

Persistent Expectation Management in Human-Robot Teaming (WP5)

Kruijff-Korabayova, Ivana; Racioppa, Stefania; Saad, Elie; Hindriks, Koen V.; Mioch, Tina; Vught, Willeke van; Consortium, the TRADR

Publication date

2018

Document Version

Final published version

Citation (APA)

Kruijff-Korabayova, I., Racioppa, S., Saad, E., Hindriks, K. V., Mioch, T., Vught, W. V., & Consortium, T. TRADR. (2018). *Persistent Expectation Management in Human-Robot Teaming (WP5)*. (pp. 1-40). FP7 - European Commission. <http://www.tradr-project.eu/wp-content/uploads/dr.5.4.main-public.pdf>

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.



DR 5.4: Deliverable D5.4

Ivana Kruijff-Korbayová[‡], Stefania Racioppa[‡], Elie Saad[‡], Koen V. Hindriks[‡], Tina Mioch^{*}, Willeke van Vught^{*}, and the TRADR consortium

[‡]DFKI, Saarbrücken, Germany

[†]Interactive Intelligence Group, Delft University of Technology, the Netherlands

^{*}TNO, Soesterberg, The Netherlands

`<ivana.kruijff@dfki.de>`

<i>Project, project Id:</i>	EU FP7 TRADR / ICT-60963
<i>Project start date:</i>	Nov 1 2014 (50 months)
<i>Due date of deliverable:</i>	Month 50
<i>Actual submission date:</i>	March 15, 2018
<i>Lead partner:</i>	DFKI
<i>Revision:</i>	final
<i>Dissemination level:</i>	PU

We report Year 4 progress in the TRADR project WP5: *Persistent models for humanrobot teaming*. We focused on the analysis, modelling and online-processing of the information-gathering tasks that the human-robot team is performing during a mission, with the goal to enable the robotic system to follow the mission (understand which tasks have been assigned to whom, what the progress is) and provide support for the management of the activities through the agent system and based on the working agreements. The reported work includes further development of team communication processing, ontology modelling, task management support, working agreements. The developed modules are integrated in the TRADR system and were evaluated during the TRADR evaluation exercise.

1	Tasks, objectives, results	6
1.1	Planned work	6
1.2	Addressing reviewers' comments	6
1.3	Actual work performed	7
1.3.1	Team Communication Processing	8
1.3.2	Using Online Resources to Extend Coverage	9
1.3.3	Working Agreements	10
1.3.4	Predictability and Transparency of Task Execution in Human-Agent-Robot Teams	12
1.3.5	Taxonomy of tasks for progress monitoring	15
1.3.6	Ontology Design for Task Allocation and Management	17
1.3.7	Agent Support for Situation Awareness and Task Management	20
1.3.8	Task Priority Decision Support	23
1.3.9	Learning Task Types from Previous System Interactions	24
1.4	Relation to the state-of-the-art	25
2	Annexes	29
2.1	Mioch, T., Peeters, M., and Neerincx, M.A. "Improving Human-Robot Co-operation through Working Agreements"	30
2.2	van Vught, W. "Modeling the predictability of task execution in a Human-Agent-Robot team"	30
2.3	van Vught, W "Taxonomy of tasks"	31
2.4	Ontology Design for Task Allocation and Management in Urban Search and Rescue Missions	32
2.5	Rozemuller, C., Hindriks, K.V., Neerincx, M.A., "Task priority decision support for effective search and rescue missions"	32
2.6	Mioch, T., "CTL analysis of team leader"	33
2.7	On the Effects of Team Size and Communication Load on the Performance in Exploration Games	33
2.8	Racioppa, S., Willms, C., "Team communication processing in TRADR"	34
2.9	Anikina, T., "Measures of Semantic Similarity for Dictionary Extension and Intent Recognition"	35
A	On the Effects of Team Size and Communication Load on the Performance in Exploration Games	41
B	Ontology Design for Task Allocation and Management in Urban Search and Rescue Missions	51

Executive Summary

This report presents the progress achieved in Year 4 of the TRADR project in WP5: *Persistent models for humanrobot teaming*, addressing Task 5.4: *Persistent Expectation Management in Human-Robot Teaming* leading to Milestone MS5.4.

The focus of our work in Year 4 was on the analysis, modelling and online-processing of the information-gathering tasks that the human-robot team is performing during a mission, with the goal to enable the robotic system to follow the mission (understand which tasks have been assigned to whom, what the progress is) and provide support for the management of the activities through the agent system and based on the working agreements.

We analyzed the TRADR verbal team communication data collected in previous TRADR experiments and on this basis we determined a set of intents and concepts to be used for the models for speech recognition and understanding and we developed these models for the Nuance Mix.NLU tool for German and for English. The communication data exhibited contextual reference phenomena, which means that the intents are uninterpretable without taking context into account. We annotated and analyzed contextual reference in the data, and developed contextual reference resolution for the frequent case of underspecified actors. Finally, we have developed experimental tools based on semantic similarity using online resources for identifying lexically similar expressions and for intent recognition (alternative to Nuance Mix.NLU).

We restructured and extended the TRADR ontology with four (actor, communication, environment, and mission) modules to support task management, including support for task allocation and progress monitoring, task priority setting, as well as the predictability and transparency of team members. This extension thus provides for additional situation awareness and expectation management as the system is now able to update the team leader about actor status and task progress. We also developed a model for task priority support and integrated this into the TRADR system. A task management interface has been developed that provides a team leader with a current and up-to-date view on tasks. The interface extends the TDS and allows a team leader to edit and add tasks. This work is related to WP3. Functionality has been developed to allow the team leader to add and extend task types which persist over multiple sorties and are used by the system to learn which tasks to propose to a team leader when a new POI is added to the system at a later point in time. Automated agent support has been further developed to use the TRADR task management ontology to provide automated task allocation support for the team leader. The agents provide support to the team leader by automatically proposing task assignments and enabling task update requests via speech as well as to provide automated support using working agreements.

The working agreements model has been further developed and integrated in the TRADR system. To provide the user with more transparency of the behavior of the system, the working agreements are presented to the user and can be adapted and personalized by the user him/herself. Besides that we improved transparency by giving the user insight in the available team members. Furthermore, we developed a task model and integrated this into the TRADR system. Based on ontological reasoning, the agents can predict elements about task execution (e.g. success or failure). Using these models, task progress is monitored based on the behavior and intents of the robot and its operator.

The TRADR Joint Exercise in June 2017 at the Firebrigade School in Rotterdam and the TRADR End-User Evaluation in November 2017 at the Deltalinqs training plant in Rotterdam provided excellent opportunities for collecting data for the evaluation of the above tools and their further improvements.

Role of Human-Robot Teaming in TRADR

WP5 deals with the issue of how a human-robot team can operate, and grow over time through its experience of working together. Approaching this from the viewpoint of the robot as well as from a human perspective, WP5 aims at developing models and algorithms for determining and recognizing human as well as robot behaviour at the (social) team-level. This encompasses the analysis and modeling of team-level communication and coordination, reasoning with role-based social behaviour at a team level, learning how to adapt that reasoning to better anticipate social behaviour, and learning how to adapt (pre-defined) strategies for team-level interaction.

Contribution to TRADR scenarios and prototypes

Issues of human-robot teaming are of central importance in the scenario chosen for TRADR, namely the response to an industrial accident consisting of multiple sorties over an extended period. The Year 4 use cases (cf. DR 7.4 of WP7) further extend those of Year 3. They involve several teams consisting of a team leader, two UGV operators and UGVs and an UAV-operator with a (piloted) UAV in multiple sorties in a larger and dynamic environment. The Year 4 use cases again include both (simultaneous) operation of individual robots and multi-robot collaboration. The teams are performing an initial assessment of an accident site, followed by subsequent information gathering sorties. The use cases provide an abundance of opportunities for teamwork. Control as well as task and resource allocation are challenging in the larger teams. An important issue with respect to team changes and multiple sorties is how information gathered by one team in one sortie can

be transferred and used by new teams in other, later sorties. The work carried out in WP5 Year 4 improved the understanding of the issues involved in these challenges and developed supportive tools and methodologies to address them.

Persistence

Persistence in WP5 is addressed by representing events from on-going sorties in persistent databases and using this information in various ways. The most direct use of the stored information is for creating interactive reports that allow users anytime and anywhere to get an overview of the progress of operations to survey their success and to provide decision support in preparing next steps and future sorties. The provided tools help users to establish common ground as shared information state about the mission.

The work on working agreements addresses the issues of how a human-robot team can grow over time through experience of working with each other. This is achieved by adapting policies either automatically or by explicit feedback.

The TRADR ontology has been continuously extended to allow the system to keep track of tasks and events throughout a mission. Additional support was created by extending the ontology for the team leader's activity of managing and assigning tasks. The team leader can now add new task types that will persist throughout a mission and which will be automatically proposed by the agents in case an associated POI is detected. The agents thus provide the team leader with automated support by proposing relevant tasks based on the persistent knowledge that is learnt over time during multiple sorties.

Adding tasks and new types of objects on the fly is a crucial ability for disaster response missions, which typically tackle unexpected and novel circumstances. This poses a challenge for the verbal team communication processing, which needs to be able to handle references to objects and tasks that have not been foreseen during the development of the language processing resources, in particular the language understanding models. To tackle this issue we have developed experimental tools based on semantic similarity using online resources for identifying lexically similar expressions and for intent recognition with the vision to facilitate coverage extension within and across sorties/missions.

1 Tasks, objectives, results

1.1 Planned work

The plan for Year 4 had foreseen WP5 to address *Persistent Expectation Management in Human-Robot Teaming* (Milestone MS5.4). The goal was to develop an account of how expectations for team-level collaboration between multiple humans and robots develop, adjust and align over time.

1.2 Addressing reviewers' comments

1. *Comment:* Activities behind schedule, but showing promise include:
 - development of tools for monitoring team activities and providing reports to build common ground
 - design of working agreements that enable robots to participate as team-members
 - Integration of the agent environment in the TRADR core enabling the agents to support situation awareness

Response: The partners have focused their efforts in Y4 on further development of these aspects.

2. *Comment:* Activities completely disconnected include the study of abstract aspects of team communication

Response: Research on these themes has been discontinued in Y4. One paper about the work done in Y3 was only recently accepted for publication, and is therefore listed in the Annexes in Section 2, because it took longer than we had expected.

3. *Comment:* It is strongly recommended to forcefully implement the contingency plan presented by the partners after the review meeting.

Response: Development according to this Contingency Plan was carried out and the resulting integrated modules were tested with users of the TRADR system in June 2017 (ITEX2, Delft and T-JEx, Rotterdam). A report was provided to the reviewers in November 2017.

4. *Comment:* For the remaining work of WP5 and the implementation of the contingency plan it must be considered that the role of the TRADR unit in an overall scenario with human response teams has changed during Y3 based on findings from the joint exercises and discussions with first responders. The TRADR unit is now a more separated unit which supports the human response teams by information gathering. This appears as a reasonable role in short- and mid-term applications. However, also mid- and long-term roles should also be foreseen in such a research project.

Response: The design of the TRADR unit is not restricted to short-term goals. Reconnaissance is a crucial task in a disaster response mission. A response always starts with situation assessment and search for persons and/or hazard sources. The use of robots in this initial stage can increase operational safety as well as expand operational capability. The TRADR use cases are designed with these needs in mind, focusing on information gathering and sharing situation awareness. The TRADR unit can function as a more separated unit (as in the Amatrice deployment), as well as in close cooperation with other parts of the (human) response forces (as in the Mirandola deployment). The TRADR exercises did not include additional response units, on the one hand, for practical/logistics reasons, and on the other hand, because of the focus of these exercises on the integration of the components of the robotic system and the interaction with the response forces within the immediate operational envelope. When going operational the TRADR unit would be a natural part of the response next to other response units. We believe that the TRADR unit is able to support additional response units and their information needs. We envisage that role to continue to be important on the mid- and long-term. To optimally benefit from the TRADR unit in a larger setting there is still work to be done, which was beyond the scope of TRADR to more tightly integrate operation into procedures and systems used by other response units.

1.3 Actual work performed

In Year 4 we focused our efforts in WP5 on the analysis, modelling and online-processing of the information-gathering tasks that the human-robot team is performing during a mission, with the goal to enable the robotic system to follow the mission (understand which tasks have been assigned to whom, what the progress is) and provide support for the management of the activities through the agent system and based on the working agreements. The following work was performed:

- team communication analysis and processing for monitoring team activities (Section 1.3.1)
- using online resources to extend team communication processing coverage (Section 1.3.2)
- modelling predictability and transparency of task execution (Section 1.3.4)
- design of working agreements enabling robots to participate as team-members in a search task (Section 1.3.3)

- development of a task taxonomy to support process monitoring (Section 1.3.5)
- design of an ontology for task allocation and management (Section 1.3.6)
- development of agent support for situation awareness and task management (Section 1.3.7)
- design of task priority decision support for team leader (Section 1.3.8)
- learning task types from system interactions (Section 1.3.9)

Below we provide a summary on these subtasks. Section 2 contains abstracts of the papers and reports where this work is presented in more detail and which constitute the annexes of this report.

1.3.1 Team Communication Processing

The ability of the robotic system to understand the verbal communication among the human team members has various benefits: (i) it enables the generation of better structured and annotated mission reports; (ii) it provides more complete mission execution information and thus enables more accurate automatic mission management support; (iii) ultimately it makes it possible for the humans to focus on their tasks and takes away the burden of entering mission management information manually.

We continued the analysis of the human-human communication in the team that we started in Year 3. We designed a set of intents and annotated the available data accordingly, through an iterative process. We then developed a training corpus and trained a model for processing German using the cloud-based speech engine Nuance Mix.nlu. In order to develop a model for processing English we translated the German corpus into English, because we did not have any English mission data. For the automatic processing we focused in particular on the *assignment of tasks by the team leader* to the operators and *task execution progress reports by the operators*. These intents served as input for the task manager and the working agreements (see below).

Speakers in natural conversation sometimes use "abbreviated" expressions/utterances, the interpretation of which depends on the preceding dialogue context. This is known as *contextual reference*. We annotated and analyzed occurrences of contextual reference in the original German data. We found that three levels of underspecification prevail: underspecified task (e.g., "Yes" vs. "Yes I will explore the area"); underspecified concept (e.g., "Send me a picture" vs. "Send me a picture of the barrel"); underspecified actor (e.g., "Explore the area" vs. "Operator one, explore the area"). We have so far implemented automatic resolution of actors. We have developed a concept for the resolution of the other cases, which involves keeping a

record of active tasks and previously mentioned entities, and a lookup algorithm based on recency of mention within the given team leader-operator communication thread.

Team communication processing has been integrated within the TRADR system and employed during T-Eval 2017. The results of using speech processing during the T-Eval 2017 missions were, however, disappointing. The intent recognition models lacked robustness and coverage. This has various causes: only very small amount of data was available from previous experiments; the tasks and mission execution has changed between experiments and therefore also the communication changed; the users did not adhere to the proposed communication protocol; they were not fluent English speakers and were improvising. Last but not least, the intent modelling required for the team communication understanding is pushing the Nuance Mix.nlu tools beyond the limits of what they can currently support. Team communication processing is described in detail in [40] (Annex Overview 2.8).

1.3.2 Using Online Resources to Extend Coverage

Disaster response missions tackle unexpected and sometimes novel circumstances. This means that the verbal team communication is likely to contain references to objects or even tasks that have not been foreseen during the development of the language processing resources, in particular the models described in the previous section. This is known as the "out-of-coverage" problem. As a result, the system is not able to assign an intent to an utterance. One way to address this problem is to try "guessing" what intent an utterance may be expressing or what objects are being referred to based on similarity to an intent/concept that is included in the model.

We have researched methods of measuring semantic similarity using wide-coverage online resources. We have developed experimental tools for (i) lexical similarity; (ii) intent recognition (alternative to Nuance Mix.NLU). The lexical similarity tool proposes synonyms and near-synonyms and can thus help "guess" concepts during processing. It is however applicable for lexicon extension also directly in the process of developing the language processing resources: the developer can include suitable proposals in the lexicon. The alternative intent recognition models use similarity to determine the nearest matching intent covered by the model. We have carried out preliminary experiments using the available TRADR data. The results were promising, outperforming the Nuance Mix.nlu models when the tests included challenging utterances. We however did not have enough true out-of-coverage data. Details about the similarity-based approach are provided in [2] (Annex Overview 2.9).

We have experimented with these methods with the view to contribute to the persistence of the language processing system across missions. The underlying idea is that when lexical similarity and the similarity-based intent

recognition result in correct intent recognition, this can be used to extend the models. A manual intervention to validate the model extension would likely still be needed, in order to maintain high accuracy of the intent recognition models. This could be done between sessions to gradually improve the system.

1.3.3 Working Agreements

One of the objectives within WP5 is to study how a human-robot team can operate, and grow over time through experience of working with each other. One way to ensure that robots act as expected is to set working agreements (WAs), beforehand and during a mission.

This year, we have specified a framework for working agreements, and implemented and integrated it into the TRADR system. The goal of the WA framework is twofold; (1) the ontology should support the generic specification of WAs, i.e., it should be possible to specify different kinds of working agreements and (2) the ontology should support reasoning about and adapting the WAs in real-time, i.e., it should be possible to determine in real-time, which WAs are accepted (by which actors) and active at any given moment, and for the different actors to change the WAs in run-time.

Following Singh [47], we identified several core factors that need to be specified when detailing a working agreement (see also Figure 1). The *creditor* is the agent that receives the WA. For each WA, at least one creditor needs to be specified. The *debtor* is the agent that is committed towards executing the WA. For each WA, at least one debtor needs to be specified. There may be several creditors and debtors, with the creditors and the debtors being disjoint (no creditor can be debtor of the same WA at a particular time, and vice versa). We thus can differentiate between two kinds of levels of agreements: an agreement between two actors (one-on-one), the individual level, and an agreement between more than two actors (either being several debtors, several creditors, or both), the organizational level. The *antecedents* are the conditions that need to be the case to make a WA *active*. An *active* WA implies that the consequent should be executed or added as a goal by the debtor. The *consequent* is a task and can have a lifespan. This lifespan determines at which point the WA is not applicable anymore. If there is no lifespan to the consequent, the working agreement is generally applicable. For a general overview of the WA framework, please see 1.

Please note the difference between a WA being *active* and *accepted*. A WA is *accepted* if both the creditor(s) and the debtor(s) have accepted the WA, which means that they are both committed to this WA. The acceptance of a WA is part of the WA framework. A WA being *active* means that an accepted WA is actually triggered, i.e., the antecedent is true. The latter is not part of the WA framework, but is evaluated by the agent or program

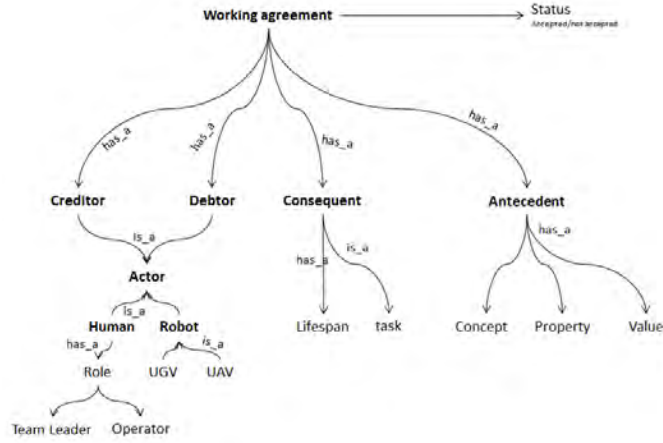


Figure 1: Working agreement framework.

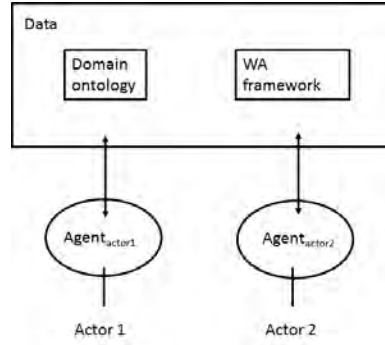


Figure 2

that executes the debtor's commitment.

The WA framework, as being a generic framework, is initialized by specific WAs, using the TRADR domain ontology (see also Section ??). TRADR agents reason about the activeness of the WAs, and execute the WAs if active, see the general overview in Figure 2. WAs should adequately take care of the responsibilities, capabilities and states of the humans and robots. Based on previous scenario analyses and exercises, the WAs that have been implemented for TEval 2017 were WAs regarding communication and task allocation. Cognitive task load [35] is an important state of (human) team members, and was taken into account for the timing of the notifications by the robot. The CTL was (objectively) observed by a human observer and inserted into the database in real-time (see also [31] (Annex Overview 2.6)).

The general WA framework was integrated into the TRADR system and has been evaluated in the TRADR Joint Exercise and the year 4 evaluation. We evaluated the validity and applicability of the WA framework and used

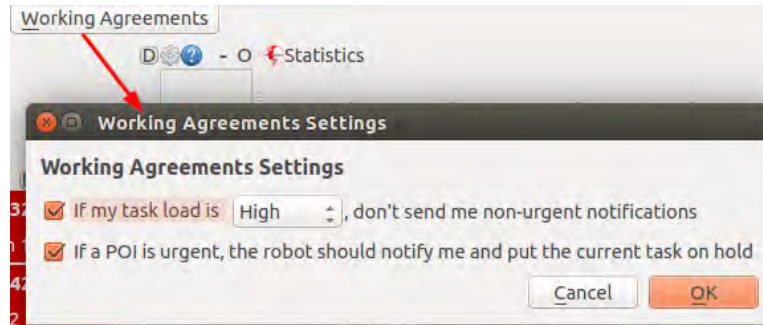


Figure 3: Dialog in TDS to accept and change working agreements.

an iterative design, evaluated particular WAs and, based on the results, improved these WAs. For a more detailed specification of the framework and the evaluation results, please see [32] (Annex Overview 2.1).

In addition, the actors can accept and adapt the WAs in run-time in TDS, making it possible to personalize the WAs and adapt them to the current mission, tasks, and needs of the actors. For a screenshot of the settings dialog to accept or change a WA, please see Figure 3.

During discussions with the fire fighter before and after the actual mission, fire fighters agreed that in general, it is important that there is (an explicit) agreement about the manner of communication and task allocation between the different team members, including robotic team members. The fire fighters mentioned that these agreements depend on several aspects, such as the specific mission the current task, and personal aspects. In addition, it became clear that there are individual differences between the team leaders on how the agreements should be set; the specific WAs can depend on the individual team leader, confirming the need for adaptive working agreements.

The specific WAs that have been implemented and tested during the scenario were evaluated and improved based on the feedback of the fire fighters and observations during the exercise.

1.3.4 Predictability and Transparency of Task Execution in Human-Agent-Robot Teams

One of the challenges that humans, agents and robots as teammates bring is optimizing the observability, predictability and directability to improve effective teamwork. This year, we created a model to make predictions about task execution by a human agent robot team that operates in a dynamic environment to improve the situation awareness for better task execution.

A model is generated that specifies a human-agent robot team that operates in a dynamic environment that can be used to reason about these real world facts and to send predictive notification about task execution

to the actor taking his capabilities, the environmental factors and planned tasks into account. The generated top level model is visualized in Figure 4. Knowledge that is made explicit in this model, is that the actor enacts a role within an environment and has capabilities to do that. To enact a role and perform a task, certain capabilities may be required. For example, if a robot has to make a map of the environment, it needs to have the capabilities to do that, e.g., it needs to have a laser scanner. However, environmental events or objects can disable these capabilities. For example, if there is a lot of smoke in the building, the laser scanner might not work.

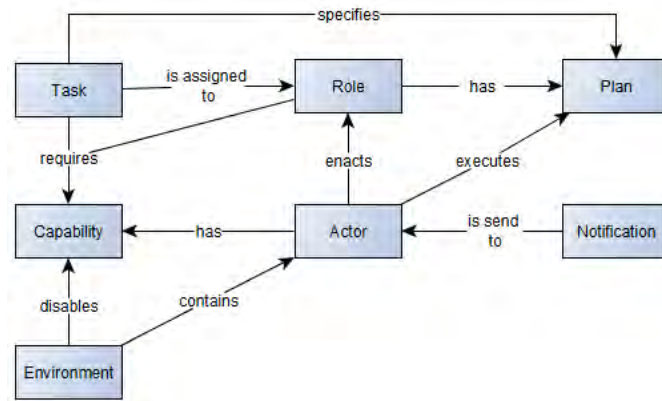


Figure 4: Main concepts and their relations to make predictions about task execution

Scenarios and detailed use cases relevant for the TRADR domain and other domains, were used as an inspiration for the design of the model and the predictive rules. Requirements and claims were set to guide the development which resulted in a detailed model and rules that can predict:

1. Obstruction on route - An environmental object disables a capability of the robot that is required for the active task.
2. Mismatch in capability UAV - During a task, a functionality of the UAV fails which is required to perform the active task.
3. Disabled capability UGV - An environmental event at the location of the objective of a planned task disables a capability that is required for the task.
4. Too low battery capacity - The status of the robot leads to the prediction that a task cannot be successfully executed or completed.

The model and most prediction rules were implemented and evaluated in the TRADR Joint Exercise (TJEX) with end users. This required additional entities and relationships within the ontology and new agent rules

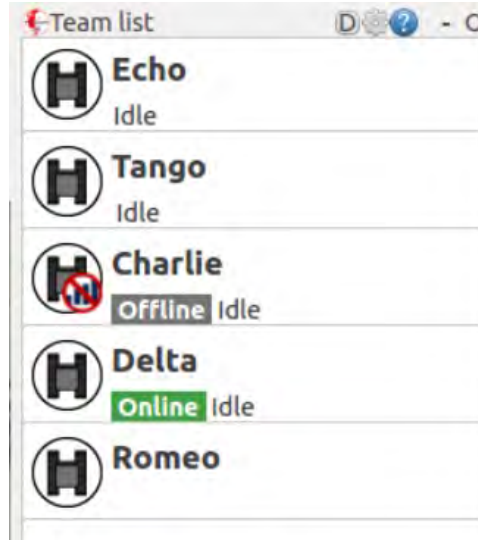


Figure 5: Team list in TDS with robot status

to reason, make predictions and notify the end users about the prediction. The predictions could be used by the end users to improve effective task execution.

In the evaluation, the (virtual) agents successfully predicted mismatches in task execution based on the state and capabilities of the actor, knowledge about active and planned tasks and environmental restrictions. The participants indicated that they found the notifications about the mismatches useful in USAR missions. Based on the evaluation, we can conclude that the proposed model is able to make predictions about task execution and increase the situation awareness for better task execution.

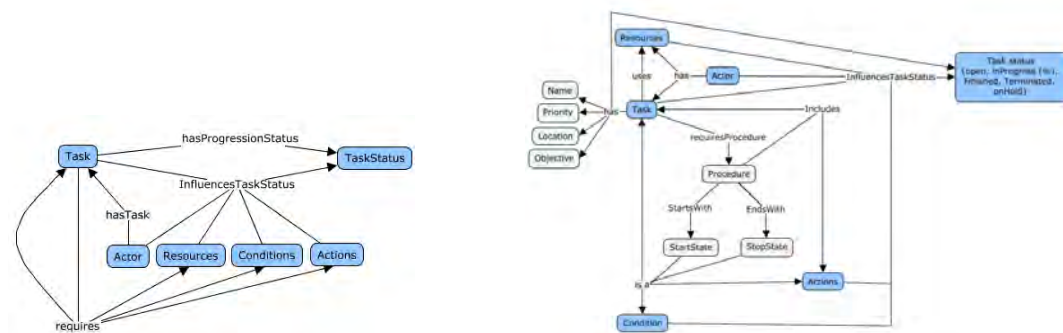
Transparency about the availability of (robotic) team members was also improved. Transparency about the availability of the team members can help to increase the situation awareness and improve task execution of a human-agent-robot team [23]. The TRADR system allows actors to change team members during a mission, and so, change the team composition. However, end users indicated that the TDS did not differentiate between robots that were actively participating in the team and robots that were non-functional (e.g. lost connection). They suggested that transparency about the availability of robotic actors should be increased. To make this aspect of the team composition more transparent, we developed new functionality for the TDS to be able to indicate - besides the status of the robot - if a robot is online or offline. An example of a team list overview is shown in Figure 5.

1.3.5 Taxonomy of tasks for progress monitoring

We developed a taxonomy of tasks which is used for detecting active tasks and monitoring the task progress and integrated this taxonomy in the TRADR ontology. This taxonomy allows the system to provide team members with an overview that contributes to team alignment, team situation awareness, and lower cognitive workload as the system now can perform low level task monitoring itself. The taxonomy also is used to make more accurate predictions about future events, and thus supports expectation management.

Based on models such as those described in the work of [27, 48] and as discussed in [50] (Annex Overview 2.2), a task model has been created that supports monitoring the progression of tasks. Figure 6a) shows the top level taxonomy and 6b) shows the more detailed taxonomy of tasks. The proposed model describes tasks executed by an actor that plays a role in the team. Each task is operationalized through procedures which can be viewed as the plan to execute a task successfully; these are also used to monitor the task status. Besides the progress within a procedure, tasks, resources and actors can influence the status of a task. An extensive description of the model is provided in [51] (Annex Overview 2.3).

This model is integrated in the TRADR system during TEval. Data from the high level database was used to monitor the status of a task, namely; speech data, location data, taken media and measurements (e.g. smoke and gas). The task status was updated by the agents in the task manager. All implemented start, stop and intermediate states are visualized in Figure 7. Unfortunately, the start and stop states of the tasks were not triggered during TEval and all task statuses had to be updated by the team members themselves. However, based on the gathered data, we can predict how the



a) Top level task taxonomy to monitor the progress of tasks. b) Detailed task taxonomy to monitor the progress of tasks.

Figure 6: Top level and extensive task model to monitor the progression of tasks

model would have behaved if the states were triggered during the mission.

To predict how the model would have behaved, we annotated speech data of the last two sorties during TEval and considered that as the ground truth of what tasks were executed and how they were updated during that mission. Besides that, we analyzed how much of these tasks and status switches were inserted during the mission via the task managers by the end users themselves, and what the model would have predicted based on the speech and sensor data. For the extensive analysis see [51] (Annex Overview 2.3).

From the first analyzed sortie (sortie 9), we saw that the actors only inserted 23% of the executed tasks and status updates by hand during the mission. The model would probably have predicted 73% of the tasks and task updates. For the second sortie, 56% of the tasks were updated correctly by hand were the model would probably predict 88% correctly. From the interviews with the end users, we found that the end users value an accurate task list, however, in the current way of working they had to switch their focus to update the tasks which was too demanding. Furthermore, because we worked with a small team and the fire department is only assigning tasks one at a time, they did not feel the need to insert all tasks in the system. They commented that if the system could insert and update the tasks itself, it would save them time and lower their workload on that aspect so they can focus better on aspects with a higher priority. However, if the system is not 100% correct, it could lead to confusions which can be crucial in these kind of situations.

Based on these findings, we can conclude that an accurate task overview is indeed of great value to the end users and contributes to the situation awareness of the actors. Furthermore, the task model would contribute to a lower cognitive task load and gives a much more accurate predictions about

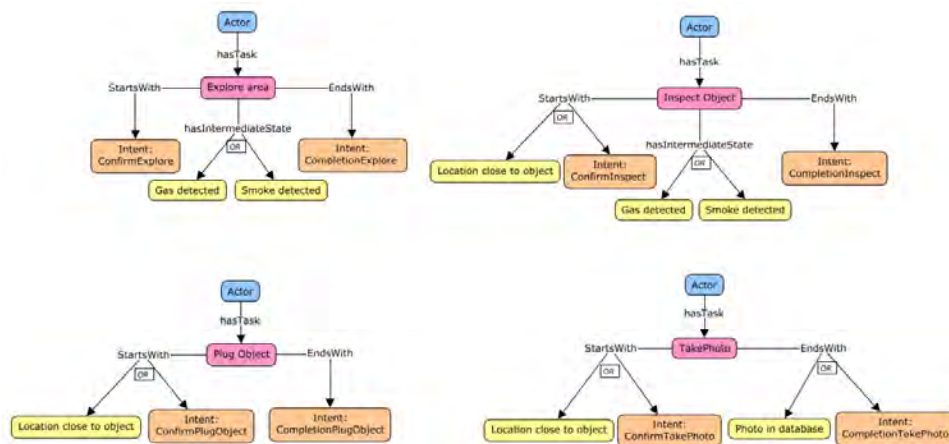


Figure 7: Integrated task triggers during TEval

current tasks than that team members update this list manually.

1.3.6 Ontology Design for Task Allocation and Management

We structured and redesigned the main ontology and extended it to cover the requirements for task management in USAR missions as described in [43] (Annex Overview 2.4). The aim has been to make the ontology (1) flexible and extensible, to be able to easily modify or add new components, for e.g. covering more use cases; (2) reusable, so it can be applied in different types of missions; and, perhaps most importantly, (3) readily understandable by firemen, the key rescuers in our domain, in order to facilitate task allocation and management. We briefly discuss the design approach we used for designing and modelling the ontology as well as the key concepts that are relevant for automating task management support.

Ontology Design Approach

When designing our ontology, we used a design approach adapted from [46]. This iterative process consists of different phases, as illustrated in Figure 8. First, we analysed the requirements in the search and rescue domain by reviewing the literature and by interviewing fire-fighters. Second, we conceptualized the required domain entities and their relationships based on common vocabulary used by firemen. Third, we implemented the ontology as RDF triples in the OWL 2 Web Ontology Language syntax¹ and visualized it through the system's user interfaces. Fourth, we evaluated the ontology with firemen using search and rescue use case scenarios. And finally, we refined the ontology by extending it with new concepts and entities needed for addressing the requirements of additional use cases.

Ontology Modeling

Our ontology is aimed at providing a common vocabulary which is useful for task management by facilitating information sharing and communication between team members. The concepts represented in our ontology therefore include the relevant entities and information categories that are needed for task allocation and management during USAR sorties. We summarized and grouped these entities into multiple modules in order to easily append additional components as we extend the core ontology. Four of these modules are relevant and part of the task management ontology. In the following, we briefly discuss the key concepts that have been included in the different modules and their relationships.

¹<https://www.w3.org/TR/owl2-syntax/>

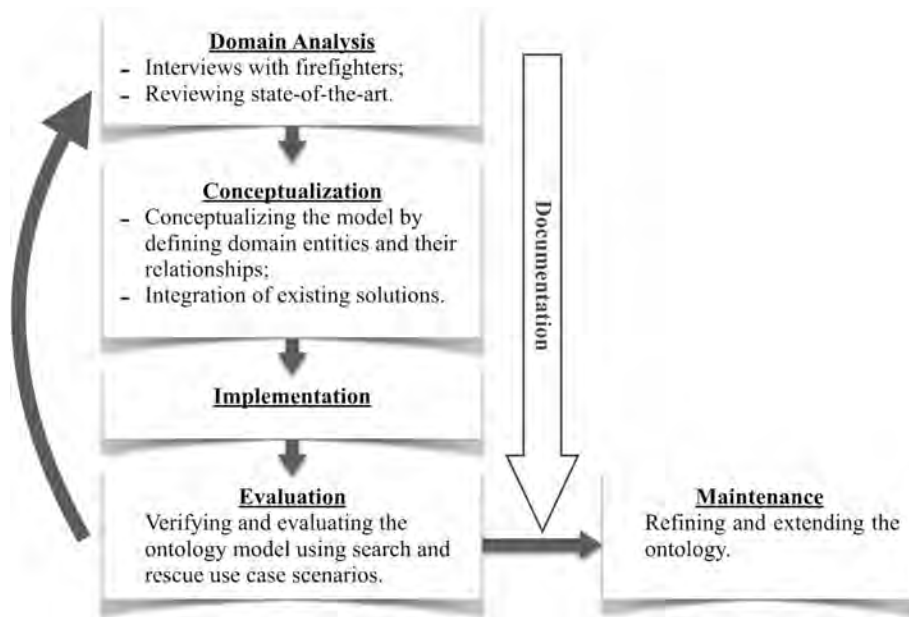


Figure 8: Ontology design approach

Actor Module The actor ontology represents the human and robot actors along with their properties. The actors are resources used for responding to the disaster. By representing the roles, status, capabilities, and related concepts in the ontology, the latter can provide a basis for automated support for task management. The system, for example, can reason which actors might be suitable for performing a specific task.

- **Actor Roles and Teams** - During an USAR mission, human and robot actors collaborate for executing the tasks assigned by the team leader. To know who will do what, human actors have different roles (UGV operator, infield rescuer, etc.) as do robot actors (e.g., ground or aerial explorers). We have based the model of the role concept on a TRADR human-robot team which we believe is sufficiently general to apply more generally. It has moreover been validated by firemen from three different countries that participate in the project (Italy, Germany, and The Netherlands). Teams are composed of a team leader, UGV and UAV tele-operated robots and their corresponding operators, infield rescuers, robot mechanics and safety officers.
- **Actor properties** - In addition to roles, each actor has a status (idle, on the move, etc.). Human actors have a workload, which is an indicator of an actor's task load during a mission. And robot actors have battery readings, which indicate the battery percentage at each moment in

time. Such properties help the team leader when assigning tasks by showing which actor is available for performing a given task.

- **Actor Capabilities** - Actors have capabilities related to their skills (paramedic, etc.) and the devices they are equipped with (infrared camera, gas detector, etc.). Knowing the capabilities of actors helps in providing automated support and suggesting available actors to the team leader when assigning a task.

Communication Module In an USAR mission, gathering and sharing relevant information is important for improving situation awareness and facilitating task management. This requires a communication network between actors.

- **Communication devices** - In this category we model the devices needed for gathering and communicating data between members, such as electronic and sensing devices (infra red sensors, thermo cameras, network devices, etc.).
- **Communication events and media types** - Relevant information is shared using different types of communication events (notifications, messages, etc.) and media types (audio, photo, text, etc.). This is helpful for keeping the team leader and other members aware of the state of a mission and alert them when something unexpected occurs while executing a task.

Environment Module Environmental objects and events in a disaster area need to be inspected or handled by performing different tasks, and therefore are important to represent in a task management ontology.

- **Environmental events** - these include the events which have happened or can occur while scouting a disaster site and which need to be monitored and handled by rescuers such as explosions, fires, etc.
- **Environmental objects** - these include concepts for representing structures, barrels, etc., and that can be present in the disaster area.

Mission Module Allocating and managing tasks helps in planning and monitoring the progress of a mission. This requires an overall view of the situation which includes, among others, the location of active actors and of what was discovered so far along with the tasks assigned and their progress.

- **Tasks and relevant properties** - Throughout a mission, the team leader assigns tasks to available actors by specifying the task objective or POI, and providing a clear description. To monitor tasks and track

their progress, additional properties have been included such as status (in progress, completed, etc.) and priority. Moreover, each task has a list of required capabilities such as sensing, locomotion and communication, which defines to which actor(s) the task can be allocated.

- **Points of interest (POIs)** - This category includes the entities which can be detected while exploring a disaster site and are meaningful for improving situation awareness when assigning tasks. Each POI has a type (victim, fire, gas, hazards, etc.) and a location. These properties help in knowing which actors to send, where they should go and what might be the risks involved.

1.3.7 Agent Support for Situation Awareness and Task Management

An important task of the team leader during a mission is to allocate and manage tasks. In order to assist team leaders, improve their situation awareness, and provide them with automated task support, we implemented new functionalities in the system interface and its agents. These include (1) monitoring tasks using an interface; (2) proposing and updating tasks by the system agents; and (3) allowing task update requests via speech.

Monitoring Tasks To allow the team leader to monitor mission tasks, we created a task management system (Figure 9). This system is built on top and uses the common vocabulary as defined in the TRADR task ontology. The system interface used by a team leader allows him or her to select, add, edit, and monitor tasks. The latter are either manually created by the team leader or automatically proposed by the agents.

Whenever a POI (e.g., victim) is detected on the disaster site, the team leader's job is to allocate specific tasks (linked to the new POI) to team members. This is done using the task editor GUI, part of the task management system (Figure 10). For example, when defining a new area on the disaster site, the team leader assigns different tasks to team members to inspect that area depending on their role (i.e., robot operator) and capabilities (i.e., sensory capability).

Proposing and Updating Tasks by the Agents To assist the team leader in its job of creating tasks, automated task proposal is provided by the agents that are part of the TRADR system. Whenever a POI is detected, these agents propose new tasks using the TRADR task ontology, exploiting in particular the known relationships between POIs and task types. The agents do so by querying the HLDB. In summary, each task is supposed to have an id, a type (i.e., InspectionTask, MeasurementTask, etc.), a point

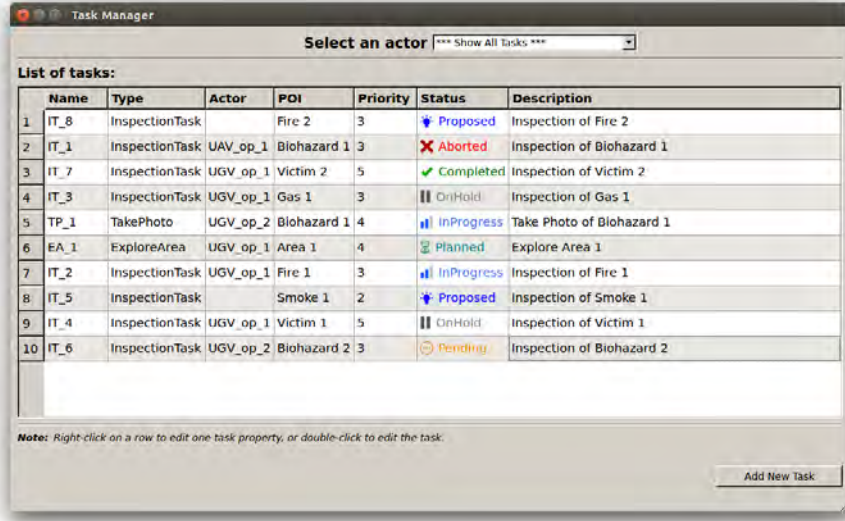


Figure 9: Task management interface for monitoring mission tasks.

of interest (i.e., Fire, Area, etc.), an actor, a priority and a status (i.e., Proposed, Pending, InProgress, etc.).

To illustrate task proposal, we include a simplified query for proposing an `ExploreArea` task after a new area is defined via the TDS:

```
INSERT DATA {
  tradr:NewTaskID a tradr:Task ;
  tradr:hasTaskType tradr:ExploreArea ;
  tradr:hasPOI tradr:POI_ID ;
  tradr:hasPriority <priority value> ;
  tradr:hasDescription <description string> ;
  tradr:hasStatus tradr:Proposed ;
  tradr:hasTimestamp <current time> .
}
```

Furthermore, the team leader can request the agents, via speech, to create and propose a new task. Beside that, team members can also request the agents, via speech, to update the progress of their allocated tasks. These requests are then sent to the HLDB using a specific ontological structure containing the task intent (i.e., `RequestExplore`, `CompletionInspect`, etc.), the actor description (i.e., `TeamLeader`) and a time stamp. When the agents detect a new request in the HLDB, they execute the appropriate action (i.e., set the task status to 'Completed' when a team member sends a request with intent 'CompletionExplore'). A sample query for a speech request is as follows:

The Task Editor GUI is divided into two main sections: 'Main Properties' and 'Default Properties'. The 'Main Properties' section contains several input fields and dropdown menus:

- Task Name:** A text input field with '(optional)' as a placeholder.
- Task Type:** A dropdown menu currently showing 'InspectionTask'. Below it is a text input field with the placeholder 'Or enter a new type (optional)'.
- Task Description:** A text input field with '(optional)' as a placeholder.
- POI Type *:** A dropdown menu showing '-- Select type --'.
- POI Name (if available):** A dropdown menu showing '-- Select POI --'.

Below these fields is a section titled 'Select one of the available actors *' which contains a table with the following data:

	Actor	Role	Status	Workload
1	Infield_rescuer_1	InfieldRescuer	Idle	1
2	Team_leader	TeamLeader	Leading	4
3	UGV_op_1	UGVOperator	Idle	3
4	UGV_op_2	UGVOperator	Idle	4
5	UAV_op_2	UAVOperator	Idle	1
6	UAV_op_1	UAVOperator	Idle	2

The 'Default Properties' section is currently empty. At the bottom right of the window are 'Save' and 'Cancel' buttons.

Figure 10: Task Editor GUI for adding new tasks or editing existing ones.

```

INSERT DATA {
  tradr:SR_ID a tradr:SpeechRequest ;
  tradr:hasTaskIntent <TaskIntent> ;
  tradr:hasActorDescription <ActorName> ;
  tradr:hasTimestamp <time> .
}

```

Situation Awareness To increase situation awareness when executing and managing tasks, the agents are supposed to generate notifications whenever a critical event occurs (i.e., battery level is low), as mentioned in Sections 1.3.3 and 1.3.5. In order to enable these features, we implemented the functionalities and ontological structures needed by the agents. These include the ones related to (1) tracking battery levels and detected POIs using the semantic modeler; (2) notifying an actor, via TDS, when they are assigned a new task; (3) saving task triggers in HLDB for measuring situation awareness; and (5) task logging.

Increasing Resiliency To address network resilience and prevent e.g. the GUIs from freezing when querying the HLDB (i.e., when the access

is slow), we implemented a multi-threading solution. The Task Manager interface and the task widget query the database to check for newly added or modified tasks. Each call to access the HLDB is now handled by a thread, so that whenever a thread is blocked or dies, the application will continue running. Note that TDS is using threading as well. As for the semantic modeler, its call-back functions for accessing the HLDB are handled by a ROS threaded spinning function. It is set to spin every second to update the HLDB based on the data received from the LLDB (i.e, pictures, POIs). Finally, the agents use a threading pool to ensure agents get access to a thread to ensure they will always be able to access data and make progress too.

1.3.8 Task Priority Decision Support

In a human-UGV search and rescue mission, the team leader should be able to quickly assess the priority of tasks and assign the most important ones to his robot-operators. However, even if the criteria for setting task priorities are clear, this remains a cognitive challenge. We designed and developed a decision support system that identifies tasks and assigns priority, in order to support a team leader in making more effective task assignments, see [42] (Annex Overview 2.5).

It is the role of the team leader to distribute tasks to the robot operators to get questions like these answered. In doing so, the team leader also needs to decide on the priority of tasks. We designed a decision support system that automates priority setting based on the intuitive priority model that fire-fighters use in practice. The system automatically proposes relevant tasks and uses all relevant factors that contribute to priority setting explicitly for sorting the tasks accordingly.

The content for our decision support model is based on interviews and exercises with firemen. We created a four factor model of priority that is applicable to the most common tasks in an industrial accident. The four factors, in order of importance, integrated in the task priority model are exploration (aimed at POI discovery), escalation (aimed at avoiding e.g. explosions), certainty (aimed at performing actions with most certain outcomes), and values (aimed at prioritizing in order of importance own safety, humans, animals, environment, economic factors). We have integrated this model into the TRADR system and performed an initial evaluation. At the time of evaluation, system efficiency issues related to the HLDB have prevented us from reaching any clear conclusions about the use of automated task priority setting support.

1.3.9 Learning Task Types from Previous System Interactions

In practice we found that team leaders want to add new task types to the system and link them to POIs. Therefore, in the current task management system, the team leader is able to add new tasks and specify their type either by manually entering the type or by selecting from a drop-down menu. Specifying the task type is found to be helpful for classifying tasks and connecting their types to specific points of interests (POIs).

For example, when a fire POI is detected, the team leader may want first to assign a task for inspecting the fire (to make sure there is no risk of explosion) before sending a team to extinguish it. In this fire example, two tasks are needed, an inspection task and an extinguish fire task. The TRADR ontology supports linking POIs to task types by means of the `hasRelatedTaskType` property:

```
tradr:POI tradr:hasRelatedTaskType some tradr:TaskType .
```

To assist the team leader in its job of assigning tasks, automated task assignment proposals are provided by agents. These agents query the TRADR ontology to propose tasks based on the known relationships between POIs and task types, as discussed in Section 1.3.7. By adding new task types and linking these to POIs, the task ontology and HLDB is continuously extended. For example, whenever the team leader creates a task with a new type and links it to a POI, the system notices the change and extends the ontology with both the new task type and its corresponding relation with the POI. Also, when the team leader links an existing task type with a POI, the link is stored. This enables the TRADR system to learn over time based on previous handling of POIs. This learnt knowledge can be used later by agents to propose the tasks the team leader preferred to assign for handling a given POI. The TRADR system agents, by using this new knowledge will, when similar POIs are detected on the disaster site, use the newly learned links from the ontology to propose these corresponding tasks.

Using the fire example we discussed above, when a team leader adds a new task type in this example, the ontology will be extended using two new relationships between `Fire` and the task types `InspectFire` and `ExtinguishFire`. To illustrate, a sample query that updates these relationships for the `InspectFire` task type is given next:

```
INSERT DATA {
  tradr:Fire rdfs:subClassOf [
    rdf:type owl:Restriction ;
    owl:onProperty tradr:hasRelatedTaskType ;
    owl:someValuesFrom tradr:InspectFire ] .
  tradr:InspectFire rdfs:subClassOf tradr:TaskType .
}
```

Upon detecting another fire, the agents will retrieve the new types from the ontology and propose two tasks, one for inspecting fire and one for extinguishing it. This means that now the team leader can use the task management system to assign these newly proposed tasks to corresponding team members. The agents do so by querying the HLDB.

In search and rescue, trust in the support systems used is essential. Since predictability is a critical aspect of trust, agent behaviours that change over time because of learning or new inputs may prevent operators from properly calibrating appropriate trust [6]. Because agent-human interactions are complex, it is important to avoid possible confusions that may arise from using the system. We believe that our approach that is based on the team leader proposing the changes to the system takes these factors into account, though evaluation in practice with fire-fighters of the approach is needed to confirm this.

1.4 Relation to the state-of-the-art

Team Communication Processing Speech interaction processing has been experiencing a boom recently. Speech recognition has reached levels of robustness and reliability that make speech-based applications usable in everyday life. Intent recognition is reliable in well-defined narrow-focus applications, such as conversational assistants in various domains, e.g., navigation in the car, travel/restaurant information, home-control. Also the processing of open-domain chats is becoming generally usable. Alexa and Siri are good widely known examples. These great advances are made possible by large amounts of annotated data that enable machine learning. No comparable resources exist for the search & rescue team communication domain, and this is the reason why we decided to start by creating a training corpus based on the data we have collected. There are various questions open for further research, for example, whether models trained on interactions in other domains could be transferred to our domain. The similarity-based intent recognition we have developed uses state-of-the art methods. But again, we have very little data.

Working agreements for human-robot teaming Much of the research on hybrid human robot teams looks at how the task balancing can be improved. However, the task aspects need to be extended by team work aspects, based on human-human teaming (e.g. [44]), but also on human-machine and human robot teaming (e.g. [36, 10]). Models of working agreements to support the team work aspects have been studied in research on normative systems, as social commitments for having robust and flexible multi-agent systems [47]. In the last years, the advantages of working agreements have been identified (e.g., [12, 15]). Working agreements seem especially appropriate for teaming, as team situations require communication

and collaboration [8, 30] in a way that is facilitated by (a) articulating tasks and functions, (b) properly allocating those functions among members, and (c) a display of the allocation of tasks among the team [14, 13]. [24] identified several design patterns that describe human-autonomy teaming in command and control of teams of autonomous systems, amongst which are working agreements. The design patterns are specified for a non-hierarchical relationship. Humans and autonomous agents team as partners (more or less). It provides the human user the means to indicate preferences concerning how task allocation (in this case) and execution should be accomplished.

Modelling the Predictability of Task Executing in Human-Agent-Robot Teams Research in HART mostly focus on making automation part of a team, instead of a tool that facilitates the human team [4]. To do that, the reactive actors (humans, agents and robots) should coordinate their actions, collaborate with other team mates, be transparent, be interdependent and the actors should be inter-predictable to reason about each other and make predictions about future events [20, 18]. Many frameworks and models exist that show a part of (human agent robot) teamwork [22, 5, 3, 52, 11]. However, most of this early work focuses on specific parts of teamwork and little attention is paid to a more general model of human agent robot teamwork. Therefore, we proposed a top level ontology that can be used to make predictions about task execution in HART and can be adapted for many domains, including the TRADR domain.

Task model for progress monitoring Many research focuses on task models, examples of known task frameworks are the Hierarchical Task Analysis, Groupware Task Analysis, Task Knowledge Structure, STEAM, and the ontology of van Welie, van der Veer and Eliëns (1998) [27, 48, 52]. For development of the task taxonomy, a combination of these existing models is used to configure the best model for task progression monitoring within the TRADR environment. Based on the ontology of van Welie, van der Veer and Eliëns (1998), basic concepts, properties and relations were specified for the task model. The TKS model [19] is used as an inspiration for the procedures. The STEAM model [48] is used as an inspiration for the influence of resources on the task status. Combining these features, we created a model that has a theoretical foundations and fits tasks within an unknown, dynamic environment, strongly hierarchical and fits the TRADR system to monitor the progression of tasks much better than with the currently way of working.

Ontology for Task Management Ontologies are widely used to represent domain knowledge and facilitate information sharing in many applications [41, 33]. According to [28], existing crisis oriented ontologies describe

concepts and characteristics related to a single subject area (type of disasters, geography, meteorology, etc.). Such ontologies are designed to address the requirements of a specific application.

In USAR robotics ontologies, work has been done on the cooperation between autonomous or semi-autonomous multi-robot teams [29]. For example, the robot ontology suggested by Schlenoff and Messina [45] is centered on representing the concepts about robots and their capabilities when assisting in USAR missions. This ontology is very relevant, but differs from our focus on providing task management support for a team leader.

Other robotics ontologies focus on the robot and its interaction with a specific environment. For example, [16] proposes an ontology for industrial use and [26] for underwater robots. The KnowRob ontology [49] offers a set of ontologies to model robots and their capabilities and actions. The Open-Robot [25] ontology (ORO), which shares many concepts with the KnowRob ontology, is focused on robot interaction with humans, but assumes that robots are completely autonomous.

Given the current state of the art in USAR robotics, we focus instead on remotely operated robots. The detailed design of our ontology has been based on our discussions and interviews with firefighters, experts in the field, and also has been inspired by Robin R. Murphy’s research on rescue robotics [34].

Task Priority Decision Support The SARPlan system [1] used optimization techniques to give recommendations for search areas based on geographical data and evidence collected so far. SARPlan is shown to be effective when searching for airplane crash survivors. A utility measure for automatic task allocation for UGVs and humans was proposed in [39]. Their utility takes into account task switching costs, capability, preference, and cognitive task load. Our model for task recommendations complements this work by focusing on the environmental factors that determine the priority of a task, instead of cognitive factors. Together the environmental and cognitive factors form a model for the sense of priority of a team leader. Under-utilization and over-trust are common issues of a partly automated system that can negatively impact performance [17]. Many of the firemen that we spoke to for this paper were concerned about what would happen if the advice given by the system would be wrong. It is therefore important to take the balance between automation and control of the user into account.

Learning Task Types from Previous System Interactions Although there is a lot of work on ontology learning in general (e.g., [9, 37]), in the context of search and rescue ontologies it is not yet feasible to fully automate the ontology development process and most learning techniques require some existing top-level ontology or seed concepts and require trading efficiency (by

automation) for higher quality (by using human input) [7].

In [38] agents ontologies also evolve as they encounter tasks that require additional information to complete. By reusing the additional information in their ontologies, they do not need to incur the cost of re-learning it in future. This work differs from our work by using the well-known rescue simulation environment which does not involve humans-in-the-loop (fire-fighters in our missions) and focuses on the size of ever-larger growing ontologies and the need to remove unused parts of an ontology again for real-time performance reasons.

In [21] a system called SHARE is proposed for operation rescue management that is based on an ontology and is able to process dynamic information that becomes available during a rescue operation. Speech technology is used to share dynamic information provided by fire-fighters while the rescue operation is evolving. The TRADR system is similarly able to share data collected on the fly by team members. Adding new task types, however, requires changing or refining the ontology and not just adding new information as well as changing the way agents use these refinements. Currently our system is based on manual entry to make such changes where the use of speech technology could provide a natural alternative that most likely would require a new dialogue for entering such changes.

2 Annexes

2.1 Mioch, T., Peeters, M., and Neerincx, M.A. “Improving Human-Robot Cooperation through Working Agreements”

Bibliography Mioch, T., Peeters, M. and Neerincx, M.A. Improving Human-Robot Cooperation through Working Agreements. To be submitted to IROS 2018.

Abstract A necessary factor for effective teamwork with team members is the ability to trust each other. An important aspect contributing to trust is predictability of a team member’s behavior. This is also the case in human-robot cooperation. However, for human team members, it often is difficult to understand the robots’ behavior, and to predict behaviors and know what to expect. Explicit working agreements (WAs) between human and robotic team members can tackle these challenges: in working agreements, the different team members agree on their expectations regarding task execution and communication. In this paper, we investigate (i) how and in what way WAs can be a solution for cooperation between humans and robots and (ii) the identification, structuring, and formalization of the working agreement concepts and interrelationships into a coherent model. The ontology and WAs are derived from theory and practice, developed in an iterative and incremental way, and implemented and evaluated in together with end-users during an exercise.

Relation to WP This reports describes the rationale and model for the working agreements developed in WP5.

Availability Restricted.

2.2 van Vught, W. “Modeling the predictability of task execution in a Human-Agent-Robot team”

Bibliography van Vught, W. Modeling the predictability of task execution in a Human-Agent-Robot team. Master thesis, VU Amsterdam, 2017.

Abstract In recent years, the view of humans, agents and robots as teammates has grown and recently became reality, which brings opportunities and challenges. One of those challenges is optimizing the observability, predictability and directability of teammates to improve effective teamwork. In this thesis, a high-level model is proposed to make predictions about task execution by a Human Agent Robot Team that operate in a dynamic environment to improve the situation awareness for better task execution. To bring structure to the design process, the situated cognitive engineering methodology is used. To build a foundation, a literature research is done

to gain knowledge about teamwork, task models and situation awareness. Based on this knowledge, use cases, requirements and claims are specified which are used to design the model. The designed model for predictions about task execution is described at two levels; one high level describing only the most important concepts and their relation, and for each concept a lower level description is provided about the required knowledge for accurate predictions about task execution. The meaning, relations and dynamics of these facts are discussed in detail. Furthermore, prediction rules in the form of system rules and working agreements are proposed that enables virtual agents to make predictions based on the knowledge in the model. The model and the prediction rules are implemented and evaluated in the TRADR project that focuses on gaining more situation awareness in urban search and rescue missions by the use of a specialized human agent robot team. In this evaluation, the virtual agents successfully predicted mismatches in task execution based on the state and capabilities of the actor, knowledge about active and planned tasks and environmental restrictions. The participants indicated that they found the agents useful in USAR missions. Based on these evaluations, we can conclude that we proposed a model that can make predictions about task execution and increase the situation awareness for better task execution.

Relation to WP This reports describes the rationale, model, implementation and outcomes for the predictability of task execution in a human agent robot team developed in WP5.

Availability Available.

2.3 van Vught, W “Taxonomy of tasks”

Bibliography van Vught, W. Taxonomy of Tasks. Memo on work performed.

Abstract To create a taxonomy of tasks that fits the TRADR environment, existing task models are studied. Different aspects of the existing task models are combined to fit the TRADR environment, ontology and agents the best. The created model and the implementations are elaborated further within this memo. Lastly, the analysis of the task model is discussed.

Relation to WP This reports describes the rationale and model for the taxonomy of tasks developed in WP5.

Availability Restricted.

2.4 Ontology Design for Task Allocation and Management in Urban Search and Rescue Missions

Bibliography Elie Saad, Koen V. Hindrinks, and Mark A. Neerincx (2018). Ontology design for task allocation and management in urban search and rescue missions. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018)*, volume 2, pages 622629, Funchal, Madeira, Portugal, January 2018. SCITEPRESS.

Abstract Task allocation and management is crucial for human-robot collaboration in Urban Search And Rescue response efforts. The job of a mission team leader in managing tasks becomes complicated when adding multiple and different types of robots to the team. Therefore, to effectively accomplish mission objectives, shared situation awareness and task management support are essential. In this paper, we design and evaluate an ontology which provides a common vocabulary between team members, both humans and robots. The ontology is used for facilitating data sharing and mission execution, and providing the required automated task management support. Relevant domain entities, tasks, and their relationships are modeled in an ontology based on vocabulary commonly used by firemen, and a user interface is designed to provide task tracking and monitoring. The ontology design and interface are deployed in a search and rescue system and its use is evaluated by firemen in a task allocation and management scenario. Results provide support that the proposed ontology (1) facilitates information sharing during missions; (2) assists the team leader in task allocation and management; and (3) provides automated support for managing an Urban Search and Rescue mission.

Relation to WP This paper documents the design approach and ontology design for the TRADR task management support developed in WP5.

Availability Published by SCITEPRESS²

2.5 Rozemuller, C., Hindriks, K.V., Neerincx, M.A., “Task priority decision support for effective search and rescue missions”

Bibliography Rozemuller, C., Hindriks, K.V., Neerincx, M.A. Task priority decision support for effective search and rescue missions. Memo on work performed.

²<http://www.icaart.org/BooksPublishedScitepress.aspx>

Abstract In a human-UGV search and rescue mission, the team leader should be able to quickly assess the priority of tasks and assign the most important ones to his robot-operators. However, even if the criteria for setting task priorities are clear, this remains a cognitive challenge. We developed a decision support system that identifies tasks and assigns priority, in order to support a team leader in making more effective task assignments.

Relation to WP This memo documents the work performed on adding support to the TRADR system for task priority setting that was developed in WP5.

Availability Restricted.

2.6 Mioch, T., “CTL analysis of team leader”

Bibliography Mioch, T. CTL analysis of team leader. Memo on work performed.

Abstract We describe the analysis of the CTL of team leaders during TEval 2017.

Relation to WP This report describes the rationale, method and analysis of the CTL of the team leader during TEval 2017. The CTL input is used during the evaluation via the working agreements developed in WP5.

Availability Restricted.

2.7 On the Effects of Team Size and Communication Load on the Performance in Exploration Games

Bibliography ³ Chris Rozemuller, Mark A. Neerincx, and Koen V. Hindriks (2018). On the Effects of Team Size and Communication Load on the Performance in Exploration Games. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018)*, volume 2, pages 221–230, Funchal, Madeira, Portugal, January 2018. SCITEPRESS.

Abstract Exploration games are games where agents (or robots) need to search resources and retrieve these resources. In principle, performance in such games can be improved either by adding more agents or by exchanging more messages. However, both measures are not free of cost and it is important to be able to assess the trade-off between these costs and the potential

³This work was completed before Y4 but only recently accepted in 2017.

performance gain. The focus of this paper is on improving our understanding of the performance gain that can be achieved either by adding more agents or by increasing the communication load. Performance gain moreover is studied by taking several other important factors into account such as environment topology and size, resource-redundancy, and task size. Our results suggest that there does not exist a decision function that dominates all other decision functions, i.e. is optimal for all conditions. Instead we find that (i) for different team sizes and communication strategies different agent decision functions perform optimal, and that (ii) optimality of decision functions also depends on environment and task parameters. We also find that it pays off to optimize for environment topologies.

Relation to WP This paper reports on Y3 work performed for WP5 on the trade-off between increasing the communication load versus the robot team size.

Availability Published by SCITEPRESS⁴

2.8 Racioppa, S., Willms, C., “Team communication processing in TRADR”

Bibliography Racioppa, S., Willms, C. “Team communication processing in TRADR”. Unpublished technical report.

Abstract This technical report describes the work on the *Nuance Mix.nlu* ASR model in TRADR. Speech analysis is an essential component to keep track of the tasks performed by the team in a rescue mission with the TRADR system. Analyzing the human-human interactions of past TRADR missions, we found out which speech acts are required to automatically set up and update the status of the tasks assigned by the Team leader during the mission. To improve the accuracy of the speech recognition, we developed and trained a model on the audio data collected from past missions, using the cloud-based speech engine Nuance Mix.nlu.

Relation to WP The report describes the core language processing component developed in WP5.

Availability Restricted.

⁴<http://www.icaart.org/BooksPublishedScitepress.aspx>

2.9 Anikina, T., “Measures of Semantic Similarity for Dictionary Extension and Intent Recognition”

Bibliography Anikina, T.. “Measures of Semantic Similarity for Dictionary Extension and Intent Recognition”. Unpublished technical report.

Abstract This report describes various similarity measures and how they can be applied to address several problems relevant for the TRADR1 project. On the one hand, similarity metrics can be used to retrieve semantically similar words which will help extend existing dictionaries. On the other hand, intent recognition methods used in the TRADR project can be evaluated and in some cases improved using semantic similarity.

Relation to WP The report describes tools for extending the coverage of the core language processing components developed in WP5.

Availability Restricted.

References

- [1] I. Abi-Zeid and J. R. Frost. SARPlan: A decision support system for Canadian search and rescue operations. *European Journal of Operational Research*, 162(3):630–653, 2005.
- [2] T. Anikina. Measures of semantic similarity for dictionary extension and intent recognition. Technical Report, unpublished, 2018.
- [3] T. Bosse, J. van Diggelen, M. A. Neerincx, and N. J. Smets. Supporting human-robot teams in space missions using epartners and formal abstraction hierarchies. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 383–399. Springer, 2015.
- [4] J. M. Bradshaw, P. Feltovich, M. Johnson, M. Breedy, L. Bunch, T. Eskridge, H. Jung, J. Lott, A. Uszok, and J. van Diggelen. From tools to teammates: Joint activity in human-agent-robot teams. In *International Conference on Human Centered Design*, pages 935–944. Springer, 2009.
- [5] C. S. Burke, K. C. Stagl, E. Salas, L. Pierce, and D. Kendall. Understanding team adaptation: A conceptual analysis and model. *Journal of Applied Psychology*, 91(6):1189, 2006.
- [6] J. Y. C. Chen and M. J. Barnes. Human agent teaming for multi-robot control: A review of human factors issues. *IEEE Transactions on Human-Machine Systems*, 44(1):13–29, Feb 2014.
- [7] C. H. Chou, F. M. Zahedi, and H. Zhao. Ontology for developing web sites for natural disaster management: Methodology and implementation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 41(1):50–62, Jan 2011.
- [8] H. M. Cuevas, S. M. Fiore, B. S. CAIDWELL, and L. StRAtER. Augmenting team cognition in human-automation teams performing in complex operational environments. *Aviation, space, and environmental medicine*, 78(5):B63–B70, 2007.
- [9] J. English and S. Nirenburg. Ontology learning from text using automatic ontological-semantic text annotation and the web as the corpus. In *AAAI Spring Symposium on Machine Reading*. AAAI Press, 2007.
- [10] J. Garreau. Bots on the ground. *Washington Post*, 6, 2007.
- [11] T. R. Giele, T. Mioch, M. A. Neerincx, and J.-J. C. Meyer. Dynamic task allocation for human-robot teams. In *ICAART (1)*, pages 117–124, 2015.

- [12] R. S. Gutzwiller, S. H. Espinosa, C. Kenny, and D. S. Lange. A design pattern for working agreements in human-autonomy teaming. In *International Conference on Applied Human Factors and Ergonomics*, pages 12–24. Springer, Cham, 2017.
- [13] R. S. Gutzwiller and D. S. Lange. Tasking teams: supervisory control and task management of autonomous unmanned systems. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 397–405. Springer, 2016.
- [14] R. S. Gutzwiller, D. S. Lange, J. Reeder, R. L. Morris, and O. Rodas. Human-computer collaboration in adaptive supervisory control and function allocation of autonomous system teams. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 447–456. Springer, 2015.
- [15] M. Harbers and M. A. Neerincx. Value sensitive design of a virtual assistant for workload harmonization in teams. *Cognition, Technology & Work*, 19(2-3):329–343, 2017.
- [16] L. Jacobsson, J. Malec, and K. Nilsson. Modularization of skill ontologies for industrial robots. In *Proceedings of ISR 2016: 47th International Symposium on Robotics*, pages 1–6, June 2016.
- [17] M. Johnson and J. Bradshaw. The fundamental principle of coactive design: Interdependence must shape autonomy. . . . , *Institutions, and Norms* . . . , pages 172–191, 2011.
- [18] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, M. B. Van Riemsdijk, and M. Sierhuis. Coactive design: Designing support for interdependence in joint activity. *Journal of Human-Robot Interaction*, 3(1):43–69, 2014.
- [19] P. Johnson and H. Johnson. Knowledge analysis of tasks: task analysis and specification for human-computer systems. *Engineering the human-computer interface*, pages 119–144, 1989.
- [20] G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich. Ten challenges for making automation a” team player” in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95, 2004.
- [21] S. Konstantopoulos, J. Pottebaum, J. Schon, D. Schneider, T. Winkler, G. Paliouras, and R. Koch. Ontology-based rescue operation management. In J. Löffler and M. Klann, editors, *Mobile Response*, pages 112–121, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

- [22] K. L. Kraemer and A. Pinsonneault. Technology and groups: Assessment of the empirical research. *Intellectual teamwork: Social and technological foundations of cooperative work*, pages 373–405, 1990.
- [23] G.-J. M. Kruijff, M. Janíček, S. Keshavdas, B. Larochelle, H. Zender, N. J. Smets, T. Mioch, M. A. Neerincx, J. V. Diggelen, F. Colas, et al. Experience in system design for human-robot teaming in urban search and rescue. In *Field and Service Robotics*, pages 111–125. Springer, 2014.
- [24] R. S. Lange, Douglas S. and Gutzwiller. Human-autonomy teaming patterns in the command and control of teams of autonomous systems. In D. Harris, editor, *Engineering Psychology and Cognitive Ergonomics: 13th International Conference, EPCE 2016, Held as Part of HCI International 2016, Toronto, ON, Canada, July 17-22, 2016, Proceedings*, pages 179–188, Cham, 2016. Springer International Publishing.
- [25] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, and M. Beetz. Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553, Oct 2010.
- [26] X. Li, S. Bilbao, T. Martin-Wanton, J. Bastos, and J. Rodriguez. Swarms ontology: A common information model for the cooperation of underwater robots. *Sensors (Basel, Switzerland)*, 17(3):569–588, Mar. 2017.
- [27] Q. Limbourg and J. Vanderdonckt. Comparing task models for user interface design. *The handbook of task analysis for human-computer interaction*, 6:135–154, 2004.
- [28] S. Liu, C. Brewster, and D. Shaw. Ontologies for crisis management: a review of state of the art in ontology design and usability. In *Proceedings of the 10th International ISCRAM Conference*, pages 349–359, Baden-Baden, Germany, 2013.
- [29] Y. Liu, G. Nejat, and J. Vilela. Learning to cooperate together: A semi-autonomous control architecture for multi-robot teams in urban search and rescue. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6, Oct 2013.
- [30] J. T. Malin, D. L. Schreckenghost, D. D. Woods, S. S. Potter, L. Johannesen, M. Holloway, and K. D. Forbus. Making intelligent systems team players: Case studies and design issues. volume 1: Human-computer interaction design. 1991.
- [31] T. Mioch. Ctl analysis of team leader. Memo, unpublished, 2018.

- [32] T. Mioch, M. Peeters, and M. A. Neerincx. Improving human-robot cooperation through working agreements. Memo, unpublished.
- [33] M. Missikoff and F. Taglino. *An Ontology-based Platform for Semantic Interoperability*, pages 617–633. Springer, Berlin, Heidelberg, 2004.
- [34] R. R. Murphy. *Disaster Robotics*. The MIT Press, 2014.
- [35] M. Neerincx. Cognitive task load design: model, methods and examples. *Handbook of cognitive task design*, pages 283–305, 2003.
- [36] M. Neerincx and J. W. Streefkerk. Interacting in desktop and mobile context: Emotion, trust, and task performance. In *European Symposium on Ambient Intelligence*, pages 119–132. Springer, 2003.
- [37] H. S. Packer, N. Gibbins, and N. R. Jennings. Collaborative learning of ontology fragments by co-operating agents. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 89–96, Aug 2010.
- [38] H. S. Packer, N. Gibbins, and N. R. Jennings. Forgetting fragments from evolving ontologies. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web – ISWC 2010*, pages 582–597, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- [39] T. R. A. Giele, T. Mioch, M. A. Neerincx, and J.-J. C. Meyer. Dynamic Task Allocation for Human-robot Teams. *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 117–124, 2015.
- [40] S. Racioppa and C. Willms. Team communication processing in TRADR. Technical Report, unpublished, 2018.
- [41] C. R. Rivero, I. Hernández, D. Ruiz, and R. Corchuelo. Benchmarking data exchange among semantic-web ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):1997–2009, Sept 2013.
- [42] C. Rozemuller, K. V. Hindriks, and M. A. Neerincx. Task priority decision support for effective search and rescue missions. Memo, unpublished, 2018.
- [43] E. Saad, K. V. Hindriks, and M. A. Neerincx. Ontology design for task allocation and management in urban search and rescue missions. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART)*, Funchal, Madeira, Portugal, jan 2018. SCITEPRESS.

- [44] E. Salas, R. Grossman, A. M. Hughes, and C. W. Coultas. Measuring team cohesion: Observations from the science. *Human Factors*, 57(3):365–374, 2015.
- [45] C. Schlenoff and E. Messina. A robot ontology for urban search and rescue. In *Proceedings of the 2005 ACM Workshop on Research in Knowledge Representation for Autonomous Systems*, KRAS '05, pages 27–34, New York, NY, USA, 2005. ACM.
- [46] E. P. B. Simperl and C. Tempich. Ontology engineering: A reality check. In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems*, volume 4275 of *Lecture Notes in Computer Science*, pages 836–854, Berlin, Heidelberg, 2006. Springer.
- [47] M. P. Singh. An ontology for commitments in multiagent systems. *Artificial intelligence and law*, 7(1):97–113, 1999.
- [48] M. Tambe. Towards flexible teamwork. *Journal of artificial intelligence research*, 7:83–124, 1997.
- [49] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz. Representation and exchange of knowledge about actions, objects, and environments in the roboearth framework. *IEEE Transactions on Automation Science and Engineering*, 10(3):643–651, July 2013.
- [50] W. van Vught. Modelling the predictability of task execution in a Human-Agent-Robot Team. Master’s thesis, Vrije Universiteit Amsterdam, the Netherlands, 2017.
- [51] W. van Vught. Taxonomy of Tasks. Technical report, TNO, Soesterberg, 2017. Unpublished.
- [52] M. Van Welie, G. C. Van der Veer, and A. Eliëns. An ontology for task world models. In *Design, Specification and Verification of Interactive Systems 98*, pages 57–70. Springer, 1998.

On the Effects of Team Size and Communication Load on the Performance in Exploration Games

Chris Rozemuller¹, Mark Neerincx^{1,2} and Koen V. Hindriks¹

¹*Intelligent Systems, Delft University of Technology, Mekelweg 4, Delft, The Netherlands*

²*TNO, Postbus 23, 3769ZG Soesterberg, The Netherlands*

Keywords: Performance, Exploration Game, Communication, Team Size, Topology, Resource Redundancy, Task Size.

Abstract: Exploration games are games where agents (or robots) need to search resources and retrieve these resources. In principle, performance in such games can be improved either by adding more agents or by exchanging more messages. However, both measures are not free of cost and it is important to be able to assess the trade-off between these costs and the potential performance gain. The focus of this paper is on improving our understanding of the performance gain that can be achieved either by adding more agents or by increasing the communication load. Performance gain moreover is studied by taking several other important factors into account such as environment topology and size, resource-redundancy, and task size. Our results suggest that there does not exist a decision function that dominates all other decision functions, i.e. is optimal for all conditions. Instead we find that (i) for different team sizes and communication strategies different agent decision functions perform optimal, and that (ii) optimality of decision functions also depends on environment and task parameters. We also find that it pays off to optimize for environment topologies.

1 INTRODUCTION

Exploration games are games where agents (or robots) need to search for resources and retrieve these resources (Hindriks and Dix, 2014). Many real-life applications are instances of such games including, for example, package delivery problems (which sometimes only require minimal search) to search and rescue missions (where search typically takes most of the time). A task in an exploration game is defined by a specific finite (sub)set of all available resources that need to be located and retrieved in a particular order (a task is defined as a sequence of resource types). The order imposed on the items to be retrieved is a key difference with typical foraging tasks. We assume that the map (i.e., topology of the environment) that needs to be explored is finite and known but that the initial distribution of resources is unknown. In this paper, performance in exploration games is measured by the time to complete a given task.

Task performance in exploration games can be improved by adding more agents because, in principle, they can perform tasks in parallel. This is true even if agents do not communicate with each other. If resources are sufficiently available and agents act rationally, it is possible to solve an exploration game without any

communication. The “only” condition that agents that do not communicate need to satisfy is that they do not waste resources (they need to ensure that resource consumption is necessary to complete the task). The performance gain of adding one more agent, however, decreases relative to the number of agents that are already deployed. Even worse, if physical size of robots and the space they occupy is also taken into account, there typically is a point where adding more robots will decrease performance again as robots become obstacles blocking each other’s movement (Rosenfeld et al., 2006). But even if we abstract from such ‘navigational issues’, as we will do in this paper, and we can safely assume that adding more agents will not decrease performance, we cannot assume that adding more agents to the mix will increase performance. Finite tasks that can be completed can only require at most a finite amount of effort, which means that there must be a point at which adding another agent will not yield any performance gain any more.

Besides by adding more agents, performance can usually also be improved by adding communication between agents. Communication, for example, can be used to avoid duplication of effort. If agents inform each other about the locations they have visited, for example, agents can avoid exploring that lo-

cation twice. Similarly, by communicating about the targets agents set themselves (their goals) agents can avoid retrieving the same type of resource twice. It is known from the literature that even a limited exchange of messages can already have a huge impact on performance, see e.g., (Farinelli et al., 2004). Given that exploration games are finite, it also is clear that there is some point at which more communication does not lead to any performance gains any more. A team of agents will only be able to perform better if they can further improve the coordination of their actions by communicating more. We aim to increase our understanding of how much communication can contribute to the performance of an agent team in exploration games.

Like foraging games, optimal solutions are unknown for exploration games in general and therefore agent-based simulation approaches are used to empirically establish the performance of a coordination strategy (Zedadra et al., 2017). The results from this empirical research can then be used to design better and more efficient coordination strategies for different types of exploration games. Our work is motivated by this and aims to provide guidelines that can inform this design. As a designer, it is particularly useful to understand how much performance can be improved by either adding another agent to the mix or by increasing the communication load for a given number of agents. The number of agents (robots) and number of messages exchanged between agents on average can be viewed as a budget that is available to a designer. It is useful for a designer to better understand the return on investment of adding another agent or increased the communication load.

In general, from a design perspective, it is simpler to add another agent to a system than to increase the messages that agents exchange. Exchanging more messages typically requires a more complicated coordination strategy to be effective and thus complicates system design because interdependencies between agents are increased. A more complicated coordination strategy, moreover, comes at the cost of higher processing power, additional requirements on hardware, and higher risks of failure. Additional design complexity, however, may be justified when communication can yield dramatic performance gains. This is sometimes the case, as we noted above, but to understand when requires an insight into when such performance gains are to be expected. Providing this kind of insight is one of the aims of this paper.

The main contributions of our paper are (i) that we provide convincing evidence that there is not a single coordination strategy that is optimal for all cases and

(ii) show which type of coordination strategies are best suited for optimizing performance for specific map topologies of an exploration game. We also show how the performance of different strategies depends on team size and communication load, and how performance is influenced by additional factors such as map and task size, and resource redundancy.

The remainder of the paper is organised as follows. Section 2 discusses related work. In Section 3 we introduce our approach and discuss the agent decision functions used in our simulations. Section 4 presents the experimental set-up we have used to study performance gains. In Section 5 we discuss our results. Section 6 concludes the paper.

2 RELATED WORK

The effects of coordination and communication on performance have extensively been studied empirically for real robot systems; (Farinelli et al., 2004) provides a good survey. This survey presents a detailed overview of coordination mechanisms that have been proposed and concludes that to obtain reasonable performance in most cases little communication is required. However, none of the reported studies provides a detailed study of the trade-off between communication, team-size, and performance.

(Pitonakova et al., 2016) demonstrates the value of coordination by showing that both social and non-social coordination mechanisms, i.e. with and without communication, can improve a robot team's efficiency. (Pini et al., 2013) studies coordination mechanisms in relation to how tasks are partitioned. They conclude that communication is beneficial to avoid duplication of effort, but has the drawback of biasing the exploration, even slowing it down in some cases as a result. These works focus on issues such as avoiding collisions and path finding, whereas our focus is more on task related coordination issues, such as avoiding duplication of effort and efficient destination allocation. Our results, moreover, go beyond these studies by providing a more detailed overview of which performance gains can be achieved by means of increasing communication and team-size, while taking the influence of various environment factors into account.

(Liemhetcharat et al., 2015) uses a set-up similar to ours, but studies heterogeneous teams instead of homogeneous teams as we do, and a setting where resources are replenished instead of consumed as is the case in our work. Our focus, moreover, is on communication load and we take more environmental factors into account.

Several simulation-based studies that also investi-

gate the relation between team-size, communication, and performance have used the Blocks World for Teams (BW4T) simulation environment. BW4T is a testbed for exploration games (Johnson et al., 2009) with blocks of different colors as resources and has been specifically designed for analysing and evaluating the ability to cooperate in multi-agent teams. Both (Harbers et al., 2012) and (Wei et al., 2014) use BW4T to investigate the impact of different types of communication on team performance. These works examined agents that use four different communication protocols: (i) agents that do not communicate, (ii) agents that only exchange information about the knowledge they have about the environment, i.e. the location of resources, (iii) agents that only communicate their intentions, i.e. what they plan to do, and (iv) agents that both communicate about their knowledge and intentions. (Harbers et al., 2012) concludes that it is more effective to communicate about intentions than about knowledge. (Wei et al., 2014) shows, moreover, that interference between robots can diminish the positive effects of communication. Running simulations using BW4T takes time, however, which has limited the size of the experiments that could be run to less than 10 agents and small environments. In contrast, we vary the size and topologies of environments in our experiments, systematically explore the impact of various other environment parameters, and vary team sizes from 1 to a 100 agents.

3 SIMULATION APPROACH

We use a discrete simulation model to empirically investigate performance of various agent decision functions for exploration games. In our approach we systematically vary parameters that define exploration games, including map topology and size, distribution of different types of resources, task size, and number of agents (team size) deployed for completing the task. We also systematically vary basic tactics that agents use for exploration and coordination.

3.1 Simulator

We use a very fast multi-agent system simulator for exploration games developed in MATLAB called MEG (short for Matlab Exploration Game). The map topology of an exploration game is modelled by a (symmetric) distance matrix that consists of distances between each pair of locations on the map. We assume that a unique location is singled out on the map as the target location where resources need to be delivered called the *drop zone*. Even though a single

drop zone somewhat simplifies the task, we believe this constraint is reasonable for the purposes of this paper and also limits the number of topologies that we need to simulate to a feasible number. The simulator keeps track of which resources are being retrieved and moved to other locations. Each time an exploration game is loaded and initialized the simulator randomly distributes a pre-defined amount of resources on locations on the map. The initialization of the task sequence is based on the redundancy of resources and on task length parameters. All agents initially are located in the drop zone.

MEG is a discrete event simulator where all agents perform actions simultaneously at a discrete point in time T and the effects of these actions on the environment are computed at the end of each time step. After each step the global clock T is increased to $T + 1$. Agents maintain a model of the environment (their beliefs) and set targets (goals to go somewhere or retrieve a known resource). The simulator automatically updates the beliefs of agents based on the resources perceived at the location an agent is at and the messages received from other agents at the end of each time step. The performance of the simulator thus effectively only depends on the time needed to compute a decision on which action to perform next for each agent and to compute the effects of performing these actions simultaneously on the environment. Our simulator is fast enough to run millions of simulations in a reasonable time (compared with, for example, a real-time simulator such as the BW4T we gain a speed increase of roughly a factor 25,000).

Figure 1 shows the control flow that is executed by the simulator. Choices are in blue, actions are in gray, and for each time step T the green block repeats the same cycle again by executing all agents for as long as the task set has not been completed yet. For each agent a , the simulator keeps track of two important parameters: the target location $R(a)$ and $N(a)$ the time that agent a is estimated to arrive at and occupy that location. As long as $N(a) > T$, at a time step an agent performs one step towards its target location lowering the distance that still needs to be travelled. Once an agent reaches its target location, i.e. $N(a) = T$, it gets access to this location if it is not occupied by another agent. If the agent gets access, it occupies the location and the boolean mapping *occupied* is updated to model this fact. We note that agent movement is not obstructed by locations that are already occupied but can move freely past such locations. An agent that *occupies a target location* either drops a resource that it retrieved previously, retrieves a resource that is available at the location if it believes that resource still

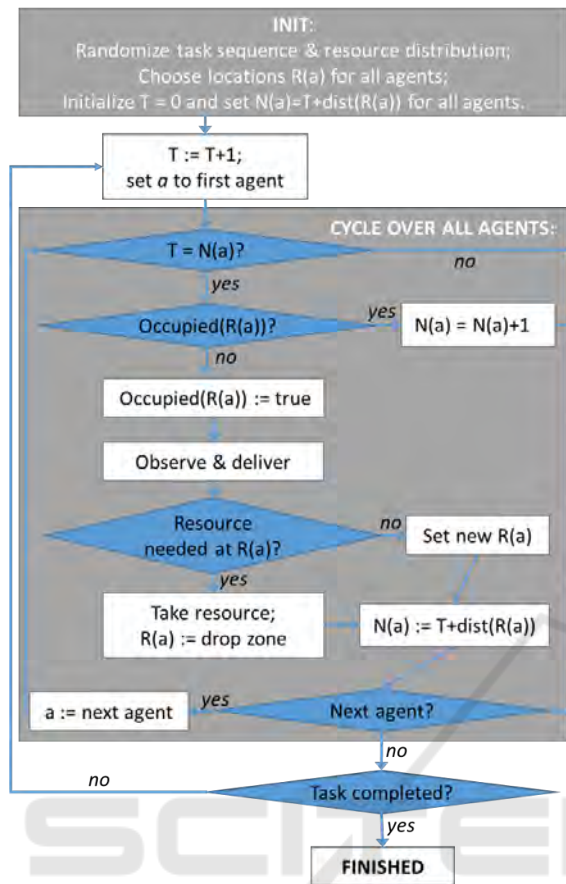


Figure 1: Control Flow of MEG Simulator.

needs to be delivered at the drop zone, or otherwise selects a (different) target location. Selection of a target location depends on the agent's exploration strategy. If the agent cannot access the location yet, the estimated time of arrival $N(a)$ is increased with one and at the next time step it is checked again whether the agent can access the location.

At a target location agents can observe which resources are available at that location. At the beginning of each turn agents are updated on task progress, i.e. on the resource types that still need to be delivered. The simulator also allows agents to communicate updates on perceived resources (belief updates) as well as changes to the targets that they set for themselves (their goals) *to all other agents*. That is, agents can broadcast their belief updates and goals. Whether agents do so depends on the decision function they use. Messages sent to an agent are available to that agent at the beginning of the next turn.

The output of a simulation run consists of the *number of turns*, i.e. T , that were needed to retrieve and deliver all resources required to complete the task and the *average number of messages each agent sent*.

3.2 Environment and Task Parameters

The simulation model allows for varying a number of parameters, including the following:

- **Size of the map**, i.e. the number of locations;
- **Map topology**, or structure of the map, i.e. whether locations are connected and, if so, what the distance between these locations is;
- **Task size**, or the length of the task sequence, i.e. the number and order of the resource types that need to be located and retrieved;
- **Resource redundancy**, i.e. a multiplier r that ensures that a the number of items of a particular resource type available on the map is r times the number that is actually needed to complete the task; $r = 1$ means that the resources available exactly match what is needed.

The map size and topology parameters determine the cost of travelling to a location. By varying the distances between locations we aim to establish when a random exploration tactic will perform better than a greedy tactic. If, for example, the distance between any two locations is the same, then randomly selecting a room cannot result in increasing the travelling costs. If, on the other hand, some locations are much more distant than other locations, then at some point it may become more efficient to apply a greedy tactic and select the closest location to avoid having to travel long distances.

By increasing the task size, we aim to establish which tactics will increase the efficiency of larger teams more because they facilitate multiple agents to perform more subtasks effectively in parallel. Finally, we aim to verify whether a higher resource redundancy factor r will favour greedy and communication tactics because resources can be assumed to be available closer to an agent's current location.

3.3 Agent Decision Functions

In order to complete a task, agents need to explore and coordinate their efforts to locate and retrieve resources. We therefore specify basic tactics for exploration and coordination which can be combined to obtain different types of agent decision functions. In this work we assume agent teams are homogeneous, i.e. all agents use the same strategy, and do not consider heterogeneous teams.

Exploration Tactics The basic tactics that we consider for exploration are a greedy and a random tactic. We assume that agents are always greedy, i.e. select

the closest location, if they know a location where a resource can be found that is needed next. (Recall that we assume the map is known.) If an agent uses the *greedy exploration tactic* it will also select the closest location that has not yet been visited. By default, we assume that an agent is a greedy explorer. The random tactic instead selects a random location from those that have not been visited yet. We further differentiate by considering when the random exploration tactic is applied: at the beginning of a game (*random start tactic*) and thereafter during the rest of the game (*random exploration tactic*). Of course, an agent may both select a target location randomly at the start as well as during the game, thus effectively applying the combination of both tactics. Furthermore, we assume that our agents are persistent: They will keep trying to get access to a location until it becomes available and will not select a new target location before they have gained access to their current target location. We thus have two tactics that an agent can use instead of the default greedy exploration tactic:

- **Random Start Tactic.** At $T = 0$, agents randomly select a location that has not yet been visited to go to instead of going to the closest location first.
- **Random Exploration Tactic.** At $T > 0$, agents randomly choose a target location that has not been visited to explore next.

The effect of randomly selecting an (initial) target location is that agents will more evenly distribute over the map. This usually will reduce duplication of effort as fewer agents will try to visit the same room. The downside of an agent that applies a random tactic is that on average it will increase the distance traveled compared to an agent that uses a greedy tactic. The random start tactic will only initially give rise to a more even distribution on the map whereas the random exploration tactic will ensure exploration of all parts of the map more evenly later on in the game.

Coordination Tactics. Agents that are careful not to waste resources do not need communication to complete a task in an exploration game. By default, we therefore assume that agents do *not communicate*. Agents, however, can coordinate their efforts better when they exchange information. We consider two communication tactics. First, agents that use the *updates communication tactic* exchange updates on the locations that they visit: They inform other agents about which locations they have visited and share the information about resources found at those locations with other agents. Other agents use this information to not (re)visit a location already visited by another agent and to (greedily) select locations where resour-

ces needed can be retrieved by using the information about resources they thus obtain. In contrast with exploration tactics, which only affect the agent's *own* behaviour, it is important to realize that communication tactics have an effect on the behaviour of *other* agents. Second, agents that use the *target communication tactic* share with other agents which resource they are delivering when they retrieve that resource at a location. Other agents use this information to not also then target the delivery of that same resource type but instead will focus on delivering the resource needed next.

- **Updates Communication Tactic.** Agents communicate about which locations they have visited, and about which resources are (no longer) available at a location. Other agents will not consider exploring locations that have been explored already and will retrieve resources based on information received from other agents.
- **Target Communication Tactic.** Agents communicate about which resource they are delivering when they retrieve a resource at a location that is required next to complete the task sequence. Other agents will anticipate and not deliver the same resource.¹

The cost of communication is based on the total count of all messages that are sent from one agent to another agent. That is, all individual messages are counted instead of counting the single broadcast action that sends a message to all other agents as a single message. We do so because each individual message demands resources and requires establishing the reliability of the transmission of a message from one agent to another agent. Generally speaking, the target communication tactic results in fewer messages being sent than the updates communication tactic. The former tactic only requires one message to be sent for each resource needed to complete the task whereas for implementing the latter tactic a message needs to be sent for every resource discovered and each room that is visited for the first time.

Admittedly, our tactics are quite basic and can be refined to obtain more sophisticated variants of these tactics. We believe, however, that for our purposes these tactics are useful as they provide for basic but fundamentally different strategies to complete tasks

¹Since communication is only available at the beginning of the next turn this introduces a new issue: Multiple agents can decide to deliver the same resource. Therefore, at the drop zone, agents also check if their resource is still required eventually. If not, they will abandon it and continue with the remainder of the task.

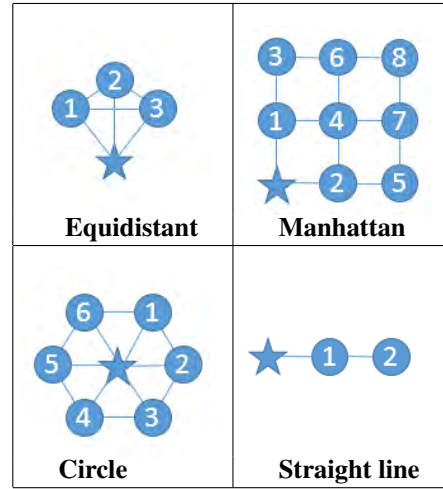
in an exploration game. They allow us to establish the effects of almost completely opposite tactics, i.e. those of greedy versus random exploration tactics and of no communication versus the communication of updates and chosen target locations. For exploration, for example, we thus can establish whether focus (on nearby locations) versus spread (aiming for visiting as many different locations as possible) improves performance in an exploration game. This allows us to investigate the efficiency of tactics that are generally applicable to robots that are tasked with exploration rather than to focus on details of very specific instances of exploration games.

Table 1: Labels of agent decision functions.

	Random Start	Random Exploration	Updates Comm.	Target Comm.	Optimal (Sometimes)
A					Yes
B			✓		Yes
C		✓			No
D		✓	✓		No
E				✓	Yes
F			✓	✓	Yes
G		✓		✓	No
H		✓	✓	✓	No
I	✓				Yes
J	✓		✓		No
K	✓	✓			Yes
L	✓	✓	✓		No
M	✓			✓	Yes
N	✓		✓	✓	Yes
O	✓	✓		✓	Yes
P	✓	✓	✓	✓	No

Decision Functions. The four tactics introduced above can be used to create variations of the default greedy exploration agent that does not communicate at all. This gives rise to 16 different agent decision functions. Table 1 introduces labels (single letters) used to reference these decision functions in the remainder of the paper. For example, agents that use decision function *G* initially choose to visit the location closest to it and then randomly visit unexplored locations until a resource that is needed is located; upon retrieving a resource they communicate to all other agents that they will deliver this resource but do not update other agents about locations visited and resources discovered. The last column in the table indicates whether a decision function is optimal at least some of the time, i.e. dominates other decision functions for a specific experimental condition. Decision functions that are never optimal are always outperformed by another decision function.

Table 2: Topologies used in our simulations.



4 EXPERIMENTAL SETUP

In our simulation experiments we varied the parameters discussed in Section 3.2. For map size, we varied the number of locations and used **10, 20 and 40 locations**. We defined **4 different topologies** illustrated in Table 2. Edges that connect locations in Table 2 have a distance of one.² In the *equidistant topology* the distance to every other location is the same. In the *Manhattan topology* all locations are placed on a grid with the drop zone located in a corner of the grid. In the *circle topology* all locations are placed in a circle around the drop zone. Finally, in the *straight line topology* all locations are placed on a straight line with the drop zone located at either end of that line.

We varied task size and used **sequences with length 3, 6 or 12 resources**. In our simulations we used 7 resource types. A task sequence of 7 resources thus could require agents to retrieve resource items each of a different type in a particular order. The constraint that resources need to be delivered to the drop zone in a particular order complicates the task if multiple agents work on it in parallel as coordination may be required to avoid duplication of effort. Resources needed are randomly distributed over available locations with a **redundancy factor of 1, 2 or 4**. If, for example, the task requires 2 resource items of type τ and the redundancy factor is 4, then 8 resource items of type τ will be made available on the map. The choice of resources that need to be collected to complete a task and the distribution of resources in the environ-

²This may not always result in a feasible geometry in 3d space using only straight lines; we use these structures to demonstrate coordination issues related to connectivity and distance.

ment, moreover, was randomized for each simulation run. Finally, we varied team size and performed simulations with **1, 2, 3, 5, 10, 20, 30, 50, and 100 agents** and used all **16 agent decision functions** specified in Section 3.3.

Combined, these variations of parameters define $3 * 4 * 3 * 3 * 9 * 16 = 15,552$ conditions. Each type of simulation condition is repeated a 100 times to average out variation, thus giving a total of 1,555,200 different simulation runs that were performed.

The parameter settings that we have chosen for our simulations allow us to evaluate the efficiency of a range of coordination strategies and tactics in the smallest possible environments that are still interesting as well as in very complex environments. We have also included team sizes that are more than twice as large as the largest number of locations which ensures we will reach a saturation point in which all locations will be permanently occupied and adding another agent can at best only have a marginal effect.

4.1 Performance Normalization

The main performance measure of a simulation run is the number of time steps T it takes to finish a task. The value T for a specific run depends on environment and task parameters, tactics, and team size and therefore these values can differ widely for different simulation runs. To be able to compare the performance for different parameter sets we normalize T by scaling it to a value between 0 and 1. We do so by means of the theoretically minimal time needed to complete a task T_{opt} , i.e. the optimal lower bound to complete the task possible, and use the average time T_{single} to complete a task by a single greedy agent as an upper bound. It is reasonable to assume that multiple agents should be able to outperform a single greedy agent and would need fewer than T_{single} steps.

We can compute the lower bound T_{opt} analytically by assuming that the locations of all resources are known and the agents available perform sub-tasks in parallel. Of course, without prior knowledge of resource locations it is very hard to get close to this theoretical optimum as it requires each agent to travel directly to the right locations. We determine the upper bound T_{single} empirically by running simulations for each set of environment and task parameters also for a single agent. We use the greedy agent A (see Table 1) as our results show that it is most efficient if we use only a single agent to complete a task.

The normalized performance measure T_{norm} then is computed for each team size consisting of a specific type of agent as follows:

$$T_{norm} = \frac{T - T_{opt}}{T_{single} - T_{opt}} \quad (1)$$

Note that we must always have $T_{opt} < T_{single}$ for tasks of length > 1 as multiple agents can perform subtasks in parallel. The value T_{norm} is independent from environment and task parameters and therefore provides an indication of the ability of the agents to coordinate their efforts.

5 RESULTS AND DISCUSSION

We computed the average performance measure T_{norm} and the communication load of agents (i.e., the number of messages exchanged per agent) for each simulation condition that was run a 100 times. We thus obtained performance measures for different types of budgets (i.e., different investments of number of agents and communication load). We plotted these outcomes in a 3D point cloud with coordinates performance, team size, and communication load. For each configuration of agent type, task size, and environment type (map size, topology, resource distribution), an optimal performance point can be found in this cloud for any team size and communication load coordinate. We averaged the best performance for each of these coordinates and linearly interpolated between the data points to obtain the 2D heat map of Figure 2, assuming that performance never degrades when more resources are invested. The colours in this map show the best performance T_{norm} that we found for each type of budget. Blue corresponds with a lower T_{norm} value, i.e. better performance, and red with a higher T_{norm} value. The color of each point in this map thus indicates the best possible performance for a given budget of agents and messages.

Figure 2 shows different bands of the same colour, e.g., green for $T_{norm} = 0.5$, where performance is the same. These iso-performance frontiers or *performance levels* visualize the trade-off between the investment of more agents or communication resources that designers of a multi-robot system can make to obtain a desired performance. For example, the figure shows that a performance of .4 can be achieved with about 20 agents but also with only 7 agents that communicate with each other. Interestingly, all of these frontiers show a more or less sharp edge around 20 messages per agent. This suggests that agents with on average that kind of communication load are a good choice to obtain a desired performance level. The performance gain of investing more agents that communicate less is small and increasing the communication load would still require almost the same number of robots to achieve a similar or better performance. Our results thus suggest that providing agents with basic communication skills can significantly reduce the

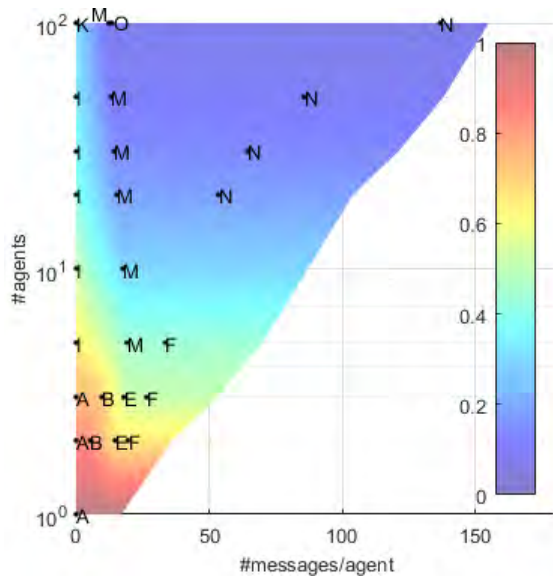


Figure 2: Performance heatmap with optimal agent types.

number of robots that are needed to achieve a specific performance level.

5.1 Performance and Tactics

Figure 2 also identifies for a given budget which agent types (see Table 1) are able to perform best and at what performance level. It is interesting to observe that for different budgets different agent types perform best. In other words, there is no single agent type that dominates all others and performs best for arbitrary budgets. For example, for budgets with less than 10 agents, agent types A, B, E, and F perform best but performance is rather low with a normalized performance $T_{norm} > .6$, whereas for budgets with more than 10 agents types K, M, N, and O perform nearly-optimal. For small budgets agent type E, a greedy agent that communicates about target delivery, seems a particularly attractive choice. For larger budgets M, a greedy agent with random start and target communication tactics, is a particularly attractive choice.

If we average over environment and task parameters, moreover, we can conclude that 7 of our 16 agent types are dominated by some other agent type. Generally speaking, we find that an initial random distribution of agents becomes more important when team size becomes larger and that target communication makes a larger team more effective. Somewhat remarkably the updates communication tactic is only useful for small budgets; this appears to be the case because the target communication tactic provides sufficient information for coordinating larger agent teams. Note, however, that if we do not average over all parameters

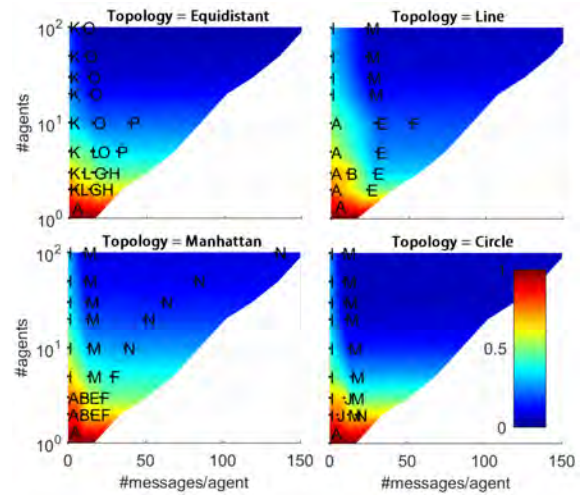


Figure 3: Heatmaps for different topologies.

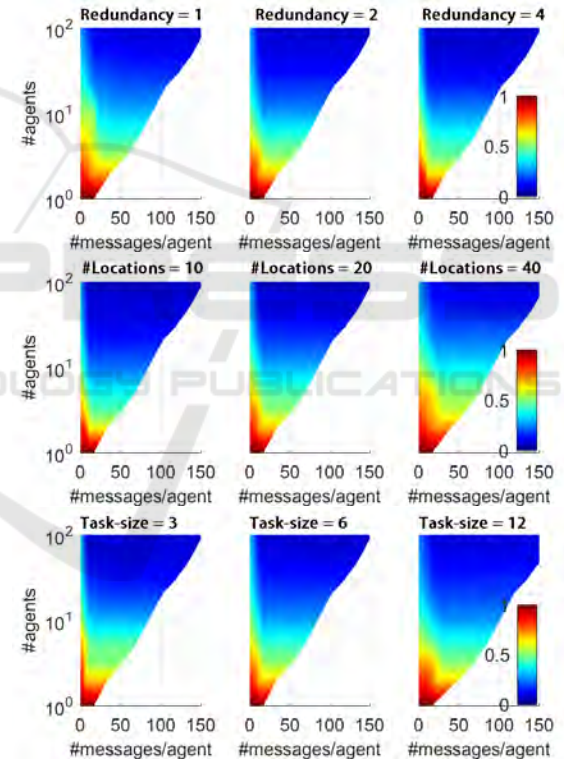


Figure 4: Heatmaps for environment parameters.

other tactics may become relevant again. For example, for specific topologies such as e.g. the equidistant topology and for small budgets the random exploration tactic (agent type G, see Figure 3) can outperform other tactics.

5.2 Environment Influence

Figures 3 and 4 show how different environment and task parameters influence the performance for different budgets. These figures show best performance for specific values of a parameter while averaging over all other parameter settings. For example, the left-upper heatmap in Figure 3 shows the performance heat map for the equidistant topology.

By comparing heat maps for different redundancy factors, we can conclude that impact of redundancy on performance is rather small; there is less need for communication for higher redundancy factors. The impact of map size on performance is as one would expect: larger maps require more agents to achieve similar performance levels. In contrast, if the task size is increased, more communication between agents is needed to achieve similar performance levels.

Finally, we find that the type of topology has a rather large effect on the shape of the performance levels that are visible in the heat maps (see Figure 3). Most notably, whereas for most parameters the agent types that perform best match those of Figure 2, this turns out to be not the case for different topologies. The heat map for the Manhattan topology matches best with the heat map of Figure 2 averaging over all parameters. But for other topologies the heat maps are quite different. For example, we find that on a line and equidistant topology the target communication tactic can significantly increase efficiency (agent I versus M and K versus O), but the update communication tactic only yields significant performance gains on a Manhattan topology. We conclude that it is particularly interesting to fine-tune and optimize an agent decision function for a specific topology.

6 CONCLUSIONS

This paper investigates what the best performance is that can be achieved with a given budget, i.e. an investment of a specific number of agents and communication load per agent. We use a simulation approach and a discrete event simulator for exploration games to empirically obtain insights in how performance depends on different tactics used for composing a strategy for deciding what to do next. Several exploration tactics including greedy and random exploration tactics and several communication tactics are evaluated. We find that there does not exist one dominant strategy but that for different budgets different sets of tactics perform best.

Our results can inform designers of multi-agent systems for exploration game type applications. First,

our results can inform the choice of budget itself and can be used to make a trade-off between budgets and performance. Moreover, we found that certain combinations of tactics are outperformed by other strategies and thus are best avoided. Finally, we have shown that for different environment and task parameters different strategies perform best. In particular, we found that fine-tuning of agent coordination strategies is particularly useful if agents only have to handle a specific type of environment topology.

In future work we plan to refine and evaluate the agent tactics used in this paper and to study particular mechanisms for optimizing performance in specific types of environment topologies.

ACKNOWLEDGEMENTS

This work was supported by European Union's Seventh Framework Programme for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763.

REFERENCES

- Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multi-robot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015–2028.
- Harbers, M., Jonker, C., and Van Riemsdijk, B. (2012). Enhancing team performance through effective communication. In *Proceedings of the 4th Annual Human-Agent-Robot Teamwork Workshop*, pages 1–2.
- Hindriks, K. V. and Dix, J. (2014). GOAL: A multi-agent programming language applied to an exploration game. In *Agent-Oriented Software Engineering*, pages 235–258. Springer.
- Johnson, M., Jonker, C., van Riemsdijk, B., Feltyovich, P. J., and Bradshaw, J. M. (2009). *Joint Activity Testbed: Blocks World for Teams (BW4T)*, pages 254–256. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Liemhetcharat, S., Yan, R., and Tee, K. P. (2015). Continuous foraging and information gathering in a multi-agent team. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15*, pages 1325–1333.
- Pini, G., Gagliolo, M., Brutschy, A., Dorigo, M., and Birattari, M. (2013). Task partitioning in a robot swarm: a study on the effect of communication. *Swarm Intelligence*, 7(2):173–199.
- Pitonakova, L., Crowder, R., and Bullock, S. (2016). Task allocation in foraging robot swarms: The role of information sharing. In *Proceedings of the Fifteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE XV)*, pages 306–313. MIT Press.

- Rosenfeld, A., Kaminka, G. A., and Kraus, S. (2006). *A Study of Scalability Properties in Robotic Teams*, pages 27–51. Springer US, Boston, MA.
- Wei, C., Hindriks, K. V., and Jonker, C. M. (2014). The role of communication in coordination protocols for cooperative robot teams. In *ICAART (2)*, pages 28–39.
- Zedadra, O., Jouandeau, N., Seridi, H., and Fortino, G. (2017). Multi-agent foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling*, 5(1):3.



Ontology Design for Task Allocation and Management in Urban Search and Rescue Missions

Elie Saad¹, Koen V. Hindriks¹ and Mark A. Neerincx^{1,2}

¹*Department of Intelligent Systems - Interactive Intelligence Group, Delft University of Technology, Delft, The Netherlands*

²*The Netherlands Organization for Applied Scientific Research (TNO), Soesterberg, The Netherlands*

Keywords: Task Management, Human Robot Collaboration, Ontology, Urban Search and Rescue.

Abstract: Task allocation and management is crucial for human-robot collaboration in Urban Search And Rescue response efforts. The job of a mission team leader in managing tasks becomes complicated when adding multiple and different types of robots to the team. Therefore, to effectively accomplish mission objectives, shared situation awareness and task management support are essential. In this paper, we design and evaluate an ontology which provides a common vocabulary between team members, both humans and robots. The ontology is used for facilitating data sharing and mission execution, and providing the required automated task management support. Relevant domain entities, tasks, and their relationships are modeled in an ontology based on vocabulary commonly used by firemen, and a user interface is designed to provide task tracking and monitoring. The ontology design and interface are deployed in a search and rescue system and its use is evaluated by firemen in a task allocation and management scenario. Results provide support that the proposed ontology (1) facilitates information sharing during missions; (2) assists the team leader in task allocation and management; and (3) provides automated support for managing an Urban Search and Rescue mission.

1 INTRODUCTION

After a disaster, such as a hurricane or an industrial accident, firefighters arrive on site with different types of robots to perform Urban Search And Rescue (USAR) response efforts (Murphy, 2004). During these efforts, human-robot team leaders have to act fast and allocate tasks to firemen (robot operators, infield rescuers, etc.) to assess the situation and save potential victims. Firemen will then collaborate with robots, such as unmanned ground vehicles (UGVs) and aerial vehicles (UAVs), to execute these tasks. Such human-robot collaboration sets new challenges concerning task allocation and management (Murphy et al., 2008; Lewis et al., 2010).

In this race against time, any wrong decision when allocating or executing a task may cause additional damages and risk the lives of both victims and rescuers. For example, after an earthquake hit Mexico City in 1985 when limited resources for inspecting a disaster site were available, many rescuers died while executing USAR tasks. Of these rescuers, 65 drowned in the area where they were assigned to search for victims (Casper and Murphy, 2003).

Nowadays, the amount of resources for

exploration and reconnaissance has increased, in particular because of the availability of rescue robots (Murphy, 2014) with various types of sensors and detectors. The downside of adding robots to the mix, however, is that this also increases the workload of the team leader who needs to select which resource(s) to use for performing a given task. Moreover, the use of robots leads to a substantial increase of the heterogeneous data gathered from the disaster site (point clouds generated by cameras and lasers, etc.) which needs to be analyzed and taken into account when deciding on the appropriate actions to take. For example, if a robot detects a gas leak close to a fire, the raw data should be analyzed by verifying the gas density and its proximity to fire, before deciding to send firefighters to that area.

An USAR team leader needs to be aware of the current situation (Riley and Endsley, 2004) and take many elements into consideration in order to allocate tasks effectively to a human-robot team. Key elements, for example, include (1) the available actors (operators, rescuers, UGVs, UAVs) and their current state (location, battery level, workload, etc.); (2) the capabilities (sensory, locomotion, communication) and devices (thermo and waterproof cameras, fire and

gas detectors, etc.) that actors have or are equipped with; (3) a map of the disaster area to keep track of what has been explored so far, including the detected points of interest (POIs), such as hazardous materials and victims; and (4) the communication options available, including their range so as not to lose contact with the team.

To address the aforementioned challenges, a sophisticated human-robot task management model is needed to support (1) building and maintaining shared situation awareness (Endsley, 1995); (2) preventing overload of operators and infield rescuers (Neerincx, 2003); and (3) handling interoperability (Missikoff and Taglino, 2004) and interdependencies (Johnson et al., 2014) within a human-robot team. We focus on creating a knowledge representation for mapping data to a unified semantics (Sheth, 1999; Xu and Zlatanova, 2007) which is shared between humans and robots. We propose the use of an extensive ontology (Schlenoff and Messina, 2005; Mescherin et al., 2013) for conceptually representing USAR domain entities and tasks, and their relationships. This conceptualization is useful for task allocation and management as it (1) captures relevant domain aspects; (2) facilitates communication and information sharing; (3) provides reasoning support throughout a mission; and (4) improves human-robot collaboration.

This paper describes the ontology we designed for assisting firemen and the team leader in mission control. It provides a common vocabulary to be used by both, firemen and robots, and automated support for task management in particular for the team leader. Throughout the paper we use a scenario from the TRADR project, short for 'Long-Term Human-Robot Teaming for Robot Assisted Disaster Response', see (Kruijff-Korbayová et al., 2015), to illustrate the main concepts. The scenario involves a reconnaissance sortie and barrel inspection which requires a team composed of a firemen who is the team leader, robot operators and infield rescuers, and robots including UGVs and UAVs. Team members receive tasks from the team leader to scout the disaster area and gather more information about a barrel that has been spotted.

The main contributions of this paper are (1) the design of an ontology; (2) the design of a user interface which displays (parts of) the ontology and is part of automated task support that assists the team leader during a mission; and (3) the evaluation of the ontology and automated task support by firemen in a task allocation and management scenario.

This paper is organized as follows. In Section 2 we review related work. In Section 3 we present the development and structure of the task management

ontology. In Section 4 we present the user interface which displays parts of the ontology and is part of a larger search and rescue system. In Section 5 we discuss the evaluation of the designed ontology based on its use during a reconnaissance sortie use case and interviews with firefighters. In Section 6 we conclude the paper and give directions for future research.

2 RELATED WORK

Ontologies are widely used to represent domain knowledge and facilitate information sharing in many applications (Rivero et al., 2013; Missikoff and Taglino, 2004). According to (Liu et al., 2013a), existing crisis oriented ontologies describe concepts and characteristics related to a single subject area (type of disasters, geography, meteorology, etc.). Such ontologies are designed to address the requirements of a specific application.

In USAR robotics ontologies, work has been done on the cooperation between autonomous or semi-autonomous multi-robot teams (Liu et al., 2013b). For example, the robot ontology suggested by Schlenoff and Messina, 2005, is centered on representing the concepts about robots and their capabilities when assisting in USAR missions. This ontology is very relevant, but differs from our focus on providing task management support for a team leader.

Other robotics ontologies focus on the robot and its interaction with a specific environment. For example, (Jacobsson et al., 2016) proposes an ontology for industrial use and (Li et al., 2017) for underwater robots. The KnowRob ontology (Tenorth et al., 2013) offers a set of ontologies to model robots and their capabilities and actions. The OpenRobot (Lemaignan et al., 2010) ontology (ORO), which shares many concepts with the KnowRob ontology, is focused on robot interaction with humans, but assumes that robots are completely autonomous. Given the current state of the art in USAR robotics, we focus instead on remotely operated robots.

3 ONTOLOGY DESIGN

In this section, we first discuss the development process we followed to build the ontology introduced in this paper. Then we present the overall structure of the ontology and discuss the main concepts that have been included to support task management.

3.1 Ontology Development Process

The development process we followed to build our ontology is adapted from (Simplerl and Tempich, 2006). This iterative process consists of different phases, as illustrated in Figure 1. First, we analyzed the task management system requirements in search and rescue domain by interviewing firefighters and reviewing the literature and state-of-the-art ontologies. Second, the required domain entities and their relationships are conceptualized based on common vocabulary used by firemen. Third, the ontology is implemented as RDF triples in the OWL 2 Web Ontology Language syntax¹ and visualized through the system's user interfaces. Fourth, the modeled ontology is evaluated by firemen using search and rescue use case scenarios. Lastly, in the maintenance phase, the ontology is refined and extended with new concepts and entities needed for addressing the requirements of additional use cases.

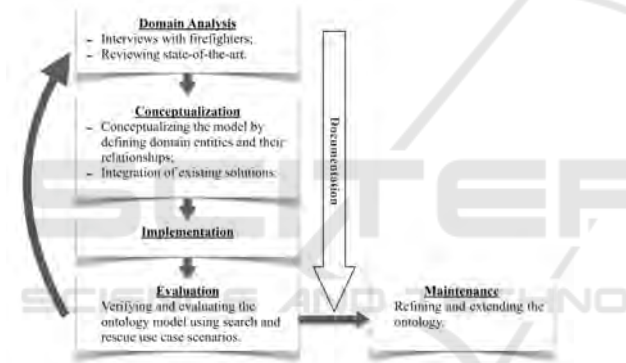


Figure 1: Ontology development process adapted from (Simplerl and Tempich, 2006).

3.2 Overall Ontology Structure

The ontology introduced in this paper has been designed to be part of a larger ontology (Bagosi et al., 2016) that is used in the TRADR system, a European project for search and rescue response efforts². We aimed to make the design of the ontology (1) flexible and extensible, to be able to easily append new components, for e.g. covering more use cases; (2) reusable, so it can be applied in different types of missions; and, perhaps most importantly, (3) readily understandable by firemen, the key rescuers in our domain, in order to facilitate task allocation and management.

To this end, we summarized and grouped the knowledge gathered into multiple modules in order

¹<https://www.w3.org/TR/owl2-syntax/>

²<http://www.tradr-project.eu>

to append additional components as we extend the core ontology. Four of these modules are relevant and part of the task management ontology, as illustrated in Figure 2. The ActorModule groups the agents, both humans and robots, as actors along with their properties such as roles and team affiliations. The CommunicationModule groups all concepts needed to facilitate data gathering and exchange between team members, such as communication events (messages, notifications, etc.), media types (video, photo, audio, etc.) and data gathering devices (infra red sensor, camera, etc.). The EnvironmentModule groups the environmental events (hurricane, flood, chemical leakage, etc.) and structures. Finally, the MissionModule contains the concepts and entities needed when setting up and planning a mission, including a taxonomy of tasks and POIs.

3.3 Modeling and Requirements

The detailed design of our ontology has been based on our discussions and interviews with firefighters, experts in the field, and also has been inspired by Robin R. Murphy's research on rescue robotics (Murphy, 2014). Our ontology is aimed at providing a common vocabulary which is useful for task management by facilitating data sharing and communication between team members. The concepts represented in our ontology therefore include all the relevant entities and information categories that are needed for task allocation and management during USAR sorties. The latter are executed using remotely operated robots. We briefly discuss the key concepts that have been included in the different modules and their relationships, as illustrated in Figure 2.

3.3.1 Actor Module

The actor ontology represents the human and robot actors along with their properties. The actors are resources used for responding to the disaster. By representing the roles, status, capabilities, and related concepts in the ontology, the latter can provide a basis for automated support for task management. The system, for example, can reason which actors might be suitable for performing a specific task.

- **Actor Roles and Teams** - During an USAR mission, human and robot actors collaborate for executing the tasks assigned by the team leader. To know who will do what, human actors have different roles (UGV operator, infield rescuer, etc.) as do robot actors (e.g., ground or aerial explorers). We have based the model of the

discovered so far along with the tasks assigned and their progress.

- **Tasks and relevant properties** - Throughout a mission, the team leader assigns tasks to available actors by specifying the task objective or POI, and providing a clear description. To monitor tasks and track their progress, additional properties have been included such as status (in progress, completed, etc.) and priority. Moreover, each task has a list of required capabilities such as sensing, locomotion and communication, which defines to which actor(s) the task can be allocated.
- **Points of interest (POIs)** - This category includes the entities which can be detected while exploring a disaster site and are meaningful for improving situation awareness when assigning tasks. Each POI has a type (victim, fire, gas, hazards, etc.) and a location. These properties help in knowing which actors to send, where they should go and what might be the risks involved.

4 USER INTERFACE

In order to be able to use and deploy the task management ontology discussed in Section 3, two different GUIs have been designed. These GUIs integrate and provide support for task management in the search and rescue tactical display system (TDS). The TDS is used for tracking the disaster area and has been developed to assist USAR teams in the TRADR project. It contains a map of the disaster site showing the location of actors and the detected POIs (victims,

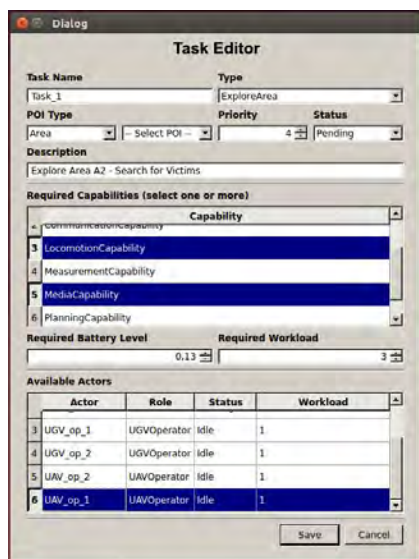


Figure 3: Task Editor for creating and editing mission tasks.

fires, chemical objects, etc.). It also provides support for assessing the disaster site situation and for gathering relevant information about it.

To allocate new tasks or edit existing ones, the team leader uses a task editor, as shown in Figure 3. In this editor, the team leader defines the task properties including (1) a task type (search, go to, take photo, etc.); (2) a POI which defines the task's objective; (3) a priority; (4) a status (pending, in progress, completed, etc.); (5) a description containing specific details or guidelines for the operators; (6) a list of required capabilities which are automatically selected by the system depending on the task type and can be modified by the user; (7) a required battery level; (8) a required workload; and (9) an actor from the list of available actors suggested by the system.

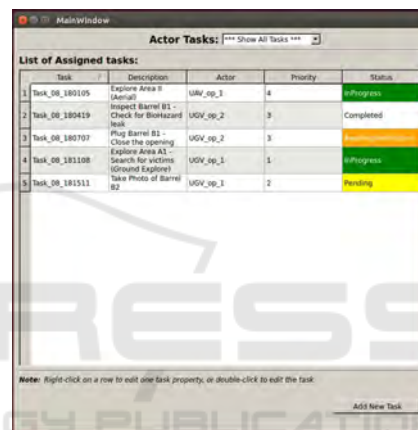


Figure 4: Task Manager for tracking and monitoring tasks.

A second GUI provides the team leader with a task manager interface (Figure 4) which has been designed to enable the team leader to track and monitor the progress of assigned tasks. For each task, the GUI displays its description, assigned actor, priority and status, to provide the team leader with an overview of the execution progress. Mission actors can track the progress of their tasks in the main display system which shows the task name, objective and status. The latter property is continuously updated by actors throughout the execution process.

For every new mission, the main ontology is initialized and loaded in a central repository (we use Stardog triple stores³ which provide support for querying, inferencing and manipulating the knowledge base stored in the repository based on the semantics defined by our ontology). To ensure this repository maintains an up-to-date state of a mission, we developed and use semantic modelers to continuously update the database with

³<http://www.stardog.com>

new knowledge acquired during a mission. These modelers map raw sensor data (e.g., point cloud, GPS coordinates, etc.) onto ontological concepts (POIs, locations, etc.) and store it in the repository.

The mapped data is then used to display and update meaningful information for monitoring the progress of a mission on TDS. It is also used to reason about the represented world and generate notifications related to the task being executed. The aim is to manipulate the gathered knowledge for (1) improving shared situation awareness; and (2) assisting the team leader in its job of assigning tasks by providing automated support. For example, when creating a task for sending an UGV on site to take a photo of a POI, the system queries the knowledge base to display the list of available human actors who operate a UGV equipped with a high-resolution camera and having enough battery life. Whereas when creating a task for picking up a sample to be analyzed, querying the knowledge base returns the human actors operating a UGV equipped with a robotic arm.

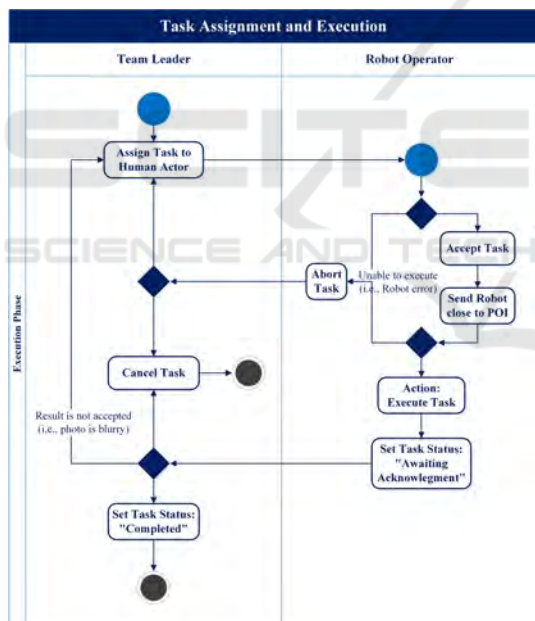


Figure 5: Activity diagram for assigning and executing mission tasks.

Throughout a mission, the team leader will continuously add new tasks or update existing ones (description, priority, etc.). The activity diagram in Figure 5 shows the workflow for assigning and executing a task. First, the team leader assigns a task to an actor. Then, the actor can accept it and start the execution or can abort it when facing technical issues (e.g., robot errors). When the task is executed, the actor sets its status to awaiting acknowledgement

using the task manager. If the result is accepted, the team leader sets the task status to completed. Otherwise, it will be reassigned or canceled.

5 EVALUATION AND RESULTS

To evaluate the use of our ontology in USAR missions, we evaluated it as part of the bigger TRADR system while executing a use case scenario that involves a reconnaissance sortie and the inspection of a barrel. The scenario was executed by firemen teams at the fire department training facility located in Rozenburg, The Netherlands. After each sortie, we interviewed the firefighters team and their leader to obtain their feedback about the ontology (its concepts) and its use for displaying task management related information and content in the task editor and management user interfaces.

5.1 Use Case Scenario

The scenario is based on an industrial accident where an explosion has occurred on site. A team is sent to (1) search for human victims; (2) gather more information about the site; and (3) inspect the area for the presence of chemical hazards and leakages.

First, the team leader has to assign a task to a UAV operator to scout the disaster area. While executing the task, the operator will spot an unidentified barrel and should notify the team leader of this. When the team leader is informed about the barrel, a task should be assigned to an actor operating a UGV with a high-resolution camera to inspect the barrel and take a photo of it. After receiving the photo of the barrel and analyzing it, the team leader should assign a task to the actor operating a UGV with a robotic arm to close the barrel opening and prevent potential chemical leaking. All tasks will be inserted in the Task Manager GUI, as shown in Figure 4, and actors are made aware of the tasks assigned to them.

5.2 Execution

During two days, three firemen teams alternated to practice the use case scenario. They received a quick introduction of the TRADR system and its interfaces for about 20 minutes before starting the execution.

At the beginning of each mission, our ontology was instantiated and loaded with the required entities. In our use case, these entities include (1) four human actors where one has a team leader role, two with UGV operator role and one with UAV operator role; (2) three robot actors where one is a UAV and

two are UGVs; (3) environmental objects such as a barrel and debris; (4) points of interest (POIs) such as fire and gas. Throughout a mission, the team leader used the task management interfaces to allocate tasks and monitor their progress. These interfaces used ontological reasoning to provide the leader with automated support by suggesting available actors for a given task and generating notifications when needed.

After executing each sortie, an informal interview took place with the firemen in order to (1) verify that they were satisfied with the support and interaction offered by the system; and (2) estimate their level of understanding of the ontological concepts represented and shown in the user interfaces (i.e., did the ontology provide a common vocabulary that firemen could understand?). Additional interviews took place with the team leaders to check whether the designed ontology fits their needs when managing tasks and executing the missions.

5.3 Results

The three firemen that played the role of team leader indicated that the Task Manager GUI is easy to use and the ontology concepts are clear and easy to grasp. They each said that the task assignment support was intuitive to use and operated as expected. Even so team leaders also indicated that they needed time to get used to it (they had to switch from their usual practice of taking notes on paper to using the user interface).

Task actors did not bother to manually change the task status when executing them. The reason is that, in a real mission (as mentioned by firemen), the team leader assigns tasks and the operators perform it. They only report back to the team leader and change the task status when they finish the task or whenever they encounter a problem during execution. Therefore, it is suggested that the task status should also be automated somehow by the system.

Furthermore, team leaders indicated that the task editor should be simplified. They suggested to provide default values to some of the fields, especially those related to robots, and hide them when creating a new task. These include the task priority, status, required capabilities, required battery level and required workload. Setting default values to these fields allows the system to provide automated support (1) by suggesting to the team leader the appropriate actors who can execute a given task; and (2) by generating appropriate notifications when a task is wrongly assigned or cannot be executed, which helps the team leader when monitoring task progress.

6 CONCLUSION

When planning and executing an USAR mission, the team leader needs to efficiently allocate tasks to team members for assessing the situation and rescuing potential victims. The leader's job is time-critical and complex which requires, among other things, an awareness of the current situation and the knowledge of the team members capabilities and their actual status. Using an ontology for assisting the team leader in task allocation and management provides a common vocabulary between team members, both humans and robots. The ontology is useful for (1) facilitating data sharing; (2) improving shared situation awareness; and (3) providing automated support in the task management process.

This paper introduced part of the ontology developed for TRADR search and rescue project. The ontology is focused on facilitating human-robot collaboration by means of providing automated support for task allocation and management during USAR missions. It is evaluated using a reconnaissance sortie and barrel inspection use case. The evaluation shows that the ontology constitutes a good basis for providing automated support to assist a team leader in mission planning and task management.

The main contributions have been the design of the ontology and related user interfaces, as well as an evaluation in a search and rescue project scenario.

6.1 Directions for Future Research

Our results helped us gain a better understanding of the needs of firemen in general and in particular of the team leader in USAR missions. The following points need to be taken into consideration and require further analysis. It has become clear that firefighters need more training to use advanced tools based on ontologies and automated support. It remains to be seen how we can further simplify system support and whether this can be achieved by further automation. It will be interesting to design, develop, and evaluate additional automated support related to task status updates. The aim should be to further decrease the workload of firefighters and prevent system automation to feel as a burden.

More evaluation, moreover, is needed and additional use cases should be designed and used for testing and evaluation purposes. Additional use cases may reveal potential gaps not yet covered by our ontology and provide new insights in what is needed to automate task management support. We are particularly interested in verifying whether our task

taxonomy is sufficient for specific subtasks in such use cases. In any case, we expect more refinements will be needed for modeling robot capabilities (e.g., robot arm manipulation). Also, our ontology does not yet provide support for robot-robot interaction, fully autonomous robot operations, underwater operations, and, for example, issues such as network resilience. The aforementioned suggestions and extensions will also require an exhaustive user evaluation to cover more parts of the ontology in different scenarios.

ACKNOWLEDGEMENTS

This work was supported by European Union's Seventh Framework Programme for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763.

REFERENCES

- Bagosi, T., Hindriks, K. V., and Neerincx, M. A. (2016). Ontological reasoning for human-robot teaming in search and rescue missions. In *11th ACM/IEEE HRI*, pages 595–596.
- Casper, J. and Murphy, R. R. (2003). Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385.
- Endsley, M. (1995). Toward a theory of situation awareness in dynamic systems. *Human Factor*, 37:32–64.
- Jacobsson, L., Malec, J., and Nilsson, K. (2016). Modularization of skill ontologies for industrial robots. In *Proceedings of ISR 2016: 47th International Symposium on Robotics*, pages 1–6.
- Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., van Riemsdijk, M. B., and Sierhuis, M. (2014). Coactive design: Designing support for interdependence in joint activity. *J. Hum.-Robot Interact.*, 3(1):43–69.
- Kruijff-Korbayová, I., Colas, F., Gianni, M., Pirri, F., de Greeff, J., Hindriks, K., Neerincx, M., Ögren, P., Svoboda, T., and Worst, R. (2015). Tradr project: Long-term human-robot teaming for robot assisted disaster response. *KI - Künstliche Intelligenz*, 29(2):193–201.
- Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R., and Beetz, M. (2010). Oro, a knowledge management platform for cognitive architectures in robotics. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3548–3553.
- Lewis, M., Wang, H., Chien, S.-Y., Scerri, P., Velagapudi, P., Sycara, K., and Kane, B. (2010). Teams organization and performance in multi-human/multi-robot teams. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 1617–1623.
- Li, X., Bilbao, S., Martin-Wanton, T., Bastos, J., and Rodriguez, J. (2017). Swarms ontology: A common information model for the cooperation of underwater robots. *Sensors (Basel, Switzerland)*, 17(3):569–588.
- Liu, S., Brewster, C., and Shaw, D. (2013a). Ontologies for crisis management: a review of state of the art in ontology design and usability. In *Proceedings of the 10th International ISCRAM Conference*, pages 349–359, Baden-Baden, Germany.
- Liu, Y., Nejat, G., and Vilela, J. (2013b). Learning to cooperate together: A semi-autonomous control architecture for multi-robot teams in urban search and rescue. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6.
- Mescherin, S. A., Kirillov, I., and Klimenko, S. (2013). Ontology of emergency shared situation awareness and crisis interoperability. In *2013 International Conference on Cyberworlds*, pages 159–162.
- Missikoff, M. and Taglino, F. (2004). *An Ontology-based Platform for Semantic Interoperability*, pages 617–633. Springer, Berlin, Heidelberg.
- Murphy, R. R. (2004). Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(2):138–153.
- Murphy, R. R. (2014). *Disaster Robotics*. The MIT Press.
- Murphy, R. R., Tadokoro, S., Nardi, D., Jacoff, A., Fiorini, P., Choset, H., and Erkmén, A. M. (2008). *Search and Rescue Robotics*, pages 1151–1173. Springer, Berlin.
- Neerincx, M. (2003). *Cognitive task load analysis: allocating tasks and designing support*, chapter 13, pages 283–305. Lawrence Erlbaum Associates, NJ.
- Riley, J. M. and Endsley, M. R. (2004). The hunt for situation awareness: Human-robot interaction in search and rescue. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(3):693–697.
- Rivero, C. R., Hernández, I., Ruiz, D., and Corchuelo, R. (2013). Benchmarking data exchange among semantic-web ontologies. *IEEE Transactions on Knowledge and Data Engineering*, 25(9):1997–2009.
- Schlenoff, C. and Messina, E. (2005). A robot ontology for urban search and rescue. In *Proceedings of the 2005 ACM Workshop on Research in Knowledge Representation for Autonomous Systems*, KRAS '05, pages 27–34, New York, NY, USA. ACM.
- Sheth, A. P. (1999). *Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics*, chapter 2, pages 5–29. Springer US, Boston, MA.
- Simperl, E. P. B. and Tempich, C. (2006). Ontology engineering: A reality check. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems*, volume 4275 of *Lecture Notes in Computer Science*, pages 836–854, Berlin, Heidelberg. Springer.
- Tenorth, M., Perzylo, A. C., Lafrenz, R., and Beetz, M. (2013). Representation and exchange of knowledge about actions, objects, and environments in the robocearth framework. *IEEE Transactions on Automation Science and Engineering*, 10(3):643–651.
- Xu, W. and Zlatanova, S. (2007). *Ontologies for Disaster Management Response*, pages 185–200. Springer, Berlin, Heidelberg.