An Event-based Communication Concept for Human Supervision of Autonomous Robot Teams

Karen Petersen and Oskar von Stryk
Department of Computer Science
Simulation, Systems Optimization and Robotics Group
Technische Universität Darmstadt, Darmstadt, Germany
{*petersen*|*stryk*}*@sim.tu-darmstadt.de*

***Abstract*—The supervisor's understanding about the status of the robots and the mission is a crucial factor in supervisory control, because it influences all decisions and actions of the human in charge. In this paper, the concept of situation overview (SO) is presented as an adequate knowledge base for a human supervisor of an autonomous robot team. SO consists of knowledge about each individual robot (robot SO) and overview of team coordination and the actions towards mission achievement (mission SO). It provides the human with relevant information to control a robot team with high-level commands, e. g., by adapting mission details and influencing task allocation in a manner that is applicable to different task allocation methods in general. The presented communication concept to obtain SO is based on events, that are detected using methods from complex event processing. These events are dynamically tagged to different semantical or functional topics, and are sent to the supervisor either as notifications of different levels, to inform the supervisor about the mission progress, unexpected events and errors, or as queries, to transfer decisions to the supervisor, to make use of implicit knowledge and the human's experience in critical situations. The robots' level of autonomy can be adapted using policies, that allow to take decisions either autonomously by the robots, or with support by the supervisor, using different query modes. The concept can be applied to fundamentally different problem classes involving autonomous robot teams and a remote human supervisor. The application of the concept is discussed for the two example scenarios of urban search and rescue and robot soccer.**

***Keywords*-human-robot team interaction; supervisory control; situation overview; complex event processing; policies**

I. Introduction

Teams of autonomous robots have the potential to solve complex missions such as urban search and rescue (USAR), but are not yet sufficiently reliable and powerful to operate without any human supervision. However, humans and robots have many complementary capabilities, which can contribute significantly to a successful and efficient mission achievement if utilized properly in combination. For example, humans are very good at cognitive tasks such as visual perception, coping with unfamiliar or unexpected situations, and prediction of future states of the world based on incomplete knowledge of the current state. Robots, in contrast, have their strengths, for example, in fast execution of well-defined or repetitive tasks, evaluation and storage of large amounts of data, or operation in areas inaccessible to humans, like narrow spaces, in the air, or contaminated areas. This complementarity has already been observed when comparing humans and machines almost 60 years ago (c. f. Section IV-A), which leads to the assumption that this situation will not change significantly in the near future despite tremendous technological developments. Therefore, these specific strengths should be used efficiently for human supervision of autonomous robot teams.

The proposed communication concept has first been presented in [1]. This article is a revised and extended version of [1], providing more details about the applied methods and the underlying concept of situation overview. In particular, a detailed definition of situation overview is provided and is contrasted with situation awareness, which is usually applied as a knowledge base for robot teleoperation. Furthermore, the applied tagging and policy concept is specified in more detail. Additionally, more detailed examples for event-based communication are provided.

A. Abilities and Scenarios for Team Interaction

The proposed concept addresses scenarios, where a team of autonomous robots can be supported by a human supervisor with high-level instructions. The main goal of the robots (called *mission*) can be subdivided into tasks, that may be known prior to the mission start or can emerge during the mission. Each task can be fulfilled by a single robot, but not every robot is able to achieve each task. A task allocation method is used to decide which robot works on which task. This can be either a centralized planner for the whole team, or a distributed algorithm, where the robots negotiate the tasks among each other. The choice of an appropriate task allocation algorithm depends on the concrete mission setup, robots and environmental conditions.

Common teleoperation interfaces require one operator per robot, which implies that having a team of robots also requires a team of operators. If a single human supervisor shall be enabled to control a whole robot team, a fundamentally different approach is needed.

Following the definition of supervisor and operator from Scholtz [2], the main difference between these two in-

teraction roles is, that the supervisor usually interacts by specifying goals and intentions, while the operator interacts at the action level. Due to an increased robot autonomy, the supervisor has a lower workload per robot, and can therefore handle more robots simultaneously than an operator. High-level commands from the supervisor in a USAR mission may be used, e. g., to confirm or discard a robot's object hypotheses (e. g., victims or fire), to classify terrain trafficability, or to specify regions that are to be searched first or for a second time by a specific robot. In a robot soccer scenario, supervisor interactions may include changing or adapting a team's tactic, or allocating specific roles to individual robots. Common for all applications is, that the supervisor should be enabled to modify the tasks' parameters and the allocation of tasks to robots, and to act as decision support for the robots, e. g., in case they are not granted sufficient authority or do not have sufficient information for good autonomous decisions.

B. General Concept

The goal of the presented concept is on the one hand to enable the supervisor to modify the mission's and tasks' details (including task allocation), and on the other hand to allow robots to transfer decisions to the supervisor, if they are not allowed or not able to decide autonomously. Mission and task allocation adaptations can be realized by introducing a layer between the task allocation and the task cost calculation, that modifies the cost calculated for executing a task. It should be noted that this approach can be applied to very different task allocation methods. However, to be able to take such decisions, the supervisor needs to be aware of the team's progress towards mission achievement and the current state of the world and the robots.

In this paper, we propose a method to create a basis for supervisor-initiated interactions with the robot team. The actual interactions are not covered here. Instead, we focus on the event-based communication between the human and the robot team, which provides the supervisor with relevant information about the robots and the environment, and enables robot-initiated interactions using queries to transfer decisions from the robots to the supervisor.

In the next section, we introduce the term *situation overview (SO)*, which includes more general knowledge about the world and the robots' status, compared to situation awareness (SA), which is usually applied for teleoperation tasks. SO is a basis for all supervisor actions. Achieving SO is a nontrivial task, especially when dealing with a robot team instead of a single robot.

For obtaining SO, discrete events instead of continuous data streams are used as communication basis between robots and supervisor. The information required for obtaining SO is usually not fixed, instead, the communication has to be adopted during runtime to the specific needs of different missions and supervisors. For this purpose, we propose to control the amount of information using policies, which define the events to be detected by the robots using complex event processing. These events are then classified according to their priority and are sent to the supervisor as notifications (to provide information) or queries (to ask for decision support).

A main advantage of SO over SA is the reduction of data that has to be communicated. This is an important factor in real-world applications where the available communication bandwidth is usually limited. Additionally, it addresses robot teams, not just single robots. SO, obtained with the presented methods, gives a human supervisor a basis for high-level team interactions, without overburdening the human with too specific information of each robot.

This procedure follows the same idea as teamwork in human teams, where a common approach is to let a team leader maintain an overview of the project's progress [3]. For example, the rescue personnel at a rescue site informs the task force leader about their progress (e. g., an area has been searched) and the search team's status (the team is available again). This concept is *"simple, reliable, and reduces information overloading and distractions to decision-makers"* [4]. In contrast to the concept provided here, in human teams the leader obtains the information from other humans, who can reason about the value of an information, to decide if it is relevant for the team leader. However, if instead a team of robots shall inform a leader about their progress to support the leader's SO, they cannot judge a situation in the same way as a human, and therefore need some rules about what information they have to send to the supervisor and in which context. This problem is addressed by the presented communication concept.

The rest of this paper is organized as follows: In Section II, the term situation overview is introduced and contrasted with other widespread notions for describing a human's knowledge about a semi-autonomous system. Related work is discussed in Section III. In Section IV, first the interactions among humans in loosely coupled teamwork are observed. Second, inspired by these findings, the methods enabling the robots to send notifications and queries to the human supervisor are described. For detecting the important incidents, methods from complex event processing are applied. The events are classified with different levels to allow filtering and discriminative representations at a user interface. The amount of messages can be controlled using policies, which can be adapted either manually or automatically, depending on the supervisor's workload. Some application examples, for general robot team applications and for concrete scenarios of USAR and robot soccer, are given in Section V. The methods are discussed and future work is described in Section VI.

II. Situation Overview

In research on robot teleoperation, the operator's required knowledge about the robot's state and the environment is called situation awareness (SA), which is adopted from pilot situation awareness and is usually measured with the same tools [5].

The term SA was defined by Endsley as *"the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future"* [6]. To achieve SA, a human has to pass three levels:

- *Level 1 SA* is the perception of status, attributes, and dynamics of relevant elements in the environment,
- *Level 2 SA* is the understanding of the meaning and the correlation of the perceived level 1 elements,
- *Level 3 SA* is the projection of the current state into the future.

Level 2 and 3 can usually be achieved easier by experienced users than by novices, because they can base their knowledge on previous experiences and already have developed detailed mental models of the system [6].

Because this definition is derived from the egocentric SA of a pilot in an aircraft, also SA for a robot operator is egocentric from the point of view of a single robot. A generalization to a whole team of robots is difficult, because usually a human cannot track such detailed information for many robots simultaneously. Instead, the presentation of all SA elements of many different robots to a single operator can quickly lead to information overflow [4].

In the USAR domain, good SA is usually associated with a complete knowledge of the status of a robot and its surrounding. This includes the robot's health (e. g., the functionality of all sensors, the battery level, software failures, mechanical breakdowns), the robot's direct surrounding (e. g., the structure of the terrain, nearby obstacles, objects of interest such as victims or hazards, other robots or humans in the vicinity), and the relation between the robot and the environment (e. g., the robot's position and 3D orientation with respect to a fixed coordinate system, the distance between the robot and the obstacles, maneuverability of the robot on the given terrain). This information is usually gathered by full video streams of the cameras or live map-data, which produce a large data volume.

Obviously, a problem occurs if SA is applied to supervisory control of semi-autonomous robot teams: A single human is not able to obtain SA for several robots simultaneously and maintain SA for a long time. However, a supervisor does not need to obtain such a detailed SA, because the tasks of a supervisor, and therefore also the interactions with the robots, are fundamentally different compared to the interactions between an operator and a teleoperated robot. This shows, that supervisory control demands for a different definition of the human's required knowledge about the system, that is more specifically designed for multi-robot teams, instead of single robots.

Drury et al. [7] use the term *Humans' overall mission awareness* to describes the humans' overview of the activities in the team and the teams' progress towards the mission goal. However, this definition is rather loose, and it seems that it only includes the teams activities, but not the status of the individual robots, or the reasonable teamwork among the robots.

Hence, both definitions (situation awareness and mission awareness) do not provide an adequate basis for high-level interactions between a human supervisor and a robot team. To overcome this, we introduce the notion of *situation overview (SO)*.

SO consists on the one hand of information related to the mission progress, and on the other hand of information concerning each robot in the team.

- *Mission SO* is the supervisor's knowledge about the mission progress in general, and the knowledge about which robot contributes to the mission progress by taking which actions, and the reasonable coordination among the team members.
- *Robot SO* describes the understanding the supervisor has about a robot's location and status, its current and planned actions, including the robot's ability to accomplish the current task. This includes the knowledge about a robot's health, i. e., if the robot is working correctly or if it needs support from either the supervisor or from an operator.

Robot SO is related to each robot's individual performance, whereas mission SO is related to the team performance, which is not necessarily the sum of the individual's performance. As an example, consider a team of robots, that has the task to explore and map an office building. On the one hand, in case all robots cluster in the same room and all follow the same exploration pattern, instead of coordinating and distributing to different rooms, the overall team performance is poor, even though each individual robot shows a good performance. On the other hand, if the team performance is good, a malfunctioning robot would not be recognized if the individual robot performance is disregarded. Therefore, mission SO and robot SO complement each other and should usually be treated together as *situation overview*.

A supervisor, who has obtained mission SO, should be able to answer the following questions:

- Is the mission advancement as planned? Are there any complications? Is it expected that all deadlines are met? Is there anything that cannot be achieved?
- Which tasks still have to be done to accomplish the whole mission? Are any problems expected for the execution of these tasks?
- Do the robots coordinate properly? Are they (as a team) working efficiently towards the mission goal? Are the

tasks allocated to the robots in a way that matches the robots' capabilities and current status?

With a good robot SO, the supervisor should be able to answer the following questions for each robot:

- Are all sensors and actuators of the robot functioning correctly? Is the battery level high enough? Which parts are not functioning? Can it be repaired?
- Is the robot idle? If yes: why?
- Can the robot fulfill its current task satisfactorily? Does it need any support, or should it even be released from this task? In which way does the current activity advance the mission goal?

Generally spoken, the supervisor should have an overview of the team's and robots' activities and status, but does not need to know specific details. The important part is to quickly recognize deviations from the "normal" status, that require supervisor interaction.

Good SO includes on the one hand knowledge about the current state of the robots and the mission, which requires knowledge about the past developing to know how the current state has to be interpreted, and on the other hand assumptions about the future state of robots and mission, to enable the supervisor to interact with the robots and correct the team's behavior as early as possible.

Compared to Drury's mission awareness, mission SO includes additional information about team coordination and overall team progress. Compared to Endsley's situation awareness, robot SO includes less details about an individual robot, which accounts for the different requirements of a supervisor and an operator, and allows a single human to maintain information of a whole robot team. However, because SO is, like SA, based on the perception and projection of information, many research results from SA can also be transferred to SO. The following results from [6] also apply to SO:

- It cannot be defined in general, for different applications and robots, which information the human needs to receive to achieve SA (SO).
- SA (SO) is influenced by several human factors, like attention, working memory, workload and stress.
- The possibility to obtain SA (SO) is dependent on which information the system provides, and how this information is presented.
- Trained or experienced users can achieve a high SA (SO) easier than novices. Furthermore, some people are in general better in obtaining SA (SO) than others.
- Achieving SA (SO) is a process over time, not a discrete action.
- Good SA (SO) can increase system performance, but bad performance does not necessarily indicate bad SA (SO).

Endsley concludes, that there are two factors in a system, that influence how good a human can obtain SA: 1) which information does the system provide, and 2) how the available information is presented to the human. This also applies to SO. In this paper, a method is presented, that addresses the first factor: Which information shall the robots send to the supervisor, while on the one hand providing all relevant information, but on the other hand not sending information that does not advance the human's SO to prevent information overflow and reduce the required network bandwidth.

III. Related Work

Especially in the USAR domain, much research has been done on teleoperation interfaces, e. g., [8], [9]. These strongly rely on video and map data, that need to be sent from the robot to the user interface in real-time. On the one hand, this allows to accurately control a robot even in unstructured and complicated environments, but on the other hand, those interfaces cannot be extended easily to control more than one robot simultaneously, and require high bandwidth, which is often not permanently available in real-world scenarios. Further, most teleoperation interfaces require extensive operator training and continuously demand maximum concentration of the operator, hence quickly leading to task overload and operator fatigue.

Approaches that allow a single supervisor to deal with robot teams and do not require continuous high bandwidth communication can be found in the area of sliding autonomy or mixed initiative. In [10], Markov models are used to decide whether a robot works on a task autonomously or is being teleoperated by an operator. This requires continuous communication connection only during the teleoperation phases. The mixed initiative system presented in [11] allows the operator to manually switch between autonomy modes, where the operator input varies from goal input to full teleoperation. Similarly, in [12], the operator can assign waypoints, move the camera, or completely teleoperate a robot. With the augmented autonomy approach used in [13], the robots in an exploration scenario can either select their next waypoints autonomously, or the operator can assign waypoints. Results show, that these methods are appropriate to deal with a larger number of robots and can produce much better results than fully autonomous or purely teleoperated systems. However, they still require periods of continuous high-bandwidth connection, and can hardly be extended to fundamentally different scenarios, where the main focus is not on search or exploration.

A completely different approach is described in [14], where the robots can ask questions to the human supervisor. Similarly, in [15], the human is treated as a source of information for the robots. The level of autonomy is controlled by adjusting the costs to contact the supervisor as decision support. The teleautonomous system presented in [16] enables the robots to detect situations where human intervention is helpful, which are in this context the states of robot stuck, robot lost or victim found. Human supported

decision taking is presented in [17], here two variants are proposed: management-by-exception, where the operator can veto against an autonomous decision, and management-by-consent, where the operator needs to confirm an autonomous decision before execution. In [18], policies are used to restrict the autonomy bounds of the robots, in this context also rules are defined about which messages the robots are required to send to the human. These approaches are promising to be applicable to larger robot teams in real-world environments, because they do not require continuous human attention to a single robot and require less bandwidth as they do not rely on video streams. However, they are still not very flexible to be adapted to fundamentally different scenarios or for on-line adaption to different operator preferences. Furthermore, the events that require operator intervention are detected manually, and yet no method has been provided to flexibly detect complex events in arbitrary complex situations.

IV. Control of Communication between Robots and a Supervisor

In this section, the teamwork among humans, that inspired the new communication concept, is described briefly. Afterwards, the specific strengths of humans and robots, that contribute to these kinds of scenarios and interactions are revised. Finally, the methods used to realize a flexible communication between the robots and the supervisor are presented and discussed.

A. Interactions in Team Work Among Humans

When observing interactions in loosely coupled work-groups [19], some commonalities can be observed regardless of the scenario, e. g., home care, knowledge work, firemen in a search and rescue scenario, soccer players coordinating with each other and getting instructions from a coach, or people in an office preparing an exhibition at a fair: In all these situations, the overall mission is subdivided into tasks, that are assigned to the team members. Everyone works on his own tasks, and reports the progress to the teammates or the leader, either explicitly by verbal or written communication, or the progress can be directly observed by the others. If someone has problems in fulfilling a task, he can ask somebody else (who is expected to be more capable for this specific problem) for support.

To understand the benefits of supervisory control, it is important to be aware of some fundamental differences between humans and robots. In [20], the superiorities of humans over machines and vice versa are discussed. One of the main outcomes is that machines are good in fast routine work, computational power and data storage, while humans' strengths are perception, reasoning, and flexibility. These findings (although almost 60 years old!) are in most points still valid and can be transferred to a large extent from machines to robots. Especially the superiority of humans over robots in problem solving and situation overview is crucial, and is not likely to change in the near future. Further, although there are several sensors that allow robots to perceive data that humans cannot sense directly (e. g., distance sensors, infrared sensors), humans are much more capable in interpreting data, especially images.

As a conclusion, if a human supervisor is aware of the overall situation, but not necessarily of all details, it makes sense to leave some high-level decisions to the human, who can be expected to decide based on implicit knowledge, situation overview and experience, that cannot easily be added to the robots' world model. Due to the complementary capabilities of robots and humans, it can be expected that humans can cope well with the problems that robots cannot solve autonomously.

If this model of human teamwork is applied to human-robot interaction, with the human taking the role of a supervisor, the robots are required to report their progress and unforeseen events to the human, and ask for support if they cannot solve their tasks sufficiently well autonomously. This is enabled by the proposed communication concept using the following methodologies: First, important or critical events are detected using complex event processing (Section IV-B). These events are dynamically mapped to semantic tags (Section IV-C), and are classified to message classes, which are different levels of notifications and different query modes, according to their criticality (Section IV-D). Finally, the message flow is controlled by policies, that define which messages need (not) to be sent to the supervisor (Section IV-E).

B. Complex Event Processing

The events to be detected by the robots can be very diverse to many aspects. Some are just special variables exceeding thresholds, others are patterns that have to be detected, or several occurrences of different events simultaneously. Certainly, the detection of every single event could be programmed manually, but this is very time consuming, can lead to many failures, and usually duplicates lots of code.

The research field of Complex Event Processing (CEP) deals with such questions, of how to detect events in communication systems [21], for example in databases or Wireless Sensor Networks (WSNs). In WSNs, the challenge is to use several hundreds of distributed sensor nodes to detect events, e. g., human presence or fire, and combine simpler events to detect complex events, that are aggregations or patterns of several events. *Simple events* are discrete events, that can be directly detected without aggregating more information, e. g., a variable exceeding a threshold, or a sensor (not) delivering data. *Complex events* are events that are composed of two or more (simple or complex) events, or events enhanced with external information. These compositions can be two events occurring simultaneously, an event chain, patterns, etc. To describe those aggregations, event algebras are used,

e. g., HiPAC [22], SNOOP [23], REACH [24]. Those algebras provide operators as conjunction, disjunction, sequence, etc., to combine two or more events to a complex event. They vary in complexity and versatility. Depending on the application, an appropriate algebra needs to be chosen, that satisfies all needs, but is not too complex, hence being more difficult to understand and leading to higher implementation efforts.

The analogy between CEP as used in WSNs and robotics is, that there are several sensors and pre-processed data available, based on this information certain events or states of the robot or the world have to be detected. The key differences are, that a robot has less, but more reliable sensors than in a typical WSN, the sensors are more complex and deliver not only scalar values. Furthermore, the "network" is more static, apart from sensor failure, because a robot's sensors are physically connected and not entirely distributed. Therefore issues like time synchronization and timeliness can be disregarded for CEP on robots. However, the tasks and capabilities of a robot team are fundamentally different from those of a WSN: robots can physically interact with the environment in time and space, while a WSN can only monitor the state of the environment over time. This allows to base expectations about changes in the environment on the actions of the robots. Furthermore, the robots' mobility allows to systematically collect data at locations where a high information gain is estimated. In case also events have to be detected that involve more than one robot, also the WSN aspects of synchronization and timeliness have to be considered, which is usually done by the robots' middleware. Overall, CEP provides good methodologies, that can be used efficiently not only in databases and WSNs, but also on robots.

CEP allows to detect events on different semantical levels, corresponding to the three SA levels: perception of the current status, interpretation of the current status, and projection of the current and past events into the future. The semantically most basic events correspond the the current state of each robot, and help the supervisor to obtain SO corresponding to the first level of SA. Correlations between these events can be modeled, e. g., the simultaneous occurrence of two different events, using CEP. This supports the supervisor in understanding the meaning of the more basic event, which enhances the SO corresponding to the second level of SA. If the correlations and the meaning of the events are modeled carefully, the supervisor usually does not need to perceive the underlying basic events separately, which allows on the one hand to save communication bandwidth, and on the other hand to reduce the risk of information overload. Finally, complex event operations can be defined to model the future state of different components. Depending on the input data, for example stochastic models, regression functions, or more sophisticated models can be applied as prediction tools. This is usually not done in WSNs, because the sensor nodes typically don't have enough computational power for extensive calculations.

For the supervisor's robot SO, each robot can individually detect relevant events, report the current state and make predictions about the future state. Events that are relevant for the supervisor's mission SO are based on the one hand on the modeling of the robots' current mission, and on the other hand on each robot's current and past activities and plans. Therefore, events from different sources have to be combined for enhancing the supervisor's mission SO. This can either be done locally by one of the robots, or an event detection module has to be active at the supervisor's computer. The former scales better for large teams, because all calculations can be distributed over all team members. However, with the second possibility, the events are detected at the location where they are needed, and the required data volume is still low, because the events for mission SO are compositions of SO events from different robots, and do not rely on raw data.

Some examples of important events in a USAR mission are of course if a robot has detected a potential victim or a fire, but also reports about the status of the exploration, e. g., if a room has been explored completely without finding a victim. All these examples are relevant for robot SO as well as for mission SO. In a humanoid robot soccer match, a robot can monitor the frequency of falling when walking or kicking, taking into account disturbances by teammates or opponents (e. g., by pushing), and can deduce if it is still capable of playing efficiently. This information is primary relevant for robot SO. The goalkeeper can monitor its benefit to the match, if it observes the frequency of jumping to catch the ball, compared to the number of goals scored by the opponent, i. e., if the goalkeeper jumps for the ball, and no goal is scored directly afterwards by the opponents, the team presumably benefits from the goalkeeper. If the opponents score, regardless of the goalkeeper jumping or not, the robot can potentially contribute more to the team's success when acting as a further field player. This information is relevant for mission SO, because it affects the teams overall performance, but it is not relevant for robot SO.

C. Event Tagging

For allowing a human supervisor to quickly manage, sort and access groups of events by topic, events can be tagged. These tags can, for example, reflect the different tasks the robots are working on, functional or mechanical components of the robots, or more high-level topics like a robot's status or goals. Therefore, the tagging allows to match events to the two components of SO, namely mission SO and robot SO. Because the tags can describe different levels or overlapping topics, it becomes clear that one tag per event is not sufficient, hence, each event can have several tags.

As examples in a search and rescue mission, there could be events related to victim detection, events related to simul-

taneous localization and mapping (SLAM), events related to the vehicle's health, or event related to the vehicle's general mission progress. In this scope, the event that a robot has found a potential victim is on the one hand tagged as victim detection, but on the other hand also as mission progress. Likewise, a defect of a laser range finder is related in general to robot health, but also affects the performance of the SLAM, and is therefore tagged to both topics.

To guarantee a high flexibility, the tags are not defined statically offline, but can be adapted during runtime. This allows the supervisor to define new categories on the fly, add event types to existing categories, or remove event types from single categories.

In addition to the manual tagging, some tags can also be generated and mapped automatically using name prefixes. This can be used if some events clearly belong to a main topic. For example, events related to victim detection in a USAR mission all have names of the form `victim.*` (e. g., `victim.found`, `victim.exploreHypothesis`, `victim.discarded`, etc.).

The mapping of events to tags is stored in a configuration file, so that it does not have to be repeated manually at every system restart. Only manually defined tags and mappings have to be stored, because the automatically generated tags can be reconstructed easily at every system start.

D. Event Classification

The supervisor shall be supported – and not confused – by the messages from the robots. To enable the user interface to prominently present critical messages and show other information when needed to obtain SO, the events are classified according to their importance and criticality. The queries are graded with different modes of action selection, depending on the desired degree of robot autonomy.

Proposed Levels of Notifications: The most prominent notification levels are the five stages as used, for example, for software development: debug, information, warning, error, and fatal. The concept provided here targets users unfamiliar with the implementation details of the robot control software, hence the debug-level can be omitted here, as these notifications would confuse the supervisor, instead of advancing the SO. Fatal are usually those errors, that cannot be handled properly and lead to program termination. Because these notifications can often not be communicated anymore, or can not be handled properly by the supervisor, also the fatal-level is omitted here.

In summary, there remain three notification levels, to be used by the robots: *information*, representing regular events (e. g., start or termination of execution of a task), *warning*, representing unexpected but noncritical events (e. g., task execution takes longer than expected), and *error*, representing critical events (e. g., sensor failures).

As examples for notifications, an information can be sent by a USAR robot, informing the supervisor that it has finished exploring a room without finding any victims. A warning can be sent by a soccer robot, that detects that it falls frequently without external influence and therefore cannot play properly. An error should be sent by a robot that detects that an important sensor, e. g., the camera or the laser range finder, does not deliver any or sufficiently meaningful data.

Types of queries: Robot-initiated interactions are enabled using queries based on the detected events. Recall, that supervisor-initiated interactions are realized with different methods, which are not covered in this paper. Depending on the desired degree of robot autonomy, there are several possibilities to take decisions. Besides deciding and executing everything autonomously, the supervisor can be integrated for confirming or vetoing decisions, or even for selecting the appropriate answer. Decisions that allow or require supervisor intervention are formulated as queries.

Three query classes are proposed:
(1) *Autonomous decision with veto:* The robot selects among several solutions, and does not start execution before a specific time t_{exec} has elapsed. The supervisor is given a time t_{veto} to contradict this decision. t_{exec} and t_{veto} are independent of each other, which means, if $t_{exec} < t_{veto}$, the supervisor can veto a decision even after the robot started execution.
(2) *Autonomous decision with confirmation:* The robot selects among several solutions and presents the selected solution and the alternatives to the supervisor. Execution does not start before the supervisor confirms or contradicts the selection.
(3) *Supervisor decision:* The robot provides several solutions to the supervisor, but does not preselect a solution. Execution starts after the supervisor selects and confirms a solution.

The robots are granted more autonomy in the first class, and less autonomy if confirmation by the supervisor is required. The second and third query classes make no difference for the robots, but for the human there is a psychological difference if a selection is proposed or not.

As an example, consider the goalkeeper from the example in Section IV-B. If this robot detects that it is either not needed (because the opponents do not shoot on the goal) or is not beneficial (because it cannot block the goal shots), the robot could instead act as an additional field player, to potentially contribute more to the team's success. Depending on how much autonomy is granted to the robot, this tactic change could either be autonomous with veto, or (to give the human more control) autonomous with confirmation.

E. Control of the Message Flow

The amount of messages that are sent to the supervisor needs to be controlled carefully. On the one hand, too many messages can result in information overflow and supervisor stress, or in complacency if most of the robot decisions are trivial, which brings the danger of overseeing wrong

decisions [25]. On the other hand, too few messages lead to a loss of SO. In general, there should not be any static rules about which events shall be communicated to the supervisor, and which decisions the robot should take autonomously or with some support by the supervisor. Rather, this is highly dependent on the current mission, the supervisor's preferences, and the supervisor's trust in the system.

In [18], policies are used to define the bounds of an agent's autonomy. Policies are positive and negative authorizations, that define what an agent is (not) allowed to do (A+ and A-), and positive and negative obligations, that define what an agent is (not) required to do (O+ and O-). Policies are applied to actions as well as to communication, e. g., sending acknowledgments when receiving new instructions. Within the scope of this paper, the only bound on autonomy is decision taking, and the communicativeness of the robots has to be controlled. Therefore it is sufficient to apply similar rules to regulate the amount of notifications and queries of the previous section.

By means of the tagged events (Section IV-C) and the different messages classes (Section IV-D), policies can be defined for groups of messages, according to their importance, according to a topic, or for single event types. Because only binary decisions have to be taken (send an event to the supervisor or not), only two different types of policies are used: send (S+) and do not send (S-). Compared to [18], S+ corresponds to an O+ policy, and S- to an A- policy. S+/- policies can be defined for single event types, tags, or importance levels.

Each event can have one of three different policy states: S+, S-, or D (default). If no policies are defined, all states are set to D. The default value can be defined centrally, and allows the supervisor to decide if the system behaves generally communicative or silent.

If a policy P is defined for a tag or a priority, the status of all events that are mapped to this property is set to P. In that way, the system behavior always complies with the most recent policies. Conflicting policies are resolved either by heuristics, or manually by the supervisor. If the old status of an event is D, no conflict occurs, and the status is simply overwritten. In case an event already has a policy S+ or S-, which is different to the new policy P, the status of this event is marked as conflicting. Table I shows an example for resulting event policies after defining policies for some tags. An "x" in the table indicates the mapping of an event to a tag. Initially, all event policies are set to the default value D. Two policies are defined sequentially. First, a policy S+ is defined for tag T_1. Because E_1 and E_3 are mapped to T_1, also the corresponding event policies are set to S+. Second, a policy S- is defined for tag T_2, therefore, also the event policy of E_2 is set to S-. For event E_3 a conflict occurs, because this event policy has been set to S+ because of T_1, and is now overwritten because of T_2. As described above, the event policy is preliminarily set to the most recent policy (S-), but is marked as a conflict (indicated by a * in the table) that has to be resolved by the supervisor.

If desired, the conflicts are sent to the supervisor as queries (autonomous decision with veto, with $t_{exec} = 0$ and $t_{veto} = \infty$). The first query allows the supervisor to decide to a) set all conflicted states to S+, b) set all conflicting states to S-, c) set all conflicting states to D, d) set all conflicting states to the most recent policy, e) set all conflicting states to the older policy, or f) decide individually for each event type. In the last case, a new query is generated for every event with conflicting state, allowing the supervisor to a) set the state to S+, b) set the state to S-, c) set the state to D. The preselection for all these queries is to set all conflicting states to the most recent policy (which caused the conflict). If no queries are sent, the events with conflicting states have to be highlighted at the user interface, to indicate the conflicts for the supervisor.

Different custom sets of policies can be stored, allowing to load specific settings for different supervisors, or dependent on the current scope of a mission. For example, if a human in a USAR mission has to supervise the victim detection, while other humans are monitoring the robots' health, a policy set can be loaded, that shows only events related to victim detection, and concerning the sensors used for victim detection. All other events, even critical events, can be disregarded, because they are outside the current scope. However, if the supervisor has to take over other tasks in addition, like monitoring the robots' health or the localization, the policies do not have to be adapted manually, instead another stored setting can be loaded.

In addition to manual policy settings, policies can also be adapted automatically, depending on the supervisor's current workload. For example, if there is a number of pending queries, only queries with supervisor selection or supervisor confirmation should be sent, because those with supervisor veto are expected to expire before the supervisor notices them. More sophisticated models, that involve other information like the mouse clicks, eye movements, or other stress indicators, could also be used here.

To illustrate the importance of user-dependent or automatically adapting policy sets, consider two examples of different applications: In the USAR scenario, a supervisor without trust in the robots' autonomous victim detection might want to get informed every time human-like temperature is detected with a thermal sensor, while a supervisor with more trust might be satisfied getting just the hypotheses that are positively verified by the robots. For the soccer scenario, if the supervisor is occupied with notifications about malfunctioning sensors or instable walking abilities, the queries about tactics changes can be omitted, because they are just of secondary importance if so many other problems have to be handled.

Tag Policy	S+ T_1	S- T_2	... ...	D T_M	Resulting Event Policy
E_1	x	-	...	-	S+
E_2	-	x	...	x	S-
E_3	x	x	...	-	S-*
...					
E_N	-	-	...	x	D

Table I
RESULTING EVENT POLICIES AFTER SEQUENTIAL DEFINITION OF TAG POLICIES.

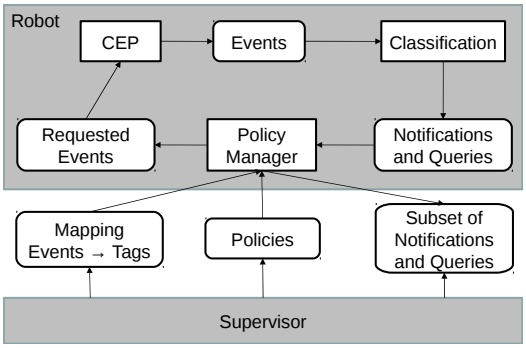


Figure 1. Visualization of the interactions among the different components of the proposed general communication concept

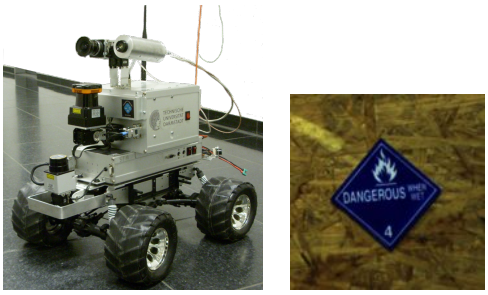
F. Discussion

All four methods applied here have been well established in entirely different fields. The new concept is, to combine them, and to use them to enable a human supervisor to obtain situation overview on a high level.

Overall, the four components are connected in a loop with external feedback from the supervisor, as shown in Figure 1: CEP detects important events, which are then classified to notification levels or query types. Based on the mapping of events to tags and the policies, the policy manager then decides which of those events are sent to the supervisor as messages or queries. For closing the loop, both the mapping between events and tags, and the policies can be adapted during runtime, either manually by the human or automatically, and therefore it changes dynamically, which events have to be detected by the CEP system. Compared to other approaches, the presented concept supports a more flexible communication, that allows to control on the one hand the supervisor's workload and the robots' autonomy, and on the other hand also the use of network capacity, if low bandwidth is an issue.

V. APPLICATION EXAMPLES

In this section it is demonstrated how the proposed approach can be applied to different scenarios from different applications. First, some general examples are given, that apply to arbitrary robot missions in general. Second, first steps of the integration of the concept into our USAR robot and our humanoid soccer robots are outlined.



(a) (b)

Figure 2. (a) Hector UGV. (b) Example of a hazmat sign.

A. Mission-independent Examples

With most robot user interfaces, the supervisor needs to be familiar with the system for deciding which system functions need to be monitored, and how they can be monitored. With the methods proposed in this paper, the robots provide methods to monitor themselves and can on the one hand inform the human about the status, and on the other hand send warnings if the status changes or is critical.

Before the start of a mission, all important sensors have to be checked for functionality. Instead of doing every check by hand – which is often omitted or only done for some samples to save time – this can be done automatically using CEP. A successful check results in an event of the type `sensor.check`, which is sent as information message. If a check fails, an event of the type `sensor.failure` is sent as error message, accompanied with an error description.

The battery status of every robot should be monitored continuously, to prevent malfunctions because of too low voltage or damaged batteries. Battery displays for each robot can be overlooked, especially if a single human has to monitor the battery status of many robots in parallel to several other monitoring or coordination tasks. With the methods presented in this paper, each robot can monitor its battery status individually, and can send a warning notification before the battery runs empty. The methods of complex event processing further allow to warn not before the voltage is constantly below a threshold for some seconds, and therefore is able to filter voltage peaks or faulty measurements.

As a general proposal, an unexperienced supervisor should start with no restricting policies, and then gradually constrain the messages, if they are not needed. On the one hand, this leads to lots of messages at the beginning, but on the other hand, the supervisor learns, which types of events are provided by the robots and can decide on this basis which messages are important for the current setup.

B. Example: Urban Search and Rescue

In the USAR setup, a team of heterogeneous, autonomous robots has to search for trapped victims in a partially collapsed building, e. g., after an earthquake, and to locate

Event	Relevant for		Notification	Payload
	Robot SO	Mission SO	Level	
Battery				voltage, estimated remaining runtime
Level	x		info	
Low	x		warn	
Empty	x		error	
TooLowForTask	x	x	warn	+ estimated task execution time
Sensor				sensor type
Check	x		info	
Failure	x		error	+ error message
Task				task name
Start	x	x	info	+ estimated execution time
Finish	x	x	info	+ result
Abort	x	x	warn	+ error message
LongExecution	x		warn	+ initially estimated execution time and actual time required so far
NoneSuitable	x	x	warn	+ reasons for not being able to execute remaining tasks
Victim				supporting sensor data (e. g. camera image)
ExploreHypothesis	x	x	info	+ location of victim and robot
Found	x	x	info	+ detailed sensor data and victim location
Discarded	x	x	warn	+ reason for discarding
SeeEvidence	x	x	info	+ evidence type, reliability
Localization				robot pose
Move	x		info	+ path and traveled distance
Lost	x		warn	+ potential alternative robot poses
Exploration				current map and frontiers
Finished		x	info	+ reason for remaining frontiers (e. g. not reachable)
NewGoal		x	info	+ position of exploration goal
Terrain				(Simplified) 3D point cloud
Ok	x	x	info	
Difficult	x	x	warn	+ problem description (high inclination, steps, ...)
Impassable	x	x	warn	+ reason and marking in point cloud
Progress				map
EnterRoom	x	x	info	+ room identification
LeaveRoom	x	x	info	+ room identification
Travel	x		info	+ distance traveled
NoProgress	x		warn	

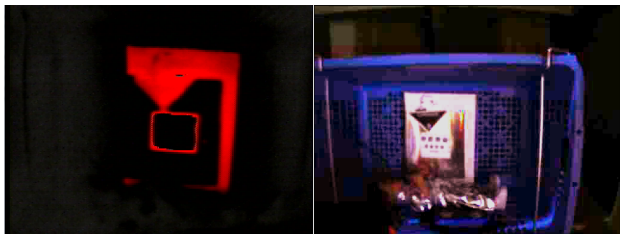
Table II
SUBSET OF THE EVENTS IN A USAR MISSION, WITH ASSOCIATED NOTIFICATION LEVELS AND PAYLOAD.

potential hazards like gas leaks or fire. The methodologies proposed in this paper apply to robot behavior that can be observed for example at a RoboCup Rescue competition, because in current real-world deployments the robots are not yet autonomous at all, while the proposed concept requires robot autonomy as a starting point.

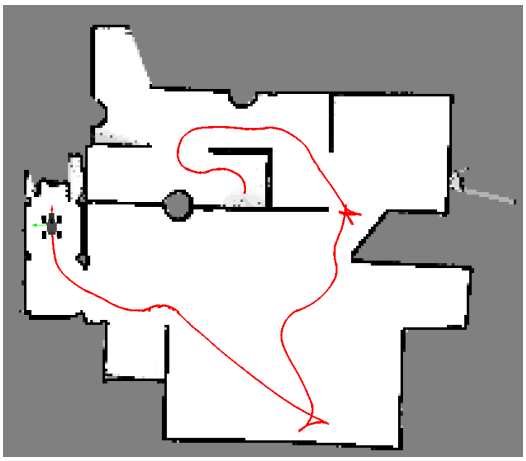
The results are discussed for the unmanned ground vehicle (UGV) of Team Hector Darmstadt [26] (Figure 2(a)). This robot can autonomously explore an environment, build a map based on 2D laser scans, detect uneven terrain like steps or holes using an RGB-D camera, and search for potential victims and markers that indicate hazardous material (hazmat signs, see Figure 2(b)) using an RGB camera and a thermal camera. A sensor fusion algorithm is applied that combines victim hypotheses from the daylight camera and the thermal camera with information from the laser range finder to build up a semantic map [27]. This algorithm is also suitable for more realistic conditions than RoboCup, i. e., to reliably find real people in environments that contain other heat sources than humans or shapes similar to humans.

Many user interfaces require the supervisor to request all relevant information from the robots manually, and only alert the supervisor when a robot has found a victim. The other extreme is, that an interface displays all information that could be of interest by default. In both cases, either some important data is potentially not monitored, or much information is sent continuously, even if it is not needed. For example, the operator requests the map generated by the robot and the camera images, but does not have a look at the output of the thermal sensor. If this sensor has a malfunction, it is potentially never noticed. We propose instead, to automatically provide the operator with relevant information, but filtering data that does not enhance the supervisor's SO.

Mission progress is usually monitored by looking at the camera images and the map in real-time. However, as the robots usually do not proceed very fast, not all information is needed all the time. With the methods provided here, it is possible to send an image of the map every time a progress is observed. Progress can be, for example, every time the robot traveled more than 3 meters, or every time a robot enters or leaves a room, which results in an event of, e. g., the type `enterRoom`, labeled to the general topic `progress`, and is published as information message. In addition, every time



(a)



(b)

Figure 3. (a) Images showing a simulated victim in the thermal image and the camera image, taken at the current robot position. (b) The current map learned by the robot.

a robot starts exploring a new victim hypothesis, this can be communicated, possibly together with attached sensor data that motivated the victim hypothesis. This results in an event of the type `victim.exploreHypothesis`, labeled as related to `progress` and (automatically) to `victim`, and is published as information message. In turn, this method also allows to detect a lack of progress (by observing that no progress events are detected for a predefined time, although the robot intends to proceed with its tasks), which can indicate a malfunctioning or disoriented robot. This is an event of the type `noProgress`, labeled with the topic `progress` and is published as a warning message. A supervisor who trusts in the robot's capabilities may not want to see all progress messages, but only those that refer to non-progress. To achieve this, two policies have to be defined: a S- policy for sending notifications labeled with `progress`, and a S+ policy for sending notifications of the type `noProgress`.

If a robot detects a victim, the resulting event is a supervisor decision query, where the supervisor can decide to (a) confirm the victim, (b) discard the victim, or (c) try to collect more information. The message also contains images from the cameras (see Figure 3(a)), and an image of the current map to display the location of the victim (see Figure 3(b)). The red line in Figure 3(b) shows the robot's traveled path. It can be seen that continuous monitoring of the map does not give more information to the supervisor than an image of the map every time a progress is observed or if the robot got stuck for a while, as it was the case in the upper right corner. Therefore, much communication overhead can be saved by omitting data transmissions that do not advance the supervisor's SO.

Queries can not only be used to let the supervisor confirm or discard potential victims or hazards, but also for decision support regarding path planning, e. g., an autonomous decision with veto can be sent, if the terrain classification is not confident enough and the supervisor should decide if a robot can negotiate an area or not.

A subset of all event types occurring in a USAR mission is shown in Table II. These events are grouped according to their automatically generated tags (e. g., `victim` or `battery`). Frequently, all events of a tag group have the same basic payload, and the single event types add some further information. For example, all `battery`-event types are accompanied with the current voltage and the estimated remaining runtime, while the `battery.tooLowForTask`-event provides additional information about the estimated task execution time. Furthermore, each event's relevance for robot SO and mission SO is marked in the table. It can be seen, that many events contribute to both parts of SO. However, for mission SO, the supervisor needs to receive the same event types from all robots in the team, not just from a single robot, to get a good overview about the team's activities and the quality of the coordination in the team.

C. Example: Humanoid Robot Soccer

In a RoboCup soccer match in the humanoid KidSize league (humanoid robots, 30-60 cm high), three autonomous robots per team play soccer on a 4x6 m large playing field. The goals, landmarks and the ball are color-coded. The robots are only allowed to have human-like sensors, restricting the external sensors mainly to directed cameras in the head and touch sensors. No human intervention is allowed during the game, except referee signals (start, stop, scored goals) and taking robots out of the game for service. This requires the robots to play fully autonomous, communicating with each other using WLAN. Therefore, the main challenges in this league are balancing (bipedal walking and kicking), self-localization based on the limited field of view of the camera, and coordination within the team. Two scenes from soccer matches of the Darmstadt Dribblers [28] at RoboCup 2011 can be seen in Figure 4.

Monitoring of the robots in a soccer match is usually done by visually observing the match, at the team Darmstadt Dribblers also the team messages sent between the robots are monitored. As direct human intervention is not allowed by the rules, the proposed concept can be used on the one hand for monitoring during the game and changing details

Figure 4. Two scenes from robot soccer matches in the RoboCup KidSize league, Darmstadt Dribblers are playing in cyan.

during game breaks, or on the other hand for tuning the robots during tests or practice games.

Monitoring the health of each robot could also be done visually, but with three or more robots on the field it is difficult to keep track of each robot's performance. With CEP, it is possible to monitor the falling frequency of each robot for different motions like walking or kicking, and its correlation with other factors like the vicinity of opponents or teammates (which could indicate that the fall was due to a collision), or motor temperature and battery status, which could be a reason to switch to a more robust behavior, e. g., dribbling instead of kicking the ball.

Further, the benefit of the specific roles can be monitored, like already proposed for the goalie in Section IV. This allows on the one hand to tune the parameters during tests to maximize each role's benefit, and on the other hand to quickly change tactics or preserve hardware during a match.

VI. Conclusion and Outlook

The communication concept presented in this paper is designed for interactions between a human supervisor and a team of autonomous robots. To make use of the specific complimentary strengths of humans and robots, supervisor interactions are focusing on high-level commands. Because SA is not an adequate knowledge base for a human supervisor of a whole robot team, the notion of SO is introduced, which consists of robot SO and mission SO, to enable the human to detect on the one hand problems related to individual robots an on the other hand performance decrements related to suboptimal team coordination. SO can be flexibly achieved for several fundamentally different scenarios using the presented methods, which are complex event processing, message tagging, message classification, and policies. This communication concept is inspired by loosely coupled human teamwork and requires a low communication overhead compared to standard teleoperation methods, because only data needed for SO are sent, while omitting details only used for exact teleoperation. The methods enable a human supervisor to gain a general SO of a whole robot team, without requiring the supervisor to be familiar with implementation details. The performance of the team can be enhanced by transferring critical decisions to the supervisor, because in this case the decision is based on SO, human experience and implicit knowledge, and is therefore expected to be more reliable and efficient for achieving the mission goal.

In general, an interface that is based on this new communication concept can provide a higher SO than standard interfaces, because the robots can send information that the supervisor would probably not request, hence problems and errors can potentially be noticed earlier. SO gives the supervisor a basis to take high-level decisions, e. g., for adapting task allocation or mission details. Preliminary results in USAR and robot soccer indicate the potential of the developed concept. These two fundamentally different setups demonstrate, that the proposed concept can be applied to a large variety of problem classes. It is furthermore planned to implement the whole concept, including the communication concept as well as high-level commands by the supervisor, for different scenarios with heterogeneous robot teams.

Future work includes experiments in simulation and with real robots to support the hypotheses and approach of this paper. It is planned to conduct user studies in different application scenarios to show the wide applicability of the proposed methods. Furthermore, the possibilities of the supervisor to coordinate robot teams based on the proposed situation overview will be examined. For dealing with larger robot teams, a basis for a large-scale interface is provided by the presented concept, as it offers the data for SO and supports efficient filtering.

Acknowledgments

This research has been supported by the German Research Foundation (DFG) within the Research Training Group 1362 "Cooperative, adaptive and responsive monitoring in mixed mode environments".

References

[1] K. Petersen and O. von Stryk, "Towards a general communication concept for human supervision of autonomous robot teams," in *Proceedings of the Fourth International Conference on Advances in Computer-Human Interactions (ACHI)*, 2011, pp. 228 – 235.

[2] J. Scholtz, "Theory and evaluation of human robot interactions," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.

[3] D. H. Fernald and C. W. Duclos, "Enhance your team-based qualitative research," *Annals of Family Medicine*, vol. 3, no. 4, pp. 360 – 364, July/August 2005.

[4] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, And Cybernetics – Part C: Applications and Reviews*, vol. 34, no. 2, pp. 138 – 153, May 2004.

[5] M. R. Endsley, "Situation awareness global assessment technique (sagat)," in *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*, vol. 3, 1988, pp. 789 – 795.

[6] ——, "Toward a theory of situation awareness in dynamic systems," *Human factors*, vol. 37, no. 1, pp. 32–64, 1995.

[7] J. Drury, J. Scholtz, and H. Yanco, "Awareness in human-robot interactions," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 1, 2003, pp. 912 – 918 vol.1.

[8] M. W. Kadous, R. K.-M. Sheh, and C. Sammut, "Effective user interface design for rescue robotics," in *HRI '06: Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*. ACM, 2006, pp. 250–257.

[9] C. W. Nielsen, M. A. Goodrich, and R. W. Ricks, "Ecological interfaces for improving mobile robot teleoperation," *IEEE Transactions on Robotics and Automation*, vol. 23, no. 5, pp. 927–941, October 2007.

[10] B. P. Sellner, F. Heger, L. Hiatt, R. Simmons, and S. Singh, "Coordinated multi-agent teams and sliding autonomy for large-scale assembly," *Proceedings of the IEEE - Special Issue on Multi-Robot Systems*, vol. 94, no. 7, pp. 1425 – 1444, July 2006.

[11] J. W. Crandall and M. A. Goodrich, "Experiments in adjustable autonomy," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, 2001, pp. 1624 –1629.

[12] J. Wang and M. Lewis, "Human control for cooperating robot teams," in *HRI '07: Proceedings of the ACM/IEEE international conference on Human-robot interaction*. ACM, 2007, pp. 9 – 16.

[13] Y. Nevatia, T. Stoyanov, R. Rathnam, M. Pfingsthorn, S. Markov, R. Ambrus, and A. Birk, "Augmented autonomy: Improving human-robot team performance in urban search and rescue," in *Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, Sept, 22-26 2008, pp. 2103 – 2108.

[14] T. Fong, C. Thorpe, and C. Baur, "Robot, asker of questions," *Robotics and Autonomous Systems*, vol. 42, pp. 235 – 243, 2003.

[15] T. Kaupp and A. Makarenko, "Measuring human-robot team effectiveness to determine an appropriate autonomy level," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 19 - 23 2008, pp. 2146 – 2151.

[16] R. Wegner and J. Anderson, "Balancing robotic teleoperation and autonomy for urban search and rescue environments," in *Advances in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 16–30.

[17] M. L. Cummings, S. Bruni, S. Mercier, and P. J. Mitchell, "Automation architecture for single operator-multiple uav command and control," *International Command and Control Journal*, vol. 1, no. 2, pp. 1 – 24, 2007.

[18] M. Johnson, P. J. Feltovich, J. M. Bradshaw, and L. Bunch, "Human-robot coordination through dynamic regulation," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 19 - 23 2008, pp. 2159 – 2164.

[19] D. Pinelle and C. Gutwin, "A groupware design framework for loosely coupled workgroups," in *Proceedings of the Ninth European Conference on Computer-Supported Cooperative Work*, Paris, France, 18-22 September 2005, pp. 65 – 82.

[20] P. M. Fitts, Ed., *Human engineering for an effective air-navigation and traffic control system*. Washington, DC: National Academy of Sciences Archives, 1951.

[21] A. Hinze, K. Sachs, and A. Buchmann, "Event-based applications and enabling technologies," in *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, 1:1–1:15, July 2009.

[22] U. Dayal, A. Buchmann, and D. McCarthy, "Rules are objects too: A knowledge model for an active, object-oriented database system," in *Advances in Object-Oriented Database Systems*, ser. Lecture Notes in Computer Science, K. Dittrich, Ed. Springer Berlin / Heidelberg, 1988, vol. 334, pp. 129–143.

[23] S. Chakravarthy and D. Mishra, "Snoop: an expressive event specification language for active databases," *IEEE Data and Knowledge Engineering*, vol. 14, no. 10, pp. 1 – 26, 1994.

[24] H. Branding, A. P. Buchmann, T. Kudrass, and J. Zimmermann, "Rules in an open system: the REACH Rule System," in *Proc. 1st Intl. Workshop on Rules in Database Systems*, Edinburgh, Scotland, September 1993, pp. 111 – 126.

[25] D. B. Kaber and M. R. Endsley, "The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task," *Theoretical Issues in Ergonomics Science*, vol. 5, no. 2, pp. 113 – 153, 2004.

[26] "Team Hector Darmstadt website," January 2012. [Online]. Available: http://www.gkmm.tu-darmstadt.de/rescue/

[27] J. Meyer, P. Schnitzspan, S. Kohlbrecher, K. Petersen, O. Schwahn, M. Andriluka, U. Klingauf, S. Roth, B. Schiele, and O. von Stryk, "A semantic world model for urban search and rescue based on heterogeneous sensors," in *RoboCup 2010: Robot Soccer World Cup XIV*, 2010.

[28] "Team Darmstadt Dribblers website," January 2012. [Online]. Available: http://www.dribblers.de/