Diggelen, J. van, Beun, R.J. & Werkhoven, P.J. (2009). Intelligent Assistants in Crisis Management: From PDA to TDA. In 21st Benelux Conference on Artificial Intelligence, (Belgian/Netherlands Artificial Intelligence Conference) BNAIC 2009, 29 October 2009 through 30 October 2009, Eindhoven (pp. 401-402).

Intelligent Assistants in Crisis Management: from PDA to TDA

Jurriaan van Diggelen[a], Robbert-Jan Beun[b], Peter J. Werkhoven[b]

[a] *TNO Defense, Security and Safety*
Soesterberg, the Netherlands
[b] *Institute of Information and Computing Sciences*
Utrecht University, the Netherlands

Abstract

This paper discusses a variety of potential applications of intelligent assistants, from personal digital assistant (PDA) to team digital assistant (TDA). We identify two challenges that arise when developing intelligent assistants in crisis management and give an overview of the different technologies that could be used to address these challenges. In particular we will focus on ontologies and policies. For both of these technologies, we present a simulation prototype and discuss insights, obstacles and opportunities.

1 Introduction

Suppose that, on a busy weekday, a collision occurs deep within a major tunnel in the Netherlands. An effective first response in such a situation demands the utmost in human performance as limited time is available, the situation may change unexpectedly and human failure is potentially hazardous. We believe that electronic intelligent assistants will, and should, play an important role in assisting first responders in crisis management.

This paper discusses a variety of potential applications of intelligent assistants, from personal digital assistant (PDA) to team digital assistant (TDA). The PDA could offer individual support by allowing the crisis workers to have wireless access to the measurements of the sensors in the environment, e.g. smoke sensors, thermometers, movement detectors. This application offers *a gateway to ubiquitous information*. A TDA could support a whole team by taking over some of the notifications which are necessary to maintain a coherent team. For example, it could ensure that all team members are aware of the plan laid out by the leader. This application can be seen as *a teamwork fortifier*.

Over the last decade, advances in wireless networks have enabled the technological infrastructure which is required for such applications, i.e. mobile devices can exchange bits and bytes in a relatively secure and consistent way. However, many challenges in artificial intelligence remain to be solved to allow different sensors and mobile devices to exchange meaningful messages, for example to communicate the current state of affairs or the planned course of action. Furthermore, the computer must be able to determine the right time and form to share a certain piece of information.

The purpose of this paper is to identify two challenges that arise as we move from PDA to TDA, and to give an overview of the work we have done towards their solutions. We follow a *simulation to implementation* approach [2]. This means that we start by simulating a scenario. This allows us to generate simulation data, which can be used for validation early in the development phase, and which helps to generate further ideas about the prototype. Later in the development phase, the technology used for the simulation artifact, can be used to implement the system itself.

Particular to our methodology is the explicit use of technology input [19]. This is important as the envisioned technology helps to shape the ideas about novel applications. Furthermore, the interplay between technology and human factors becomes an explicit and integrated part of the development process. In this paper, we will discuss two different technologies. We will focus on *ontologies*, and the role they play in representing and sharing information about actions and circumstances. Furthermore, we will investigate the use of *computational policies* to model the teamwork aspects of crisis management support.

The paper is organized as follows. Section 2 describes the two challenges we will be concerned with in this paper. In Sections 3 and 4, we address both of these challenges by discussing the proposed technological solution, the envisioned applications, and by presenting simulation results. Section 6 concludes the paper and presents directions for future research.

2 Two Challenges for Intelligent Assistants

The most basic form of electronic assistance is a software agent[1] supporting a human in performing his or her task [12]. This approach focuses on the specific tasks that an individual must accomplish as part of the team, regardless of what the team as a whole must accomplish. An example of such an approach is described in [11], where an intelligent agent runs on a PDA to notify a mobile police officer about nearby accidents.

We call this PDA application *a gateway to ubiquitous information*. The ubiquitous computing paradigm implies [20] that an environment contains hundreds of networked computing devices and sensors which could assist people in performing their tasks. In this way, the situation awareness of crisis workers can be improved by allowing them to have access via their PDA's to the information possessed by the sensors in their environment.

Designing the information infrastructure for these kind of systems is far from straightforward. Because different devices may be developed by different manufacturers, they are likely to represent their information heterogeneously, complicating the sharing of information. Furthermore, the system is open, i.e. it is not known beforehand which sensors, PDA's and other devices will constitute the system. This makes it impossible to fix the communication infrastructure in advance.

The different components in the system communicate via a high level communication language. This means that we can view the communication mechanism as a way to exchange *meaningful* messages. This requires a shared understanding on the meanings of terms, i.e. the agents must be semantically interoperable. Whereas more basic forms of interoperability are also required for successful agent communication (for example, to establish a network connection between two agents), this paper focusses on the semantic issues of interoperability.

Not only enabling information exchange is a challenge, but also doing it *efficiently*. We will give three reasons for this claim. Firstly, when vast amounts of data are available, information overload becomes a serious issue due to limited storage and computing resources. Secondly, because the system often consists of mobile devices with limited energy supply, mobile communications, which are heavy on power consumption, should be minimized [6]. Thirdly, the system may also contain humans which are even more easily prone to information overload than computers.

The first technological challenge can be summarized as follows:

Challenge 1 *To allow an intelligent assistant to efficiently exchange semantic information in ubiquitous computing environments*

Things are even more challenging as we move towards the application of a TDA as a teamwork fortifier. In this approach, agents do not focus on the specific tasks performed by humans, but they assist the team as a whole [12]. They facilitate teamwork between humans by aiding communication, coordination, and focus of attention [8]. This approach requires a model of how a team works, in terms of general team objectives, roles and strategies. The teamwork model must be computational, allowing it to be processed by agents and to be useful as a basis for running computer simulations. Summarizing, we can phrase the second challenge as follows:

Challenge 2 *To make an intelligent assistant aware of teamwork processes*

The two challenges outlined above are ordered in increasing complexity, in the sense that the solution for the first challenge is also required for the second challenge. For this reason, we will start our discussion with the most elementary challenge.

3 Individual Support

This section addresses challenge 1, which we will discuss in the context of a PDA offering a gateway to ubiquitous information. As an example, we will consider a crisis management application where information

[1] In this paper, we will refer to computers, PDA's, sensors and other computing devices as *agents*

from different sensors in a tunnel must be combined to inform a crisis worker via a PDA about the potential risks of fire and traffic accidents. In the following subsections, we will discuss ontologies, techniques for communication efficiency, and a simulation environment for agent-agent communication in ubiquitous computing.

3.1 Ontologies

A common ontology has been advocated as the key to achieve mutual understanding between agents as it guarantees that every agent uses the same terms to represent the same meanings [5]. In ubiquitous computing, however, different components typically represent their information at different levels of abstraction, i.e. they view the world differently. These different world-views lead to heterogeneous ontologies which could cause misunderstandings [13]. To deal with this problem, we propose a system with *layered ontologies* [14] where the agents only have parts of their ontologies in common. In this way, every agent maintains its own ontology tailored to its task, and use the common parts of their ontologies for communication.

For example, consider a system used in a tunnel consisting of six sensors (numbered 0 to 5), two computers (numbered 6 and 7) capable of gathering and combining sensor information (which we will call *interpreters*), and one PDA (numbered 8) informing a crisis worker about the current state of the situation. The situation, as we modelled it in our prototyping tool, is depicted in Figure 1. The sensors use simple on-

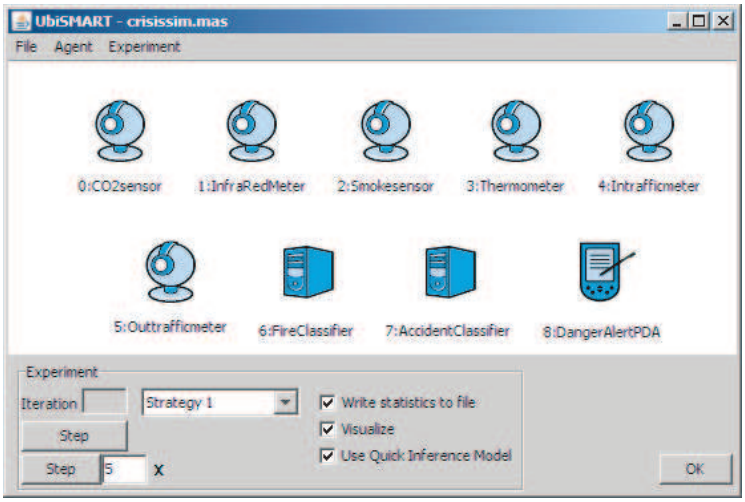

Figure 1: Example setup

tologies allowing them to represent and reason with concrete and measurable information such as *infrared intensity*, *smoke*, *temperature*, *carbon dioxide*, *amount of incoming traffic*, *amount of outgoing traffic*. The two interpreters also possess these ontologies, which allow them to understand the information contained in the sensors. In addition to these, they also possess ontologies containing more abstract terms, such as *fire* and *traffic-accident*. The ontologies of different levels of abstraction are related by *mappings*, i.e. rules which translate between the different levels. For example, *infrared intensity* (which indicates the presence of flames), *smoke*, *temperature*, and *carbon dioxide* are used to derive a potential *fire*. The *amount of incoming traffic* combined with the *amount of outgoing traffic* is used to derive a traffic jam inside the tunnel (which could be the sign of a *traffic accident*). The PDA can process the information produced by the interpreters and raises the level of abstraction in order to present it to the user. For example, it possesses a concept *crisis-level* to indicate the severity of the crisis. For a more in depth discussion on these issues, the reader is referred to [16].

3.2 Communication and Efficiency

Although none of these agents has exactly the same set of ontologies as another agent, they are still capable of communicating with each other. Communication in the system is initiated by an agent aiming to resolve its information needs. Suppose, for example, that the PDA has information need *crisis-level*. It starts by

looking for other agents that can provide valuable information for this concept. Because no other agent in the system has the concept *crisis-level* defined in its ontology, the agent translates this concept to a more concrete concept, such as *fire* and *traffic-accident* and queries it to the interpreters. This query raises an information need for the interpreters, which they try to resolve using the same strategy, i.e. by translating it to the lower-level concepts *infrared intensity*, *smoke*, *temperature*, *carbon dioxide* in order to pass the query on to the sensors. Because the sensors can obtain their information directly from the world, the chain of queries ends there.

Besides enabling information exchange, we also strive for efficiency. In total, we implemented four mechanisms to minimize the information flow between agents. For example, we have ensured that our agents always obtain their information in a way which is most efficient, i.e. by making the right choice between Query and Publish/Subscribe mechanisms. For an extensive and formal treatment of these efficiency measures, the reader is referred to [15].

3.3 Simulation environment

We have implemented a test environment, called Ubismart, which allows developers to easily prototype a ubiquitous computing system corresponding to the architecture discussed above. Figure 1 shows a screen shot. By clicking on one of the icons, a window is opened which can be used to configure the agent. Most importantly, the ontology of the agent is selected here. The ontologies are specified in the OWL language [10], a language for specifying ontologies developed by the semantic web community. Whereas OWL nicely conforms to all kinds of syntactic standards, the language is not very well readable for humans. Therefore, we have used Protégé [7], which is a graphical ontology editor containing an open source Java library for OWL. To understand complex ubiquitous systems involving multiple ontologies, proper visualization of ontologies is crucial. For sensors, also the location of its sensory input must be specified. For PDA's, also the information needs must be specified, i.e. which information the PDA is supposed to present to the user.

Using this tool, the developer can easily obtain hands-on experience with the ontology design of these systems. Furthermore, simulation experiments can be performed to study the information flow between the different components. For example, we used the tool to validate the different communication efficiency measures. Our simulation experiments revealed that, in a typical ubiquitous computing setting, these benefits can be substantial.

4 Teamwork

This section describes the challenge that arises as we move towards the application of a TDA as a teamwork fortifier (challenge 2). Similar to the PDA application discussed in the previous section, this application also involves agent-agent and human-agent communication. In the TDA application, however, communication is not about the current status of the user's physical environment, but is about the status of the user's team members. To realize this, we require a computational teamwork model.

Our approach to teamwork is based on three observations. The first observation is that collaboration almost always entails a reduction in autonomy. We call an actor[2] autonomous if it has control over its own actions and internal state. For the purpose of achieving joint team objectives, we are willing to give up some degree of autonomy. In this way, the agents' activities are constrained towards collaborative activities.

The second observation underlying our approach, is that the behavioral constraints which are applicable to an actor, are dependent on the *role* an actor enacts in the team. For example, a leader has different obligations and authorizations than its subordinates.

The third observation is that some common goal exists which binds the team together. The pioneering research of Cohen and Levesque [3] introduced the notion of a joint persistent goal as the ultimate driving force behind teamwork. In our framework, we adopt the idea of a *collective obligation* [4] for a similar purpose. Whereas an individual obligation describes which actions must be performed by an individual, a collective obligation describes which actions must be performed by a group of agents, regardless of which agent does what. Because a collective obligation usually does not direct activity at the level of the single agent's behavior, we must find a way to translate the collective obligation to the individual level. Specifying this translation forms the core of our teamwork research.

[2]We use the term actor to indiscriminately refer to agents or humans

4.1 KAoS Teamwork Policies

The KAoS policy and services framework [1] possesses useful characteristics for implementing team behavior. It allows the specification of different roles, together with behavioral constraints applying to those roles. We have extended KAoS so it can handle collective obligations, making it an ideal environment for teamwork modelling.

A behavioral constraint in KAoS is specified by a policy which is defined as "an enforceable, well-specified constraint on the performance of a machine-executable action by a subject in a given situation" [1]. There are two main types of polices; authorizations and obligations. Authorization policies specify which actions are permitted (positive authorizations) or forbidden (negative authorizations) in a given situation. Obligation policies specify which actions are required (positive obligations) or waived (negative obligations) in a given situation. Policies are represented in OWL. To hide the complexity of OWL, a dedicated tool called KPAT allows human users to create, modify and manage policies in a very natural hypertext interface.

Below, we list the intuitive meanings behind the teamwork policies we have implemented in KPAT to translate collective obligations (CO's) to the individual level. For a more complete description of the underlying teamwork model as well as for implementation details, the reader is referred to [18].

1. *The leader of a team should adopt the collective obligations of its team as its own individual obligations*
2. *Team members should notify their leader when the collective obligation of their team is triggered*
3. *The leader of a team may request members of its team to perform actions*
4. *The leader of a team may create plans*
5. *An agent should notify the requester after it has performed a requested action*
6. *If the agent knows who will perform the subsequent action, it should notify that agent after it finishes performing its own action*
7. *If the agent knows who will conduct the subsequent action, it is not required to notify the requester after it finishes performing its action*
8. *When no leader is present, the CO is triggered, and the agent knows it can fulfill the CO, it should assume the leader role*
9. *When no leader is present, the CO is triggered, but the agent cannot fulfill the CO, it should notify the whole team of the CO trigger*
10. *An agent should not notify its team about a CO trigger, when it has been notified itself by another team member about that CO trigger*

If there is a team leader, it has a special responsibility and must be treated by the other agents in a special way. The purpose of the first four policies is to lay down these responsibilities. The purpose of policies 5 to 7 is to describes how actions in a plan should be coordinated, i.e. to ensure that the actions in the plan are executed in the appropriate order. The purpose of policies 8 to 10 is to guide the behavior of agents that find themselves in a leaderless team. This may happen either because nobody has been appointed as a leader or else the leader is (temporarily) unavailable.

Our policy-based approach to teamwork has several benefits for developers of agent teams. The first concerns reusability. Because the policies describe near-universal teamwork aspects, they are domain independent and can apply to many kinds of applications, thus saving development time. The second benefit concerns sharedness. Because teamwork requires maintaining common ground among the participants [15], agents benefit when the code that generates team behavior can be shared by all agents. By introducing a shared collection of teamwork policies for the whole system, in conjunction with KAoS monitoring and enforcement capabilities, newly added agents fit easily into the team, no matter who developed them or which language they are programmed in. Next, there is the benefit of separation of concerns. By using KAoS policies, the code that implements teamwork is cleanly segregated from the rest of the agent code. Finally, KAoS policies are very straightforward to read and understand, making them more suitable to implement this kind of behavior than generic rule languages or more low-level programming languages.

In addition to the benefits for agent developers, we also believe that this approach is more conducive to scientific progress towards the much more ambitious goal of human and machine joint activity [18][8]. Although the policies described in this paper are relatively simple, they are fundamental in normal human teamwork. Hence, when agents adopt important aspects of human teamwork, people may find them more predictable and understandable.

4.2 Teamwork Simulation

We tested the policies using a Mars mission scenario developed in the Mission Execution Crew Assistant (MECA) project [9]. This long-term project aims at enhancing the cognitive capacities of human-machine teams during planetary exploration missions by means of an electronic partner. The e-partner helps the crew to assess a situation and determine a suitable course of actions when problems arise. A large part of the project is devoted to developing a requirements baseline, taking into account human factors knowledge, operational demands, and envisioned technology. Developing new prototypes using emerging technologies, such as this one, is a continuous activity in the project.

One of the use-cases that has driven the development of MECA's requirements baseline concerns an astronaut suffering from hypothermia. The initial situation is depicted in Figure 2. Herman is in the Habitat; Anne, Albert and two rovers are in team A; Benny and Brenda are in team B. Benny and Brenda are on a rock-collecting procedure. Suddenly, Benny's space suit fails. Brenda and the MECA system diagnose the problem together and predict hypothermia. Immediate action is required. A rover from team B comes to pick Benny up and brings him to the habitat. Someone with surgery skills and someone with nursing skills await him there and take care of Benny, after which he safely recovers.

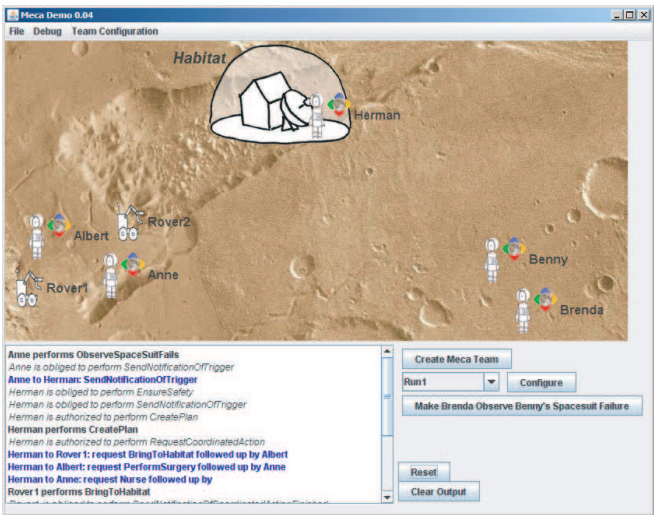

Figure 2: MECA scenario

The seven agents in the example (five astronauts and two rovers) are implemented in Java. Because most of the agent behavior in this demonstration is implemented by the policies, the Java implementation could remain very simple. The agents have the collective obligation to ensure safety after a safety critical event has occurred. The most important aspect of this demonstration is the unfolding of the scenario after the spacesuit failure is observed. This is driven exclusively by KAoS policies. An event trace of the teamwork is shown below.

Brenda performs ObserveSpaceSuitFails
Brenda is obliged to perform SendNotificationOfTrigger
Brenda to Herman: SendNotificationOfTrigger
Herman is obliged to perform EnsureSafety
Herman is authorized to perform CreatePlan
Herman performs CreatePlan
Herman is not authorized to perform RequestCoordinatedAction

Herman is authorized to perform RequestAction
Herman to Rover1: request BringToHabitat
Rover1 performs BringToHabitat
Rover1 is obliged to perform SendNotificationOfRequestedActionFinished
Rover1 to Herman: SendNotificationOfRequestedActionFinished
Herman to Albert: request PerformSurgery
Albert performs PerformSurgery
Albert is obliged to perform SendNotificationOfRequestedActionFinished
Albert to Herman: SendNotificationOfRequestedActionFinished
Herman to Anne: request Nurse
Anne performs Nurse
Anne is obliged to perform SendNotificationOfRequestedActionFinished
Anne to Herman: SendNotificationOfRequestedActionFinished

The events printed in bold are actions; the underlined events are communication actions; the italicized events represent policies that were triggered. In this event trace, some important teamwork properties can be observed, such as maintaining common ground, planning, leadership, etc. Whereas this team is strongly centered around a leader, i.e. it is a *centralized* team, we have also implemented more decentralized types of teams, such as leaderless teams, or self-coordinating teams. A detailed discussion about these teams and their pros and cons is provided in [17].

Besides obtaining simulation results, we believe that the teamwork policies are also useful for implementing the MECA system itself. The teamwork policies could strengthen teamwork between astronauts by aiding communication, coordination, and focus of attention. Because most of the policies we have discussed in this paper are obligations to notify other agents, we could easily let MECA take over these notification tasks. Furthermore, they could help to make robots more predictable and understandable to humans. A good example of an agent enacting the role of an equal team member is the Rover in the hypothermic astronaut scenario.

5 Conclusion

In this paper, we have identified various ways in which intelligent assistants could support humans in crisis management. We have discussed two challenges for developing these assistants and proposed various technological solutions to these challenges. We have demonstrated the use of these technologies using simulations. The next step would be to use the technology in a real prototype, and use that as a way to perform human in the loop evaluations. These are left as topics for future research.

References

[1] J. Bradshaw and et al. Representation and reasoning for daml-based policy and domain services in kaos and nomads. In *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS)*. ACM Press, 2003.

[2] W. J. Clancey, M. Sierhuis, C. Seah, C. Buckley, F. Reynolds, T. Hall, and M. Scott. Multi-agent simulation to implementation: A practical engineering methodology for designing space flight operations. In *Engineering Societies in the Agents World VIII: 8th International Workshop, ESAW 2007, Athens, Greece, October 22-24, 2007, Revised Selected Papers*, pages 108–123, Berlin, Heidelberg, 2008. Springer-Verlag.

[3] P. Cohen and H. Levesque. Teamwork. In *SRI International.*, Menlo Park,CA, 1991.

[4] F. Dignum and L. Royakkers. Collective obligation and commitment. In *Proceedings of 5th Int. conference on Law in the Information Society*, Florence, 1998.

[5] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[6] S. Gurun, P. Nagpurkar, and B. Y. Zhao. Energy consumption and conservation in mobile peer-to-peer systems. In *MobiShare*, pages 18–23, New York, NY, USA, 2006. ACM Press.

[7] H. Knublauch, R. Fergerson, N. Noy, and M. Musen. The protégé OWL plugin: An open development environment for semantic web applications. In *3rd Int. Semantic Web Conference*. Springer, 2004.

[8] T. L. Lenox. *Supporting teamwork using software agents in human-agent teams*. PhD thesis, Pittsburgh, PA, USA, 2000. Adviser- Michael Lewis.

[9] M. Neerincx, A. Bos, A. Olmedo-Soler, U. Brauer, L. Breebaart, N. Smets, J. Lindenberg, T. Grant, and M. Wolff. The mission execution crew assistant: Improving human-machine team resilience for long duration missions. In *Proc. of the 59th International Astronautical Congress (IAC)*, 2008.

[10] M. Smith, C. Welty, and D. McGuinness. OWL Web Ontology Language Guide. *http://www.w3.org/TR/owl-guide/*.

[11] J. W. Streefkerk, M. van Esch-Bussemakers, and M. Neerincx. Designing personal attentive user interfaces in the mobile public safety domain. *Computers in Human Behavior*, 22(4):749–770, 2006.

[12] K. Sycara and M. Lewis. Integrating agents into human teams. *Team Cognition: Understanding the Factors that Drive Process and Performance.*, 2004.

[13] J. van Diggelen, R. J. Beun, F. Dignum, R. M. van Eijk, and J. J. C. Meyer. Ontology negotiation: Goals, requirements and implementation. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 2007.

[14] J. van Diggelen, R. J. Beun, R. M. van Eijk, and P. J. Werkhoven. Modeling decentralized information flow in ambient environments. In *Proceedings of ambient intelligence developments (AmI.D '07)*, 2007.

[15] J. van Diggelen, R. J. Beun, R. M. van Eijk, and P. J. Werkhoven. Agent communication in ubiquitous computing: the ubismart approach. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multi-agent Systems (AAMAS08)*, pages pp. 813–820. ACM Press, 2008.

[16] J. van Diggelen, R.-J. Beun, R. M. van Eijk, and P. J. Werkhoven. Efficient semantic information exchange for ambient intelligence. *The Computer Journal*, 2009.

[17] J. van Diggelen, J. Bradshaw, T. Grant, M. Johnson, and M. Neerincx. Policy-based design of human-machine collaboration in manned space missions. In *Proceedings of Space Mission challenges for Information Technology (SMC-IT)*, 2009.

[18] J. van Diggelen, J. M. Bradshaw, M. Johnson, A. Uszok, and P. J. Feltovich. Implementing collective obligations in human-agent teams using kaos policies. In *Proceedings of International Workshop on Coordination, Organizations, Institutions, and Norms (COIN)*, 2009.

[19] P. van Maanen, J. Lindenberg, and M. Neerincx. Integrating human factors and artificial intelligence in the development of human-machine cooperation. In *Proc. of the 2005 International Conference on Artificial Intelligence (ICAI'05)*, Las Vegas, NV, 2005. CSREA Press.

[20] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.