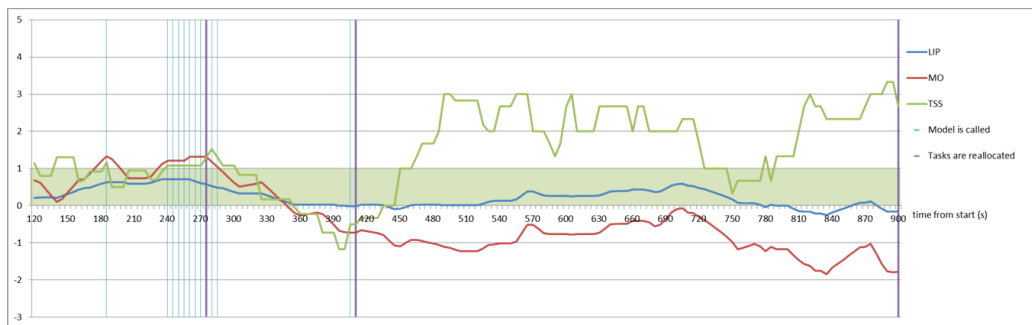
Participant 13 The CTL of participant 13 was mostly in a neutral zone (90%), sometimes in cognitive lock-up (8%), scarcely in underload (2%) and never in overload or vigilance. Tasks were reallocated a total of five times. The first reallocation is the same for all three levels and comparable in terms of timing. The second reallocation is also the same for all three levels, but differs in timing (in the second level it is too late to be suitable for analysis).



Participant 13, level 1: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

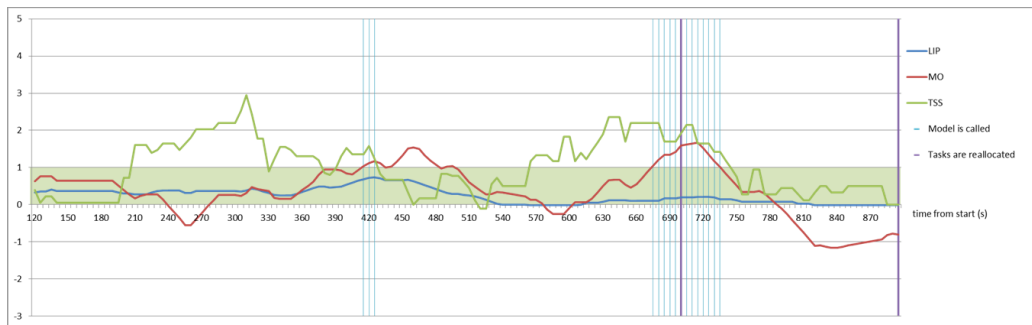


Participant 13, level 2: The first reallocation switches the LoA of Obs from 1 to 3.

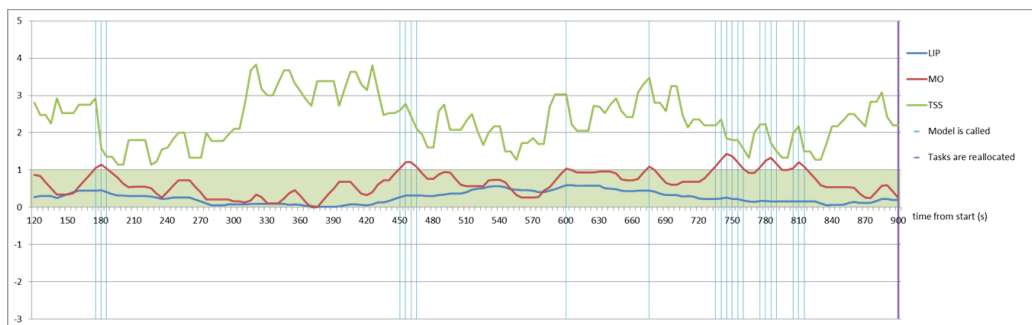


Participant 13, level 3: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.

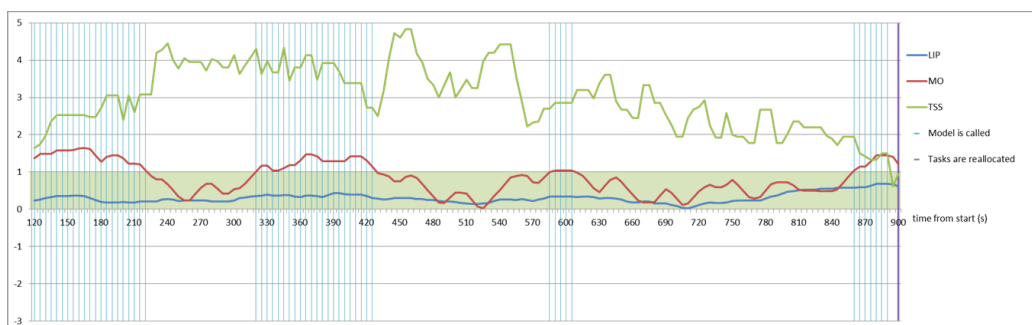
Participant 14 The CTL of participant 14 was mostly in a neutral zone (80%), sometimes in cognitive lock-up (20%) and never in overload, underload or vigilance. Tasks were only reallocated once in total, this was in the first level. Most likely, this difference is explained by the fact that in the first level the participant spend much time on adding obstacles to the tactical display (part of Obs). In the second and third level, the participant spend almost no time on adding obstacles to the tactical display, releasing the need for the robot to take over this task.



Participant 14, level 1: The first reallocation switches the LoA of Obs from 1 to 3.

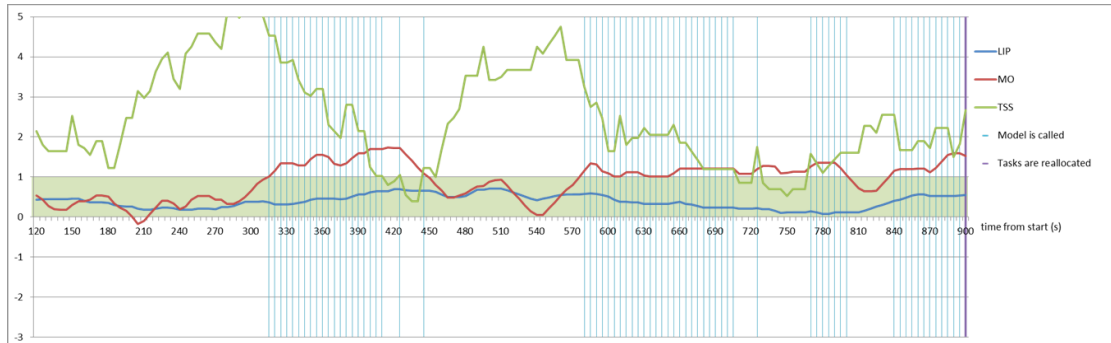


Participant 14, level 2: No reallocations occurred.

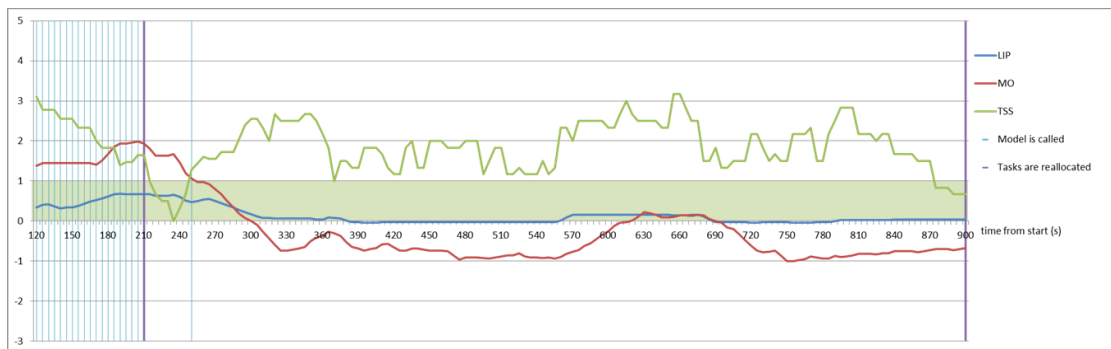


Participant 14, level 3: No reallocations occurred.

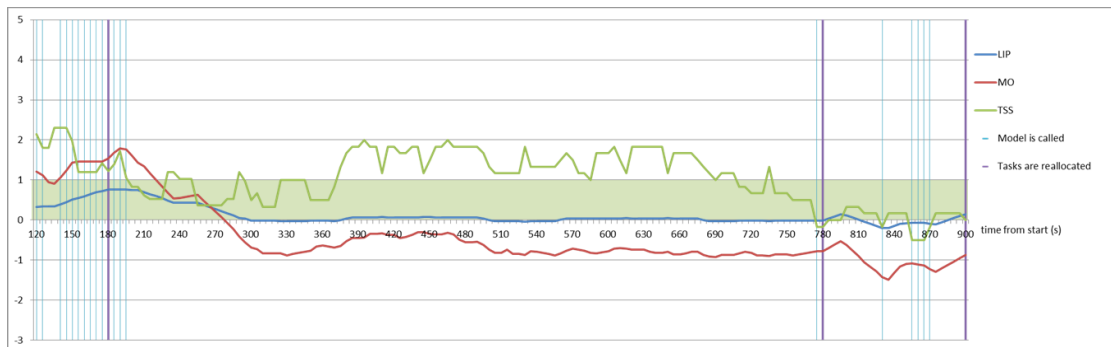
Participant 15 The CTL of participant 15 was mostly in a neutral zone (76%), sometimes in cognitive lock-up (22%), scarcely in underload (1%) and never in overload or vigilance. Tasks were reallocated a total of three times.



Participant 15, level 1: No reallocations occurred.



Participant 15, level 2: The first reallocation switches the LoA of Obs from 1 to 3.



Participant 15, level 3: The first reallocation switches the LoA of Obs from 1 to 3. The second reallocation switches the LoA of Nav from 1 to 2.