SAIL: A Social Artificial Intelligence Layer for Human-Machine Teaming

Bob van der Vecht(✉), Jurriaan van Diggelen, Marieke Peeters, Jonathan Barnhoorn, and Jasper van der Waa

TNO, The Hague, The Netherlands
{bob.vandervecht, jurriaan.vandiggelen, marieke.peeters, jonathan.barnhoorn, jasper.vanderwaa}@tno.nl

Abstract. Human-machine teaming (HMT) is a promising paradigm to approach future situations in which humans and autonomous systems closely collaborate. This paper introduces SAIL, a design method and framework for the development of HMT-concepts. Starting point of SAIL is that an HMT can be developed in an iterative process in which an existing autonomous system is enhanced with social functions tailored to the specific context. The SAIL framework consists of a modular social layer between autonomous systems and human team members, in which all social capabilities can be implemented to enable teamwork. Within SAIL, HMT-modules are developed that construct these social capabilities. The modules are reusable in multiple domains.

Next to introducing SAIL we demonstrate the method and framework using a proof of concept task, from which we conclude that the method is a promising approach to design, implement and evaluate HMT-concepts.

Keywords: Human-machine teaming · Social artificial intelligence Collaboration · Design method

1 Introduction

It is hard to imagine a future society without cognitive robots and agents. Artificial systems will fulfil all kinds of societal roles, e.g. personal assistant, tutor, companion, and coach [1]. Rest aside these futuristic developments, autonomous vehicles in civilian transport and autonomous robots for the military are already here. As the integration of autonomous systems in everyday life is increasing the complexity of human-machine interaction, meaningful interaction and collaboration between humans and agents becomes a pressing problem. This has led to an increased interest in this topic, and initiated debates discussing the balance between human control and system autonomy. For example, the discussion on *meaningful human control* [2] addresses a growing concern about system autonomy potentially leading to conflicts with humanitarian law and ethics. The question arises how autonomous systems can support interaction with humans in a way that ultimately allows humans to stay in control.

Human-machine teaming (HMT) is a commonly proposed paradigm to realize this [3, 4]. The challenge of implementing an agent as a team member is often mistakenly

Y. Demazeau et al. (Eds.): PAAMS 2018, LNAI 10978, pp. 262–274, 2018.
https://doi.org/10.1007/978-3-319-94580-4_21

construed as simply *placing a user interface on top of an autonomous system*. In fact, it requires a whole layer of artificial intelligence that encompasses a range of social capabilities enabling the autonomous system to behave in a observable, predictable, and directable way [5].

Whereas current autonomous systems are typically designed to meet the objectives of a specific problem in a specific domain, collaborative team members must be designed also to be part of human-machine teams and require a number of generic social capabilities, for example:

- explaining their behavior in a way that is understandable to humans [6]
- proactively sharing information within the team based on the information needs [8]
- growing towards more mature teams, i.e. human-agent team development [4]

The requirements regarding social capabilities and team functions strongly interact with the properties of both human and artificial team members, as well as with the task environment. For example, if the human cannot do some tasks well in specific circumstances, they must be transferred to the machine. Furthermore, the social capabilities depend on the capabilities of the autonomous system. For example, if the system is able to autonomously adapt its behavior in reaction to the workload of an operator, then the system should be provided with the workload information, whereas a less sophisticated system does not need this. Also the task environment may shape the type of collaboration, e.g. spatial constraints may limit the amount of interactions. Each of these different perspectives must be integrated when designing the HMT. Figure 1 shows how the multiple disciplines come together in defining and implementing collaboration in HMT.

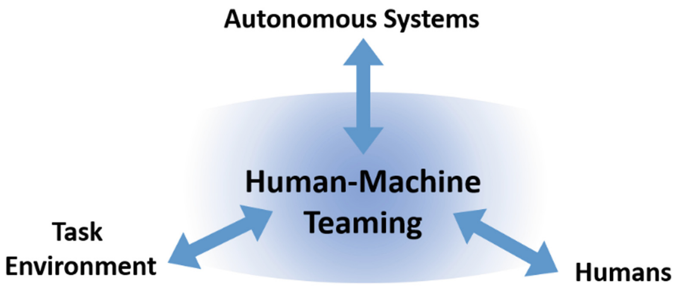


Fig. 1. The HMT requirements for a specific problem are influenced by factors from multiple disciplines: human factors, autonomous systems and the task environment.

Most research on human-machine teaming focuses on the one or two of the relevant topics. Human factors research, for instance, focuses on the relevant properties of HMT from a user perspective, such as trust in automation [7], and shared situation awareness [8]. Based on these results, a few requirements engineering methodologies for HMT have been proposed (e.g. [5]), but they do not provide a system architecture. To implement HMT systems based on these requirements, the engineer could employ techniques from the multi-agent systems (MAS) community (such as policy-based systems [9], agent organizations [10]). However, these agent-based frameworks have been developed with a focus on multi-agent interaction, and do not address particular human factors issues present in HMT. Furthermore, testing methods for MAS (such as [11]), do not provide ways to perform human-in-the-loop tests.

Whereas HMT contains generic properties, a one-size fits all solution for HMT does not exist, as the types of tasks, humans and autonomous systems is different per context. This motivates the need for a method to develop, implement and evaluate HMT in a modular way. As solution we introduce the SAIL (Social Artificial Intelligence Layer) method and framework. The purpose of SAIL is to integrate the human factors and autonomous systems in one design method that is supported by a software framework for implementing and testing HMT-concepts. Starting point of SAIL is that an HMT is developed as an additional layer of social intelligence which complements the autonomous capabilities of the system. The social capabilities to interact and collaborate with the autonomous systems are made available by connecting the system to the SAIL framework.

The SAIL method consists of an iterative development cycle of four steps (Fig. 2). The approach is based on solid systems engineering practices, such as rapid prototyping, short implementation-test cycles, component reuse, and user-centered design.

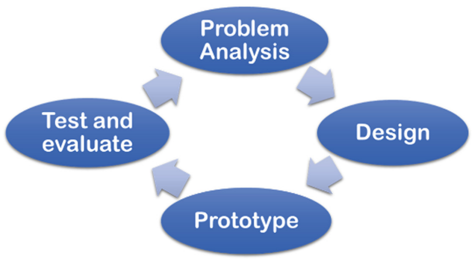


Fig. 2. The SAIL method to develop HMT consists of four phases

With SAIL we facilitate the design and development of reusable components for the most common HMT requirements in the following aspects that are considered relevant:

- It enables in-house development of artificial social intelligence that can be used on top of closed-source systems.
- They can be applied to a variety of autonomous systems with different underlying architectures (i.e. independent of the underlying control logic [12]).
- It allows for evaluating team performance, to iteratively improve HMT-concepts and to facilitate longitudinal team development, as team behavior changes over time [13].

SAIL components interact with each other via human-readable standardized communication languages and protocols. Using SAIL, social requirements for autonomous systems to properly function in HMT can be defined. In the end these requirements can be adopted in the development of autonomous systems themselves by their producers. The paper will further describe the SAIL framework and the development method.

2 SAIL: A Social Artificial Intelligence Layer

2.1 Modular HMT Approach

The base idea behind SAIL is that the autonomous systems that need to be incorporated in the HMT already exist, and we minimize modifications to their internal behavior. Our HMT development approach aims at establishing a social layer between humans and autonomous systems that allows them to function as a coherent team. For this layer we use a modular approach where a single HMT-module facilitates a specific desired aspect for the HMT. Examples are a module to provide status information, or a module to enforce behavior policies. Each module is a single software component that receives certain input and provides a specific output. Modules can be linked in parallel and sequence, as such they can provide each other with enriched inputs.

Figure 3 shows an example of a modular SAIL. On the left are the human users and on the right the autonomous systems. The HMT-modules are situated in between, receiving their input either from the systems or the users.

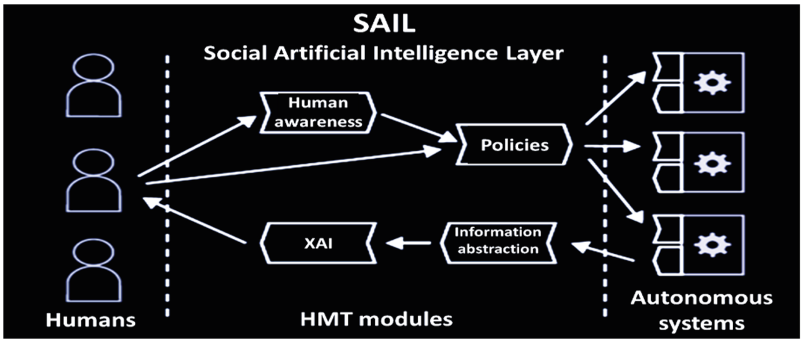


Fig. 3. The modular approach to a social AI layer

This modular approach requires three key components; (1) a library of HMT-modules, (2) a language in which modules and systems can communicate and (3) the ability of autonomous systems to provide internal information and receive external control. In the next paragraphs we elaborate these components of SAIL.

2.2 Communication Language

The actors and modules in the SAIL framework need to communicate and understand each other. As part of SAIL, we introduce HATCL (Human-Agent-Teaming Communication Language), a communication language specifically designed for human machine teaming. HATCL therefore has a central place in the SAIL framework.

The concept of communication languages has been studied since the nineties [14]. For example, FIPA-ACL [15] was developed with the aim to standardize communication in multi-agent systems. Whereas the use of FIPA-ACL decreased in recent years, we partly reuse its ideas in HATCL. HATCL is a language which is used by the different agents (users, autonomous systems and HMT-modules) to communicate. HATCL specifies three main elements: the message format (including performatives), ontologies and protocols.

- Ontologies: following the ontology approach of [16], HATCL consists of a top-ontology laying out the HMT-concepts and message structure. Furthermore it includes domain ontologies for domain specific data structures.
- Performatives specify the intention of a message, such as *inform*, *request*, *confirm* and also *permissions* and *obligations* in a directive setting.
- Protocols are predefined conventions of message exchange to facilitate coordination between actors. E.g., the specification for the expected response to a request, or an interaction scheme to set up a contract for collaboration.

Key is that HATCL facilitates the communication between software, but also human operators. Therefore, the language has to be understandable for humans, and match with their mental concepts. Furthermore, it contains concepts that can be used to set up team functions for coordination and collaboration.

2.3 Interfacing with an Autonomous System

Another requirement for SAIL is the ability of the autonomous systems to interface with the SAIL framework via the HATCL ontology. We assume that the developers of the HMT do not have control about the internal software of the autonomous system. The autonomous systems may not have been designed for the social interaction, however, social capabilities become within reach when an interface with SAIL is created.

Within SAIL, all modules are able to communicate with one another using the HATCL, plus domain ontologies. Regarding the autonomous systems, “semantic anchors” are required that define how ontology concepts are linked to internal data structures and control logic of the autonomous system, as displayed in Fig. 4. The interface of autonomous systems and SAIL therefore consists of the specification of input ontologies (the information it accepts), output ontologies (the information it sends) and the semantic anchors. The semantic anchors form a crucial connection between the developers of the autonomous systems and the developers of the teaming functions. The semantic anchors are the only required adaptation to the autonomous system.

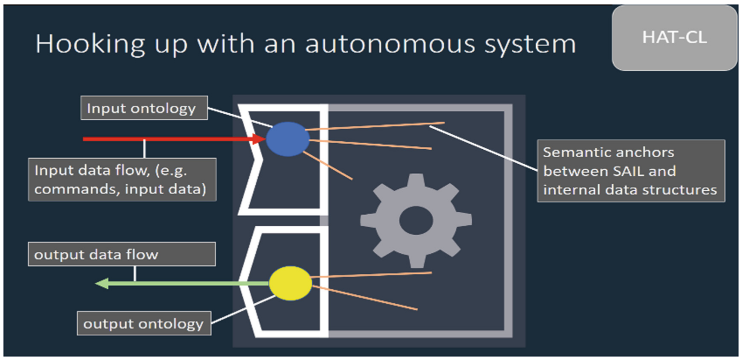

Hooking up with an autonomous system
HAT-CL
Input ontology
Input data flow, (e.g. commands, input data)
output data flow
output ontology
Semantic anchors between SAIL and internal data structures

Fig. 4. Interfacing with an autonomous system requires input-output specifications using ontologies and semantic anchors that connect the inside of the autonomous system

3 Design Guidelines

This section provides guidelines for the four design phases of the SAIL method as shown in Fig. 2. The starting point of SAIL is that an HMT is constructed without changing the internals of the autonomous system. We are able, however, to define an interface to connect it to SAIL.

3.1 Problem Analysis

The key to the problem analysis is to describe the three influence factors shown in Fig. 1, i.e., the autonomous systems, the human factors and the task environment. This leads to constraints and requirements regarding social capabilities. The goal of this phase is to identify the gap between the social capabilities and needs of the autonomous system compared to the '*as is*'-situation, where no SAIL is added to the system. In the next step, the SAIL will be designed to overcome this capability gap. Dimensions that need to be explored for the three influence factors are:

- Autonomous systems: How many systems are involved? Do they differ in capabilities? What are the communication capabilities? Which ontologies are accepted?
- Humans: How many humans are involved? In what role? What is their skills (e.g., trained versus untrained)? Which information is required for their task?
- Task environment: What is the time scale, time pressure? Is it a routine task or does it require creativity? What are the risks of wrong decisions? What does the physical environment look like (e.g., physical distance, environment dynamics, uncertainty)?

Besides these three bullets, an important aspect to consider is the team phase of the HMT that will be designed and evaluated. Are they in a training phase where new HMT-concepts are tested, and the teaming functions are learned? Or is the team already trained and working with certain routines?

3.2 HMT Design

Based on the capability gap between collaborative needs of the team and the capabilities of the autonomous system, the SAIL will be designed. In a modular way components can be designed to meet these social requirements. The philosophy of our method is that these collaborative needs are generic to some extent, and therefore, it is possible to build a library of those HMT-modules. We have identified some categories of functions that support teamwork in HMT (this list does not aim to be a complete list):

- **Situation Awareness:** situation awareness functions share information within the team about the task progression and the environment. An example is a knowledge base that presents information about relevant environment context and position of team members. Such a module facilitates coordination of tasks within the team. To a human team member, a visual user interface might be connected to provide SA.
- **Human Awareness:** human awareness functions specifically disclose information about human team members. Technology is available to monitor presence, attention or trust of humans. If this information is made available to artificial team members, they can adjust their behavior or way of interacting.
- **Explainable AI:** explainable AI (XAI) functions specifically disclose information from artificial team members to human operators. This typically concerns more advanced information than the SA type of functions. It allows human team members to understand the rationale for the system's decisions. For example, an analyst who receives recommendations from an algorithm needs to understand why the algorithm has recommended certain activity.
- **Working agreements:** with working agreements we refer to general behaviour policies that should be followed by team members. In literature the other terms are also used: (social) norms [10], contracts or policies [9]. The working agreements define the boundaries within which team members may act autonomously. They may specify the applicable permissions or obligations, and interaction protocols.

The HMT-modules that are designed in this phase can be drawn in a network configuration to specify how they interact and exchange information among themselves and to the user enriched with details about the information exchange.

3.3 Prototype Implementation

SAIL comes with a software framework in which the autonomous system(s) and HMT-modules can implemented. Through the SAIL GUI, a human-agent team can be configured by defining the active actors, modules and the lines of communication between them. The framework, therefore, allows for rapid testing of different HMT configurations. A set of generic HMT-modules can be activated or deactivated, and for case-specific circumstances new modules can be implemented. Communication between modules is done via sockets. Therefore modules and systems that are implemented in different programming languages can all be connected.

Ontologies can be loaded into the SAIL framework. Typically a version of the HATCL is added, plus other specific domain ontologies if they are needed. The modules that are included in the configuration publish which ontologies they accepts. Therefore the validity of the HMT configuration can be checked.

3.4 Test and Evaluate

The SAIL framework is developed for human-in-the-loop testing and evaluation in order to facilitate an iterative development process and longitudinal team development, as teamwork may change over time when both human and artificial team members are getting used to one another. The fourth phase in the SAIL method is meant to set-up user experiments with the prototype implementation of HMT-concepts. Typically multi-criteria analyses are needed to assess team performance and usability aspects.

4 Proof of Concept

To illustrate the use of the SAIL method and framework with a proof of concept (PoC) task, we implemented a simulated futuristic defense task. In the task, the human operator takes on the role of a marine officer on board of a stationary naval vessel. His mission is to protect the ship from incoming hostile drones together with a swarm of autonomous unmanned aerial vehicles (UAVs) while minimizing collateral damage. In principle, the UAVs autonomously locate themselves around the ship and independently detect, identify (as friend or foe) and eliminate hostiles. The UAVs have a varying identification accuracy. The operator can improve team performance by supervising the task and prioritizing targets. Furthermore, the operator has access to external information regarding identification of contacts, which (s)he can interpret, whereas the UAVs cannot. The team task has the following properties that justify HMT:

- The UAVs are able to do the task autonomously, but they make mistakes.
- The human operator can control and supervise the UAV's, however, due to workload the operator cannot do everything by hand.
- The operator has knowledge that the autonomous system does not have.
- Human-in-the-loop solutions are typically preferred in this type of defense scenario.

We used this task environment to create a setting where a user repeatedly cycles through a mission simulation and after-action-review, allowing team development by optimizing the set of working agreements. The set of available HMT-modules determines how the user can affect the swarm's behavior.

The PoC served a number of goals. First, we gained experience with implementing a task using the SAIL framework. Second, we tried out the task environment, and whether it was sufficiently rich, realistic and challenging for evaluating HMT-concepts. Third, we tested our first, rudimentary implementation of the set of HMT-modules.

4.1 PoC Problem Analysis

The task can be characterized by: an unpredictable and dynamic environment, high differentiation within the team as the user and UAVs have very different skills; short repeated work cycles, allowing study of team development; high time criticality; and a high risk situation.

The UAV is able to autonomously execute the force protection task. Its behavior model is a state machine with three states: *surveillance*, *identification* and *engagement*. When operating in the swarm the UAV's do not communicate with each other. The UAVs have a varying identification accuracy, which leads to mis-identification, and to the elimination of neutral contacts, which is interpreted as collateral damage. The UAVs have no capabilities to exchange information, and therefore, continuously act based on local knowledge, which lead to suboptimal coordination.

4.2 PoC HMT Design

The task allows studying all main teamwork function classes including situation awareness, explainable AI, and working agreements. Four HMT-modules were developed:

1. The *situation awareness* module provides the user with real-time awareness of the positions of the UAVs and of the sensor objects. Via this module the user can intervene by manually identifying sensor objects.
2. The *performance tracking* module provides insight in the identification performance of each UAV. This is a typical XAI function.
3. The *grouping module*. This module allows the user to define groups of UAVs based on their attributes. For example, a group *North* can be defined as being in the upper part of the area, or groups can be defined for better or worse performing UAVs.
4. The *working agreements* module. The user can set up working agreements. Each working agreement consists of three parts: the group that the agreement applies to, whether it concerns permission or prohibition, and third, the action.

The combination of the four modules allows new team coordination possibilities between the human user and the swarm of UAVs. For example, a user is able to define a group *North* and set up a working agreement that this group is prohibited to engage. Although the UAVs themselves have no concept of groups or geographic areas or working agreements, the modules provides an interface for the user to coordinate with the UAVs using these concepts. The grouping, performance tracking and working agreements modules provide an example of the interplay between HMT-modules: several may be combined to obtain a certain functionality. At the same time they are implemented separately, and are reusable for other domains or problems as well.

4.3 PoC Implementation

In order to be able to function in a human-agent team, the UAV is linked to the SAIL framework. The SAIL framework contains an initial version of HATCL that specifies the HATCL message format and domain ontology in plain old java (pojo) classes. All communication between HMT modules and the UAVs follows this ontology.

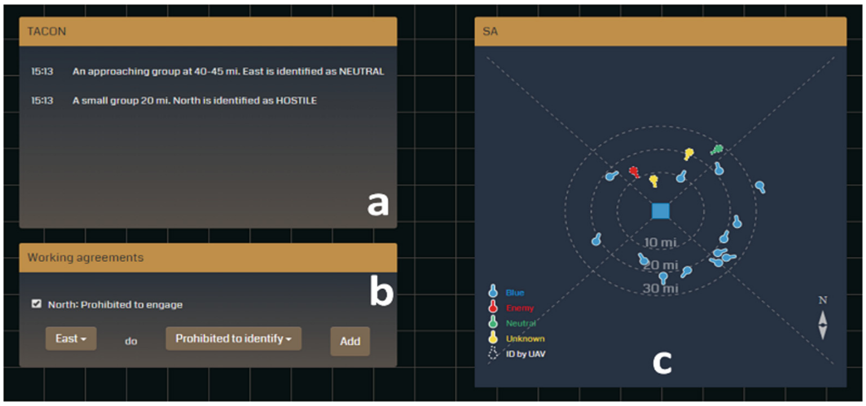
An interface with semantic anchors is defined that make the UAV interpret and produce HATCL messages. The following information elements are exchanged:

- *Attributes*: ID, location and direction. These are published continuously by the UAV
- *SensorObjects*: sensor objects within the UAV's sensor range are published continuously, including their identification
- *Actions*: the states of the behavior model refer to actions in the HATCL ontology

The interface defines how the UAV responds to HATCL messages. This is defined in the following way:

- *Prohibition*: a prohibition of an action blocks transitions to the corresponding state
- *Permission*: a permission of an action allows transitions to the corresponding state
- *Obligation*: an obligation enforces the transition to the corresponding state
- *Inform*: an inform message with the identification of a contact overrules the internal identification of the UAV

With these HATCL semantic anchors, the UAV can be connected to SAIL. Therewith the UAV can be integrated in a HMT, and will be equipped with social capabilities that allow team collaboration. The elements of communication are added in the domain ontology that is loaded in SAIL. The HMT-modules are implemented as independent scripts that respond to input messages in HATCL (Fig. 5).


TACON

15:13 An approaching group at 40-45 mi. East is identified as NEUTRAL

15:13 A small group 20 mi. North is identified as HOSTILE

Working agreements

☑ North: Prohibited to engage

East do Prohibited to identify

Fig. 5. The GUI of the task environment in the mission phase with external information messages (a) and with the HMT-modules: working agreements (b) and situation awareness (c)

4.4 PoC Evaluation

We set up a pilot experiment to evaluate different HMT concepts in the PoC task. Therefore, we have asked two naïve participants to perform the PoC task with four conditions: two levels of crowdedness of the scenario (many vs. few active UAVs in the task environment) and two HMT-configurations (sparse vs. rich HMT). In the sparse HMT condition they operated with only the *situation awareness* module between them and the UAVs. In the rich HMT condition they had all modules available, and therefore much richer interaction possibilities. The participants were naïve to the task environment and our goals. We measured task scores, specifically the number of hostile actors killed, neutral actors killed (collateral damage), and hits on the main vessel. Also, we asked the participants feedback using a semi-structured interview.

We evaluated the participants performance for the distinct HMT configurations and over multiple runs. Where an improved performance with a rich HMT was expected, this was not the case. First of all, the addition of more HMT modules resulted in worse instead of better performance (see Table 1). Also, the performance improvement with subsequent runs was negligible on all dependent measures (data not presented). Importantly, reduced performance was likely due to imperfections in the implementation of our working agreements module which resulted in unexpected behavior.

Table 1. Average scores in the different experiment settings. Each participant performed three runs of the first three conditions, and two runs of the fourth condition

	Enemies hit	Neutrals hit	Vessel hits
1 Crowded, sparse HMT	59	7	5
2 Crowded, rich HMT	38	4	13
3 Not-crowded, sparse HMT	27	7	2
4 Not-crowded, rich HMT	21	8	6

The results from the participant feedback were very insightful. A first insight is that participants reported that the PoC task and scenario were realistic and sufficiently challenging. The participants did not feel completely in control but partially dependent on the autonomous system, which is exactly the type of HMT situation we wanted to study. Furthermore, it offers broad opportunities for future extension, enabling the development and study of more sophisticated HMT-concepts like trust calibration.

Secondly, participants reported that they wished that the GUI was more refined, for example, showing UAV performance during runtime. Third, participants noted that, on the one hand, they would prefer to have more control over the behavior of the UAVs and that, on the other hand, their cognitive task load was high. This is an interesting tension field in designing an HMT-layer: providing more control options will likely increase task load even more. Fourth, participants noted that they were unsatisfied with system performance when certain working agreements were activated. Specifically, when a group of UAVs was prohibited to identify, the other UAVs did not take this into account when locating themselves. This is an example of a need for system explanation; the user expected more than the system was capable of.

The overall goal of the pilot experiment with the PoC task was to evaluate the usability of the SAIL framework and method to develop and experiment with HMT concepts. We conclude it has shown to be successful for this purpose.

5 Conclusion

We have presented SAIL, a design method and framework for the development of HMT-concepts. The SAIL method integrates human factors and autonomous systems aspects in one design method that is supported by a software framework for implementing and testing HMT-concepts. Starting point of SAIL is that an HMT can be developed without changing the internal capabilities of the autonomous system. The method consists of a iterative development cycle of four steps: problem analysis, HMT-design, prototype implementation and evaluation.

The SAIL framework consists of a modular social layer between autonomous systems and human team members, in which all social capabilities can be implemented that enable teamwork. We have argued that these social capabilities are to some extent generic. Examples are functions for situation awareness, human awareness, explainable AI, working agreements and tasking. Within SAIL, HMT-modules are developed that construct these social capabilities. The modules are reusable in multiple domains.

We have evaluated the SAIL method and framework using a proof of concept task environment. In general we conclude that more complex HMT-systems, the iterative development and evaluation cycle requires a careful approach, as imperfections in the system in early cycles make it hard to properly evaluate HMT-concepts. Nevertheless, our simplified, rudimentary task environment was sufficiently challenging to set up and evaluate a first HMT-concept. It led to valuable feedback and opened up opportunities for more sophisticated HMT-concepts, informing design and implementation early in the development process. Therefore, we believe that the SAIL method and framework have shown to be a promising approach to design, implement and evaluate HMT-concepts. Using SAIL, social requirements for autonomous systems may be developed, that in the end can be adopted in the development of autonomous systems.

References

1. Vinciarelli, A., Pantic, M., Heylen, D., Pelachaud, C., Poggi, I., D'Errico, F., Schroeder, M.: Bridging the gap between social animal and unsocial machine: a survey of social signal processing. IEEE Trans. Affect. Comput. **3**(1), 69–87 (2012)
2. Roff, H., Moyes, R.: Meaningful Human Control, Artificial Intelligence and Autonomous Weapons. In: Briefing Paper Prepared for the Informal Meeting of Experts on Lethal Autonomous Weapons Systems, UN Convention on Conventional Weapons, April 2016
3. Parasuraman, R., Barnes, M., Cosenzo, K., Mulgund, S.: Adaptive automation for human-robot teaming in future command and control systems. Int. J. Command Control **1** (2), 43–68 (2007)
4. van Diggelen, J., Looije, R., van der Waa, J., Neerincx, M.: Human robot team development: an operational and technical perspective. In: Chen, J. (ed.) Advances in Human Factors in Robots and Unmanned Systems. AISC, vol. 595, pp. 293–302. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-60384-1_28
5. Johnson, M., Bradshaw, J.M., Feltovich, P.J., Van Riemsdijk, M.B., Jonker, C.M., Sierhuis, M.: Coactive design: designing support for interdependence in joint activity. J. Hum.-Robot Interact. **3**(1), 2014 (2014)

6. Langley, P., Meadows, B., Sridharan, M., Choi, D.: Explainable agency for intelligent autonomous systems. In: AAAI, pp. 4762–4764, February 2017
7. Lee, J.D., See, K.A.: Trust in automation: designing for appropriate reliance. Hum. Factors **46**(1), 50–80 (2004)
8. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Hum. Factors **37**(1), 32–64 (1995)
9. Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkarni, S., Lott, J.: KAoS policy and domain services: toward a description-logic approach to policy representation, deconfliction, and enforcement. In: 2003 Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks, POLICY 2003, pp. 93–96. IEEE (2003)
10. Dignum, M.V.: A model for organizational interaction: based on agents, founded in logic. SIKS (2004)
11. Fortino, G., Garro, A., Russo, W.: An integrated approach for the development and validation of multi-agent systems. Int. J. Comput. Syst. Sci. Eng. **20**(4), 259–271 (2005)
12. Hayes, B., Shah, J.A.: Improving robot controller transparency through autonomous policy explanation. In: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, pp. 303–312. ACM, March 2017
13. Van Lent, M., Fisher, W., Mancuso, M.: An explainable artificial intelligence system for small-unit tactical behavior. In: 2004 Proceedings of the National Conference on Artificial Intelligence, pp. 900–907. AAAI Press, July 2004
14. Labrou, Y., Finin, T., Peng, Y.: Agent communication languages: The current landscape. IEEE Intell. Syst. Their Appl. **14**(2), 45–52 (1999)
15. FIPA: FIPA ACL message structure specification. Foundation for Intelligent Physical Agents (2002). http://www.fipa.org/specs/fipa00061/index.html. Accessed December 2017
16. Peeters, M.M., Bosch, K.V.D., Neerincx, M.A., Meyer, J.J.C.: An ontology for automated scenario–based training. Int. J. Technol. Enhanc. Learn. **6**(3), 195–211 (2014)