Memorandum **TNO PUBLIEK**

Kampweg 55
3769 DE Soesterberg
P.O. Box 23
3769 ZG Soesterberg
The Netherlands

www.tno.nl

T +31 88 866 15 00
F +31 34 635 39 77

To
Jurriaan van Diggelen

From
T Haije, JS van der Waa, K Veltman

Subject
TNO 2021 M12516: Connecting AI and HMT innovations: SAIL

Date
22 December 2021

Our reference
TNO 2021 M12516

Contact
T Haije

Direct dialling
+31888661155

1 Introduction

As technology progresses, it integrates continuously with our society in the form of intelligent systems. As these systems become faster, smarter, and more capable, their tasks also shift from simple tasks such as vacuum cleaning, to more important and high-risk tasks as often seen in the aviation, healthcare, and defense sectors. Successful intelligent systems often have some level of autonomy for completing complex tasks, such that they can exhibit goal-directed behaviour while adapting to the dynamic world we live in.

A recurring pattern is that there is always a need at some point during task execution where the autonomous system needs to interact with a human. Furthermore, a team is required for any task of sufficient complexity or criticality that requires greater knowledge, skill, ability, or redundancy than a single operator can provide in the available timeframe (Schneider et al., 2021). For instance, safety-critical tasks require resilience that may not be guaranteed by brittle capabilities of autonomous systems in the face of dynamic events. In that case human adaptability and ability to understand context is crucial. Another important reason for human-machine teams is to prevent a responsibility gap caused by consequences of an autonomous system their actions that acted without a human in the loop.

The human-machine teaming (HMT) paradigm takes this a step further and describes the autonomous system not as a tool, but as a team member. And for teamwork, communication is key. In fact, many of the other required capabilities for teamwork have a strong social factor as well (Lyons & Havig, 2014). For instance, optimal teamwork requires the team members to pro-actively share information and uphold the situational awareness of the other members, as well as being able to explain their actions. For each of these capabilities, communication serves as a basis (Wynne & Lyons, 2018). For human-machine teams this is a major challenge, as humans and autonomous systems have their own internal representations which may not be understandable for the other.

As such, a primary technical challenge is how to facilitate the communication between humans and autonomous systems through a kind of middleware that enables and supports effective collaboration on their joint task. In addition, it should be a solution that can serve as a basis for other social capabilities that are needed to make the autonomous system (or human) a good team member. Finally, overcoming this technical challenge would enable humans to collaborate with a range of autonomous systems types, ranging from complex and extensive systems to simpler AI systems with a more narrow intelligence, as the social building blocks for teamwork can be provided by SAIL.

This memo describes a solution to this problem in the form of the Social Artificial Intelligence Layer (SAIL). At its core, SAIL acts as a central translation server that reconciles human mental models and those of autonomous systems. Using SAIL, the human can convey their intentions in a natural way that are translated to machine comprehensible information via so-called semantic anchors, as well as vice versa where information from the autonomous system is translated to human comprehensible information.

The newly developed and implemented version of SAIL, SAIL v2.0 henceforth annotated as just SAIL, builds on prior TNO-research that led to the development of the concept of SAIL (van der Vecht et al., 2018). SAIL acts as *middleware*[1], sometimes called "software glue", between human and autonomous systems, supporting the development of social intelligence that contain the social capabilities needed for successful human-machine teaming. Compared to the original SAIL, the new iteration as presented here has been built from the ground up in-house using a bottom-up approach, based on several lessons learned with the original SAIL. The SAIL translation functionality acts as the foundation for other *social intelligence modules,* as to enable the development of use case specific or if possible generic social intelligence to facilitate human-machine teaming. This differed from the original version where SAIL was not so much a middleware, but instead acted as a container for interdependent innovations aimed to facilitate human-machine teaming. SAIL v2.0 can be much more compared to other client-server frameworks, but specifically developed to support human-machine teaming.

In section 2, this memo elaborates on the background for the need for communication between human and autonomous systems as part of a team, AI-middleware as a solution, and alternatives used for comparable problems. The current state of SAIL as developed this year, incorporating the information management, ontologies and semantic anchors that enable translation between humans and autonomous systems are described in detail in section 3. Open challenges identified in the past year are described in section 4, followed by a discussion in section 5 and a conclusion in section 6.

2 Background

Effective human-machine teaming requires a multitude of capabilities from both humans and autonomous systems, such as observability, predictability and directability (OPD) (Johnson et al., 2014; Mcdermott et al., 2018). Although there is a plethora of informative literature on what capabilities are required for HMT, there are little to no documented system architecture solutions that support software implementations of these capabilities. Research is often performed in isolation where a specific HMT capability is implemented to suit the needs for a specific task and human-machine team, resulting in highly specific (software) solutions that are difficult to reuse outside the original use case.

A solution to this issue exists in software engineering in the form of middleware. Middleware "provides common services and capabilities to applications outside of what's offered by the operating system"[2]. Translated to HMT terms, middleware can be defined as providing common services and capabilities to applications incorporating (social) capabilities that the autonomous system or user interface do not provide. By providing common services and capabilities such as protocols, structure and content for communication, HMT researchers can focus on creating applications that implement the required HMT capabilities for successful teaming. In addition, new applications can be added to the middleware, such that they can be reused and applied in other contexts as well.

In its basis, collaboration between humans and autonomous systems requires knowledge on human factors as well as on autonomy and multi-agent systems (MAS). Although many system architectures, including middleware solutions, are available for both research fields, they ultimately fall short when applied to HMT as they do not take the needs of both human-machine teaming and autonomous systems into account (van der Vecht et al., 2018).
SAIL aims to fill this gap, by presenting a middleware solution for HMT that provides as fundamental common service the translation of intent and information between human and autonomous systems.

[1] https://www.redhat.com/en/topics/middleware/what-is-middleware
[2] https://www.redhat.com/en/topics/middleware/what-is-middleware

Doing so, it can streamline development and thus research into HMT. By abstracting some of the complexity, new software that connects autonomous systems and humans in new ways can be made more rapidly, robustly, and with more impact by improving reusability of created software.

A limiting factor of middleware in general is that it often has an impact on system performance, compared to a system developed for the specific needs of a particular use case. In addition, in middleware that encompasses many applications the complexity might increase as well, especially when troubleshooting. Finally, middleware such as SAIL still requires changing the autonomous system in some manner (integrating semantic anchors) and requires a human to specify a proper translation table usable for communication between team members. A final uncertainty is whether it is possible for middleware to provide useful services and capabilities relevant to HMT that are truly generic and useful over many different HMT use cases.

3 The Social AI Layer; SAIL

The latest version of SAIL is an evolution on prior research but shares many similarities with the original concept as presented in (van der Vecht et al., 2018). The goal of SAIL remains the same in that it strives to provide a software framework that acts as a social layer between autonomous systems and humans such that they can optimally perform as a human-machine team. A difference is that SAIL this year has been re-implemented in-house in the Python programming language with a more operationalizable focus, as the codebase of the previous SAIL was not maintainable anymore.

As described by (van der Vecht et al., 2018), “a one-size fits all solution for HMT does not exist, as the types of tasks, humans and autonomous systems is different per context” (p. 3). Although the statement concerned an entire HMT solution that is generic, it is also valid for the creation of generic social AI modules itself, as another lesson learned from the original SAIL was that making every social AI module generic from the onset is extremely difficult and leads to long development times. As such, the new SAIL takes a bottom-up approach to create a lean, useful framework with a limited well-worked out set of generic services and functions. As a result, SAIL now offers a generic method for communication between autonomous systems and humans by providing a communication language, ontology, and Application Programming Interface (API)[3] that have been integrated with generic delegation concepts from human-machine teaming (van Diggelen et al., 2021). Furthermore, other use cases aside from delegation can be supported as well by adding to the ontology and alter the translation table of SAIL to fit specific needs.

The targeted domains for SAIL are military, national security, and other users that plan to obtain autonomous systems and use them in a human-machine team. SAIL can help in providing the substrate for teamwork between all actors in the team. These target groups profit from a lean bottom-up approach with little development time to a working human-machine team. Generic functionality can afterwards be extracted and added to SAIL on a function-by-function basis.

[3] API - Wikipedia - https://en.wikipedia.org/wiki/API

This approach is similar to how Team Design Patterns are utilized within projects, and sometimes generalized to become part of the TNO-wide Design Pattern Library.

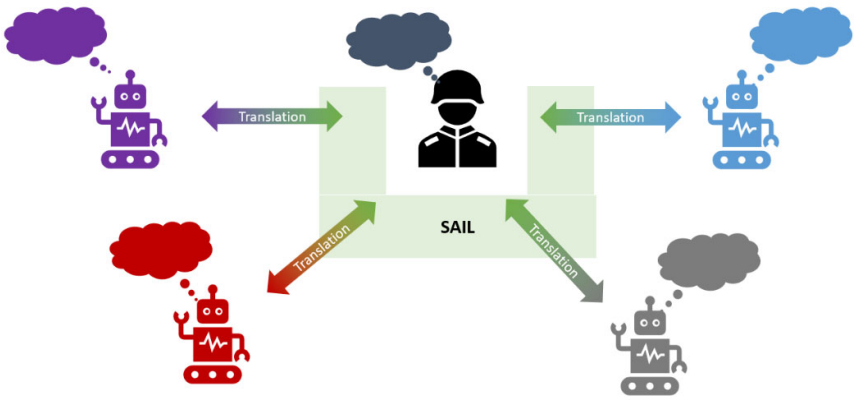

Figure 1 Overview of the primary SAIL v2.0 functionality. Which is supporting the communication and translation of relevant concepts (e.g., tasks, notifications, user profiles, goals, context information, etc.) between humans and various autonomous systems.

The main functionality of SAIL v2.0 as described above is visualized in Figure 1.
The following subsections describe the most important of those capabilities which form the current implementation of SAIL: the capability for humans and autonomous systems communicate in a shared language supported by an ontology targeted to human-machine teaming applications and are linked to the internals of an autonomous system in the form of semantic anchors. In section 4, more attention is given to potential future improvements of SAIL's translation server, as well as potential new social capabilities in SAIL.

3.1 SAIL Function; Communication

For an HMT to function, communication between all team members should be facilitated, whether human or machine. SAIL supports this by acting as a centralized mediator for this communication. It supports the connections of various software components such as the human interface or the API of an autonomous system. SAIL assumes a single connection per autonomous agent in the team towards the SAIL server. Think of agents such as an autonomous UGV, a swarm of drones, or a virtual cyber-agent. Aside from this, SAIL assumes a single connection per user interface, such as a mission planner, a system dashboard, or a handheld device. Which could imply that multiple humans share a connection to SAIL if they have the same role and use the same interface. For example, patrolling soldiers with the same handheld acting as their interface towards their drones. On the other hand, a single human might have multiple connections as well. For example, a commander having access to the same handheld as the patrolling soldiers but also access to a mission planner interface. This notion is depicted in Figure 2 below.

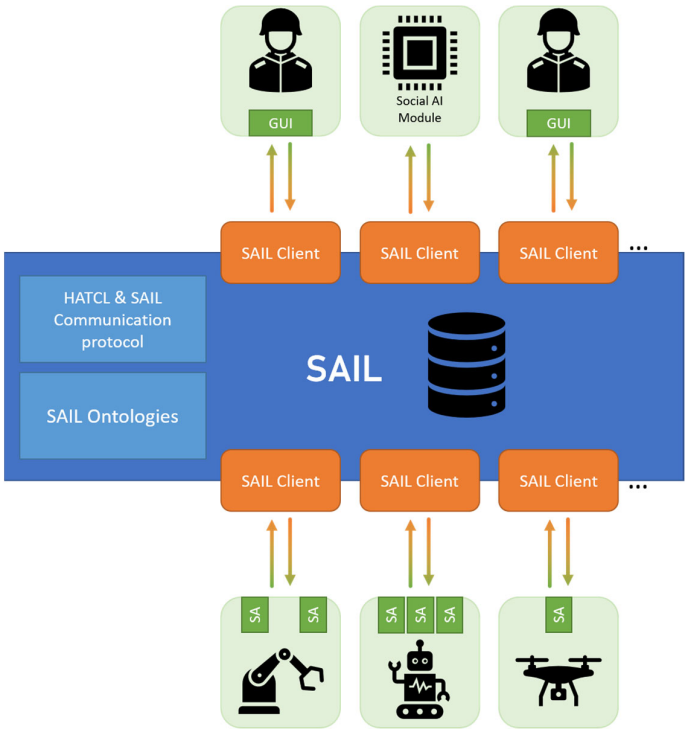

Figure 2 An illustration of SAIL. In particular this depicts how several different SAIL Clients can be connected to different autonomous systems but also various human team members. This allows SAIL to support various user interfaces. It even allows additional components or modules processing and adding to the data within SAIL, which other clients can access.

Communication between humans and autonomous systems is supported by SAIL with a communication protocol, describing rules for the communication flow and data format of messages. The SAIL communication protocol is based on the HATCL standard (J. van Diggelen, T. Mannucci, M.M.M. Peeters, B. van der Vecht, 2019). HATCL stands for “Human-Agent Teaming Communication language”. This protocol consists of several types of communication messages;

- **Query:** The communication act to ask for all known information about a shared concept (e.g., a detected person, an online information source or a sensor output). This act is initiated by a team member looking for information; it is a pull for information.
- **Inform**: Similar to Query, but different in that this communication act is initiated by a team member having information that is deemed valuable for another agent; it is a push of information.
- **Request**: The communication act to ask for a team member to take a certain action (e.g., to move to a certain location, to search through an online information source or to activate a certain Play).
- **Subscribe**: This is a request for receiving any new information acquired by a team member on a specific concept. A persistent query of sorts. This act is initiated by one team member, expecting the receiving member to keep the sender updated.

- **Propose**: The communication act where a team member proposes to alter known information or for the receiving member to commit itself to uphold a certain agreement (e.g., to send notifications when detecting people or to ensure it is at a certain position in time).
- **Accept / Reject**: Both are responses to a Propose message, respectively signifying that the receiver will or will not commit to the alteration in information or the proposal to commit to an agreement.
- **Understood / Not-understood**: Both are responses to an Inform message, respectively signifying that the receiver has successfully processed the information or not. The Not-Understood message is added to support traceability as well as to offer team members a chance for an alternative.
- **Cancel**: The communication act to signify a previous message should be ignored as of receiving this (e.g., to stop sending notifications, to cancel a subscribe, or an action or Play).

Three protocols are defined by HATCL based on these messages; the Inform / Cancel protocol, the Query protocol, and the Propose / Subscribe protocol.

- **Inform / Cancel Protocol**: This describes that any Inform or Cancel message should be followed by either an Understood or Not-Understood message (see Figure 3). SAIL supports this protocol by automatically sending a Not-Understood on a timeout (e.g., an autonomous system cannot be reached), or when the message content is not part of the shared ontology (e.g., the concept of cognitive workload is not a shared concept within the team).

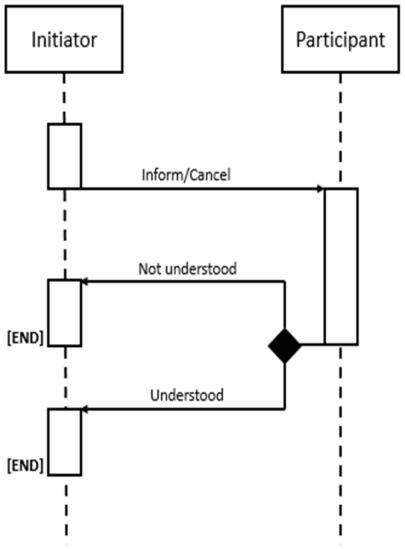

Figure 3 The overview of the Inform / Cancel Protocol. This is in large part automated within SAIL based on a timeout setting and reasoning over the shared ontologies (which depict what each team member can understand).

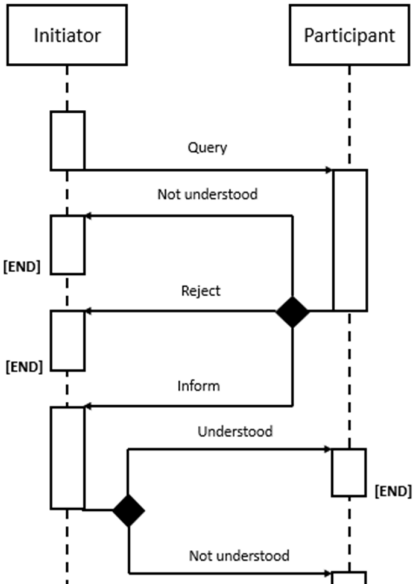

Figure 4 The overview of the Query protocol. The protocol's Not-Understood and Reject messages are partly automated by SAIL based on timeout settings and reasoning over the shared ontologies.

- **Query Protocol**: This protocol describes how should be responded to a Query message (see Figure 4). On receiving a Query message, it can be replied to with a Not-Understood, Reject or Inform message. If the latter occurs, the Inform / Cancel Protocol is initiated. The Not-Understood messages are automated by SAIL similarly as in the Inform / Cancel Protocol; when a timeout is reached or when the ontology cannot resolve the necessary concept translation.
- **Propose / Subscribe Protocol**: This protocol mandates that a Propose or Subscribe should be followed by either a Not-Understood, Reject or Accept message (see Figure 5). The Not-Understood message is automated by SAIL when communication time-outs or the message content cannot be translated by the ontologies. Ideally but currently not implemented yet, in the case of a Propose message the receiver can follow-up on a Reject with a counter Propose message of its own. This would then also be tracked by SAIL as a negotiation between team members for future references.

With the adaptation of HATCL, the SAIL messages are defined as well as structured by protocols. The instantiations of the ontologies described in the next section (see section 3.2) define the potential content of these messages. This, combined with SAIL's own reasoning on these ontologies, allows for the automation of parts of the protocols that can be tedious to track as an autonomous system or HMT developer. In addition, this automation also reduces the amount of actual communication over all of SAIL's connections. For example, a Query that cannot be translated into understandable concepts for the receiver causes SAIL to immediately send back a Not-Understood message. The Query does not need to be sent, cutting back on the required communications by half.

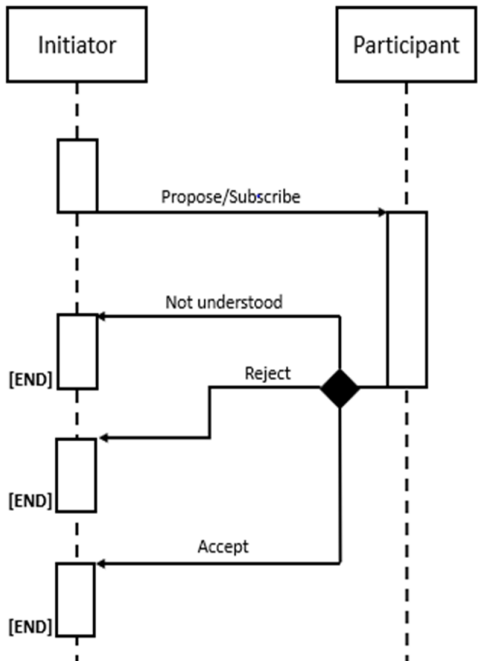

Figure 5 The overview of the Propose / Subscribe protocol from HATCL. The Not-Understood message is automated by SAIL in based on either the timeout settings or when concepts within the Propose or Subscribe cannot be translated.

There are two disadvantages of this communication functionality of SAIL. The first revolving around SAIL functioning as a centralized server. This makes SAIL more brittle compared to a decentralized approach. Redundant servers need to be made available in case the original server fails. However, this might harm data integrity and completeness. Instead, a more decentralized approach where each software component within the team can act as a server would ensure that as long as there exist team members within range of each other, their communication and collaboration will be supported.

The second disadvantage of this communication approach is that only parts of the HATCL protocol can be automated with the use of SAIL's ontologies. For instance, automation of a proposal acceptance is difficult, as the only way to know for certain that a proposal was accepted is by an explicit message from that partner. However, the current reasoning for the Not-Understood messages can also be extended to Reject messages. This would entail the inclusion of each team member's current capabilities into SAIL's ontology. Subsequently, this allows SAIL to determine whether a proposal is feasible given the intended receiver. This is similar to current research on an autonomous system's reasoning about its own capabilities (e.g., see the SNOW Project[4]), extended to the team-level.

3.2 SAIL Function; Ontological translation

To let SAIL function as middleware between the human operator and the autonomous systems, SAIL needs to be able to understand the intentions and meanings of both sides. SAIL does this by using an ontology and associated database structure to translate concepts. The need for this concept translation comes forth from the observation that terminology that might be relevant or applicable to an autonomous system may differ from the natural way a human usually communicates. An example that illustrates this principle is that humans can communicate that they want to be informed when a person wearing a specific logo is found. For the autonomous system this would mean that the human wants to be informed when there is a *detection* of a *human* in combination with that specific *logo*. Another example could be on vehicle level: an autonomous system has a lot of parameters that a human might not want to consider when delegating a task. The human might say that they want to search the street safely, without specifying what that means. The autonomous system may not have a notion of 'safely', but might have a setting that specifies that it needs to keep at least a 1-meter distance of any vehicles. SAIL can make such translations.

[4] https://appl-ai-tno.nl/flagships/snow/https://appl-ai-tno.nl/flagships/snow/

The SAIL ontology and data structures are based on several earlier developed ontologies. The CD-ROM ontology which describes out of which components an autonomous system is built and which capabilities these components (and therefore the autonomous system) can realize (Joris Sijs, 2022). The Tasking Ontology, which describes how tasks can be decomposed in subtasks and how these relate to capabilities (Vught & Veltman, 2021). For the delegation element of SAIL, a Play ontology was developed, which is briefly described below (van Diggelen et al., 2021).

3.2.1 Play ontology

The main structure of the play ontology can be seen in Figure 6, this play ontology is partly based on the Task Ontology (Vught & Veltman, 2021). In this Figure 6, rectangles represent entities or classes, diamonds represent the relation between certain entities and oval represent attributes or characteristics of an entity.

The **Goal** describes the overall intention of the current mission/assignment. A **Goal** can be realized by a **Play**. A **Play** describes a manner of execution that is relevant/wished for. So, in a simplistic version, a **Play** describes how a certain goal or task can be executed. Therefore, a **Play** decomposes into one or multiple **PlayTask(s)** that specify the desired course of action in more detail. Note that these **PlayTask(s)** can be decomposed further (into sub PlayTask(s)). Every decomposition specifies the course of action in more detail and could even go to the level where very little autonomy or intelligence is demanded from the autonomous vehicle. Note that while this is possible, this is not necessarily desirable as it would ignore any intelligence within the autonomous system.

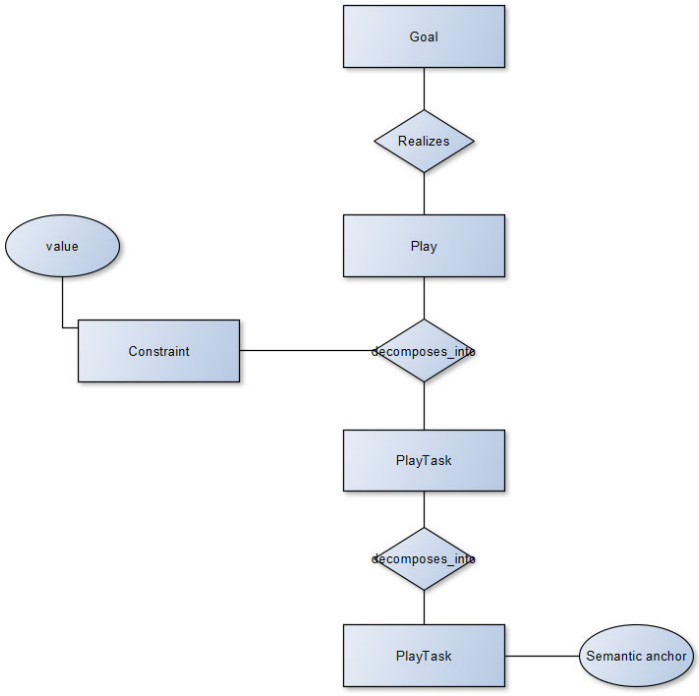

Figure 6 Main elements of the play ontology.

On the lowest level of decomposition, the **PlayTask** has a semantic anchor, which can link to an **AgentTask**, a task that is known and can be executed by the autonomous system. Again, the level where this semantic anchor lives is dependent on the autonomy and intelligence of the autonomous system.

It is important to note that a **Play** does not only describe how a high(er) level intention can be decomposed into lower-level intentions, it can also express **Constraints**. **Constraints** can be used to limit and guide the freedom of the autonomous system to ensure that the system behaves in the desired fashion.

3.2.2 Communication protocol

Since SAIL is a piece of software that facilitates the communication of information, a data structure was developed to build this communication on. In this structure several types of messages were defined as well as the desired content of those messages. The message types and the way they are supposed to be used is based on HATCL (J. van Diggelen, T. Mannucci, M.M.M. Peeters, B. van der Vecht, 2019). Every type of message (see section 3.1: SAIL Function; Communication) has a specific data format that fits the message type. For example, a DetectionInform message *always* contains a label, a confidence, an image, and a detection ID. By (fully) specifying the information a message contains, the developer/scientist always knows what to expect and what information is delivered.

The defined message types and their structure are created in such a way that they fit with HATCL (J. van Diggelen, T. Mannucci, M.M.M. Peeters, B. van der Vecht, 2019), CD-ROM (Joris Sijs, 2022), and the PlayOntology (Vught & Veltman, 2021).

Advantages of using these fully specified information objects and ontology lie mostly in understanding and transparency. It is much easier for the developer as all possible information that can be received is specified, as well as what information is to be sent in return. The ontology facilitates the translation between autonomous system and operator, by relating human and machine concepts. A potential downside is that the developer has less freedom to declare anything they want. If, for example, the developer would like to add the position of the detection, they need to request that information to be added to the DetectionInform message, or another message needs to be defined. To somewhat overcome this issue, a special and more relaxed message type was introduced. This message queries the knowledge of the autonomous system with the information not specified in the message type in Python's pattern-matched *keyword arguments.* If the information is not present in the knowledge base the autonomous system ignores it.

3.3 SAIL Function; Semantic anchors

A human-machine team will contain several types of agents from various manufacturers. For the military, police and other domains it is not feasible to enforce each commercially available agent to be compatible with SAIL, nor to develop such agents on their own. Instead, SAIL aims to require minimal changes to these agents, whether they are autonomous systems or cyber agents. Ideally no chances, if the agents provide a sufficiently detailed API. This is supported through the concept of Semantic Anchors. They act as a connecting layer between an agent's provided functionalities and SAIL, similar as an API but in a more ad-hoc fashion.

A Semantic Anchors plays the role of utilizing its SAIL connection to share its internal state and acquired data with the SAIL database for other team members to query or subscribe to (e.g., location, detections, etc.). In addition, they can receive SAIL messages and if a message and its content is supported make the additional changes in the agent's internal state (e.g., to prioritize a certain task, to go towards a waypoint, etc.).

The concept of Semantic Anchors allows Defense to discuss and specify the functionalities and capabilities with a manufacturer to support the envisioned human-machine team, but without requiring full access to the agent's internals. It differs in this from an API in the sense that a Semantic Anchor can be specifically tailored to an envisioned usage. It is likely that a Semantic Anchor will make use of an agent's API if available. Another added benefit of Semantic Anchors is that they tend to be independent of each other. Thus, adding more anchors to an agent only expands on its capabilities to collaborate with its team partners.
Within SAIL we differentiate between various categories of Semantic Anchors;

- **Information sharing anchors**: These anchors allow SAIL to access information from the system as well as add or adjust information known to the system.
- **Delegation anchors**: These anchors allow SAIL to assign tasks, goals, plays, constraints and similar concepts to the system.

3.4 Current implementation

The current implementation of the SAIL middleware was developed in python 3.6+. A client-server model[5] was used. For this a ZEROMQ[6] implementation was used with a REQ-REP pattern[7]. This pattern can be described as follows: the client initiates the communication by sending a request to the server. The server awaits incoming requests. When the server receives a request, it formulates an answer belonging to that request (see Figure 7).
For the communication of information several classes were identified in accordance with HATCL. We differentiate between Responses and Information pushes/requests. A Response is always formulated in response to another message. Several types of responses (sub classes) were created, the most used are the Accept (understood and OK) and the NotUnderstood (could not handle the request).

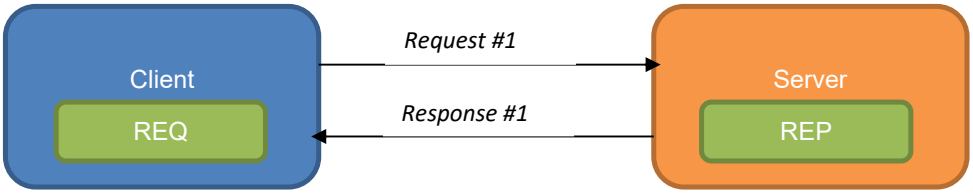

Figure 7 Example representation of the client-server model with a REQ-REP pattern.

[5] https://en.wikipedia.org/wiki/Client%E2%80%93server_model
[6] https://zeromq.org/
[7] https://zguide.zeromq.org/docs/chapter3/

To further specify the meaning of these classes, sub classes we created with a clear usage and definition. Below is an overview of these message types (note that these classes were highly inspired by HATCL, see section 3.1, and that this list reflects what is currently implemented, not what is possible):

1. ***Inform***: a data push, when one actor wants to provide another actor with a piece of information. E.g., a robot may formulate an Inform request when it has detected a human being.
    a. ***DetectionInform***: Inform message for a visual detection, containing information about the label of the detection, the confidence, and an image.
    b. ***BatteryStatusInform***: Inform message of the battery status of an autonomous system.
    c. ***LocationInform:*** Inform message of the location of a team member.
    d. ***ImageInform***: Inform message to send an image to the receiver, no detection required.
    e. ***PointCloudInform***: Inform message used to send information about the point cloud of the autonomous system, can be used to inform the receiver about the perceived environment of the actor.
    f. ***SpeechRecognitionInform***: Inform message containing the result of speech recognition, the detected language, the words detected and the confidence of the algorithm.
    g. ***HeatMapInform***: Inform message containing heatmap information, can be used to provide insight into where an autonomous system suspects activity or locations it has recently visted.
    h. ***ConnectionStrengthInform***: Inform message containing the strength of the connection with, for example SAIL or the command-and-control center.
    i. ***PlayInform***: Inform message indicating the currently active play.
    j. ***GoalInform***: Inform message indicating the currently planned for goal.
    k. ***TaskInform***: Inform message indicating the current task of a team member.
    l. ***ActionInform***: Inform message indicating the current action of a team member.
    m. ***PlanInform***: Inform message indicating the current plan of a team member
    n. ***IntroductionInform***: Inform message to announce the availability of a team member for the current mission.
    o. ***ModeInform***: Inform message indicating the current mode of the team member. Can indicate whether a mission is executed in silent or careful mode for example. This has effect on the way certain tasks or actions are executed.
2. ***Proposes***: a request where one actor proposes a change of specific information to another actor. E.g., the human operator may propose to the autonomous system that the label of the detection is not human, but doll.
3. ***Queries***: a request where one actor queries to another for information. E.g., the human operator might query the current location of the autonomous system.
    a. ***DetectionQuery***: Query message to ask a team member for information about a specific detection.
    b. ***BatteryStatusQuery***: Query message to ask an autonomous system about its current battery state.
    c. **LocationQuery**: Query message to ask a team member about its current location.
    d. ***ConnectonStrengthQuery***: Query message to ask a team member about its connection strength.
    e. ***PlayQuery***: Query message to ask a team member what its current play is.
    f. ***GoalQuery***: Query message to ask a team member what its current goal is.
    g. ***TaskQuery***: Query message to ask a team member what its current task is.
    h. ***ActionQuery***: Query message to ask a team member what its current action is.
    i. ***PlanQuery***: Query message to ask a team member what its current plan is.
    j. ***ModeQuery***: Query message to ask a team member what its current mode is.

4. ***Request***: a request where one actor asks another to perform a certain action, task, or play. These requests focus a bit more on delegation. E.g. The human might ask the autonomous system to start the play “Monitor human being.”
    a. ***PlayRequest***: Request message to start a certain play.
    b. ***TaskRequest***: Request message to perform a certain task.
    c. ***ActionRequest***: Request message to perform a certain action.
    d. ***ModeRequest***: Request message to execute certain tasks and actions in the specified mode.
5. ***Subscribe***: a request for information, to be informed as soon as anything specific happens. E.g., the human might subscribe to all detections the autonomous system does, to be immediately aware that a detection occurred without specifically asking for detections at a certain moment in time.
    a. ***DetectionSubscribe***: Subscribe message indicating that the sender wants to be informed of all detections (of a specific team member).
    b. ***LocationSubscribe***: Subscribe message indicating that the sender wants to be informed of all location updates (of a specific team member).

As a rule of thumb for HMT interaction, one SAIL server is implemented and two clients: one for the autonomous system side, and one for the human side. In correspondence with the server-client model, the clients initialize communication with the SAIL server. The client sends a message to the SAIL server, the SAIL server interprets this message and translates it (if needed) before sending it to the human or autonomous systems side. Below is an example:
In the case the autonomous system has detected a terrorist, it formulates a DetectionInform message with as receiver the human. The message is sent to the SAIL server, the server translates this message so the human can understand the message in a language that comes naturally. The translated message is sent to the human and the human can now see the detection done by the autonomous system and delegate the autonomous system (if needed) based on this information (see Figure 8).

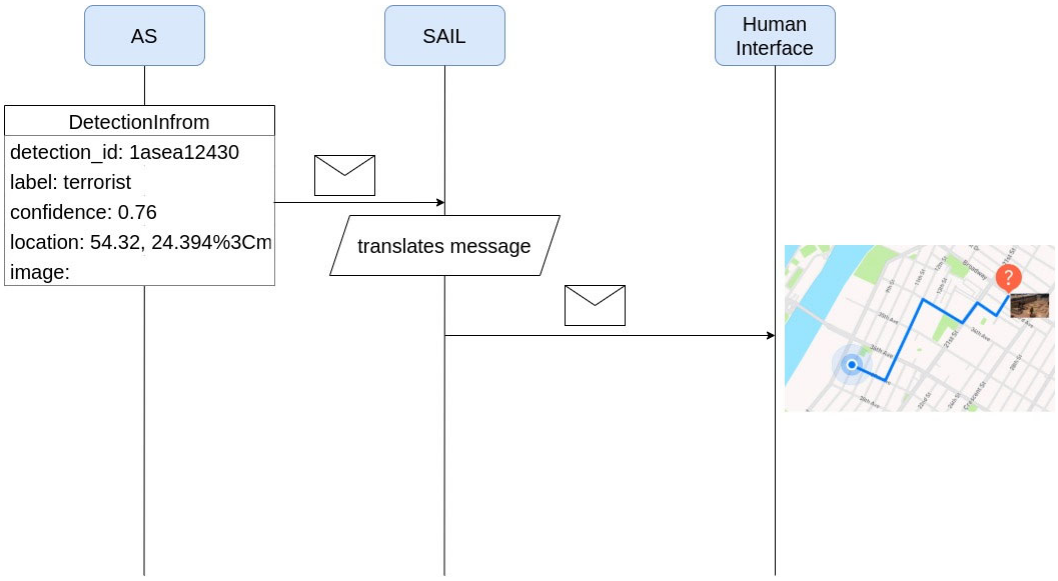

Figure 8 Example of the information flow for a DetectionInform that is formulated by an autonomous system, is translated by SAIL and sent to the human client that visualizes the information in a natural way.

The other way around works the same, the human can send a PlayRequest (a message to ask the autonomous system to start a certain play). This message is sent to the SAIL server and translated so the autonomous system knows how to handle the request. The translated message is sent to the autonomous system, who interprets the message and handles it accordingly.

The idea of SAIL was to make it agnostic, in the sense that it is independent of the human operator or autonomous system, while being specific in required inputs and expected output. This means that the information objects, the messages, and their content are fully specified. The handling of a received message is dependent on the actor; therefore, handlers of specific messages can be defined during the initialization phase.

If for some reason an actor cannot formulate a response to a request, a timeout was added to all requests. Once the time limit of the timeout is reached, SAIL returns that a timeout was received, while continuing with any other open requests. This ensures that SAIL does not lock on a request that cannot be answered.

4 Open challenges

Not all functions are present in SAIL, many are still envisioned. Here we introduce several still open challenges and envisioned functionalities and offered services.

4.1 Envisioned Function; Traceability

When working with autonomy, it is vital that when needed you can understand what led to a certain decision the autonomous system makes, this is called traceability. When an autonomous system decides to start its monitor play, the human operator may want to gain insight into what triggered this decision. This can be done by monitoring the detections of the autonomous system, or by tracking the communication. Especially with multiple actors, it might be interesting to see where the event chain originated.

SAIL can help support traceability by persistently storing the communicated concepts and known information. As SAIL works as middleware, it receives all communicated concepts and information before translating the information and sending it to the receiver. Because of this characteristic SAIL can monitor information flows without requiring additional functionalities. Furthermore, all messages and decisions can also be traced back to the originating actor.

Traceability is not yet part of SAIL in its current form. Although the communication and information flows in SAIL are sufficient, several challenges need to be addressed first for SAIL to support the envisioned traceability. For instance, tracing back an external origin of an event might become more difficult (e.g., a sunset makes the detection rate much lower, causing the system to miss a nearby child playing), as not all external events/concepts are mapped into the communication protocols and ontologies defined. In addition, traceability requires extra effort with regards to interaction design to prevent an information overload of the human due to the large amount of possible traced information. Concepts such as drill-down if needed from DASH (van Diggelen et al., 2021) where a user initially receives abstract information and can request more detailed information when needed, are relevant when determining how traceability should be supported by SAIL.

4.2 Envisioned Function; Centralized team AI

For SAIL to fulfill its role as a social AI layer between the human operator and autonomous systems, it should support collaboration with multiple autonomous systems and humans at the same time. Section 3.1 of this memo described how SAIL is technically able to do this by allowing connections between multiple humans and autonomous systems simultaneously. However, for teamwork of larger teams with multiple robots to be effective and efficient, a conceptual change in collaboration might be needed.

The primary issue in tasking or requesting information in a team with multiple team members is finding a suitable team member to ask. Who is capable and available? In the current SAIL implementation finding out this information is left to the human. However, a more efficient solution would be if the human can request *some* team member to complete the task, and let SAIL find the most suited one. In this manner the human – and GUI and HMT innovations it relies on – only has to think in terms of tasks they want to delegate and let SAIL do the rest.

For SAIL to support this capability, it requires a centralized team AI that can receive such a message addressed to multiple or all team members and mediate within the team who will execute the task. In its most rudimentary form, the centralized team AI could use a brute-force approach, where the message is forwarded to each team member until one accepts. However, this effectively shifts the effort from the human sending the message to the other team members who each must interpret the message, see if they are capable and available, and reply.
More efficient methods would incorporate the knowledge of the team members' capabilities and/or availability into the centralized team AI, such that it is able to make a more educated guess in finding the best suited actor for the task.

How to extract the information on capabilities and availability of team members in a human-machine team are active fields of research. For instance, gathering knowledge on the capabilities of team members is similar to current research on an autonomous system's reasoning about its own capabilities (e.g., see the SNOW Project[8]), extended to the team-level. Other research has the creation of mental models within a team as focus which incorporates team member availability and capabilities (Schadd, M.P.D., Schoonderwoerd, T. A. J., van den Bosch, K., Veltman, H.J., Visker, O.H., Haije, T. 2021).

The unique challenge of this potential SAIL function is finding an efficient method for identifying team members' capabilities and availability. If the method is too high-effort for team members, it might degrade task and total team performance more than just using the brute-force approach. On the other hand, there might be a preference towards making the work of a single human team member as easy as possible, even if that would result in more work for other team members.

4.3 Envisioned Function; pluggable SAIL modules

In the original vision for SAIL, focus was placed on the modular structure of the software framework, enabling developers to mix and combine several generic social AI modules living inside SAIL, as to provide the optimal social intelligence for the needs of that specific human-machine team.

Several example social functionalities for such modules could be measuring cognitive workload, human awareness, human attention, or a module for the specification of work agreements.

[8] https://appl-ai-tno.nl/flagships/snow/

This pluggable social intelligence concept is still particularly useful and is also already largely supported by the current SAIL implementation as presented in this memo. SAIL allows any number of clients to connect to its server and specify a name. As such, it is possible to connect social AI modules to SAIL via their own client. An example of what this setup could look like can be seen in Figure 2. A social AI module that measures cognitive workload may be connected to SAIL as a SAIL client, subscribe to the information it needs to calculate the cognitive workload (e.g., a camera steam or number of tasks), and make its information available to other clients. The human client, which may be a graphical user interface, can then subscribe to all information it requires from SAIL, including information from autonomous systems, as well as from social AI modules such as the cognitive workload module.

Although this functionality is supported in theory, testing in practice remains to make sure it all works as intended. Due to the lean and bottom-up approach to SAIL this was not possible yet to test in the current iteration of SAIL.

4.4 Other potential functions

Aside from the envisioned functions described above, another set of functions are envisioned which require more thought before they can be developed. These include a persistent SAIL memory, a team planner as part of the centralized team AI, and role-based permission management as part of SAIL. In this section these early ideas are shortly explained.

The persistent SAIL memory would be akin to a buffer for all information that passes SAIL. In this way, the case where tens of SAIL modules request the same information does not cause a flood of information requests to one specific autonomous system, as the information is buffered in SAIL (for a certain specified duration) greatly reducing the requests a human or autonomous system must deal with.

The second idea is that of extending the proposed centralized team AI with a multi-agent planner. Using such a multi-agent planner, the proposed centralized team AI could distribute tasks within a single Play over multiple systems, considering the capabilities, availability, and planning for the autonomous systems.

Finally, an interesting extension could be to extend SAIL with a permission management system. Within HMTs there may exist differences in hierarchy and roles, thus introducing the need for permission/access management. This also relates to the concept of meaningful human control as permission management may restrict specific actors from performing specific tasks without input from others such as to enforce a human-in-the-loop for situations. Secondly, a permission management system with roles is also advantageous for traceability, where more easily can be traced who performed what decisions and if it matched protocol or not.

5 Discussion

In human-machine teams, people collaborate with autonomous systems to achieve a common team goal. It is argued that successful teamwork relies on social capabilities, with at its core the capability for communication and mutual understanding. This memo presents SAIL as a middleware solution for social intelligence capabilities within HMTs. However, aside from the technical requirements embedded and presented as part of SAIL in this memo, there is a need for policy requirements to be taken into account as well.

An innovation such as SAIL both guides and is guided by the policy of the primary stakeholders and users, which for SAIL are the primary TNO DSS clients that aspire to incorporate autonomous systems in their organization in the form of a human machine team. Rich discussions sparked on topics such as meaningful human control over autonomous systems for defense, with much literature and policy recommendations (Horowitz & Scharre, 2015; ROFF, 2015). As such, the policy and organizational requirements for future HMT is an actively developing field for clients such as defense or national security, and should be taken into account in the further development of SAIL. Think of functionalities such as supporting traceability (see Section 4.1, and de Sio & van den Hoven, 2018), or supporting setting (international) standards on the development of autonomous military systems.

Aside from the technical and organizational requirements, some attention should go to the operational requirements as well. As the technical back end to support and enable an HMT, is it necessary for SAIL to provide a solution that is robust, redundant, and secure.

6 Summary and conclusion

Human machine teaming (HMT) requires the capability for humans and autonomous systems to communicate following a clear procedure, send messages that encode their intent, tasks or other information relevant to human machine teaming, and understand each other's intent through some form of translation. In addition, the solution should support teams with varying number of teammates, and be easily applicable to autonomous systems used by defense, national security, and other interested users for this technology. In this memo a second iteration of the Social Artificial Intelligence Layer (SAIL) is presented as a solution to the aforementioned requirements for successful human machine teaming. SAIL functions as middleware that serves generic services and functions for the HMT, such as providing a clear communication protocol, a communication message specification, and ontology targeted towards the generic HMT concepts of task delegation. Furthermore, the interchangeable ontology and other SAIL configuration makes it useful for a great number of other use cases as well.

The second iteration of SAIL has been built using a bottom-up in-house development style, ensuring that it can be quickly applied and adapted by users. Furthermore, a number of open challenges have been identified that came from policy, operational and technical requirements not yet implemented by SAIL. As a response, future functions with theoretical solutions have been described to envision how SAIL can be extended in the future to tackle challenges such as supporting traceability, delegation to multiple autonomous systems, and supporting the integration of other social capabilities required for HMT. As such, SAIL provides the thrust for the boat that is the human-machine team to propel forward. Now it only waits for a gust of wind to propel the HMT to new places.

7 References

de Sio, F. S., & van den Hoven, J. (2018). Meaningful human control over autonomous systems: A philosophical account. *Frontiers Robotics AI*, *5*(FEB), 1–14. https://doi.org/10.3389/frobt.2018.00015

Horowitz, M. C., & Scharre, P. (2015). Meaningful human control in weapon systens: A primer. *Working Paper*, 2–16.

J. van Diggelen, T. Mannucci, M.M.M. Peeters, B. van der Vecht, J. van der W. (2019). HATCL Specification. *TNO Report*.

Johnson, M., Bradshaw, J. M., Feltovich, P. J., Jonker, C. M., Van Riemsdijk, M. B., & Sierhuis, M. (2014). Coactive Design: Designing Support for Interdependence in Joint Activity. *Journal of Human-Robot Interaction*, *3*(1), 43. https://doi.org/10.5898/jhri.3.1.johnson

Lyons, J. B., & Havig, P. R. (2014). Transparency in a human-machine context: Approaches for fostering shared awareness/intent. *Lecture Notes in Computer Science (Including Subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *8525 LNCS*(PART 1), 181–190. https://doi.org/10.1007/978-3-319-07458-0_18

Mcdermott, P., Dominguez, C., Kasdaglis, N., Ryan, M., Mitre, I. T., & Nelson, A. (2018). Human-Machine Teaming Systems Engineering Guide. *The MITRE Corporation*. https://www.mitre.org/publications/technical-papers/human-machine-teaming-systems-engineering-guide

ROFF, H. and M. R. (2015). Meaningful Human Control , Artificial Intelligence and Autonomous Weapons. *Article 36*, 1–6.

Schadd, M.P.D., Schoonderwoerd, T. A. J., van den Bosch, K., Veltman, H.J., Visker, O.H., Haije, T. (2021). I'm afraid I can't do that, Dave. Getting to know your buddies in a human-agent team. *In Review*, 6–7.

Schneider, M., Miller, M., Jacques, D., & Peterson, G. (2021). Exploring the Impact of Coordination in Human-Agent Teams. In *Journal of Cognitive Engineering and Decision Making* (Vol. 00).

van der Vecht, B., van Diggelen, J., Peeters, M., Barnhoorn, J., & van der Waa, J. (2018). Sail: A social artificial intelligence layer for human-machine teaming. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10978 LNAI*(September), 262–274. https://doi.org/10.1007/978-3-319-94580-4_21

van Diggelen, J., Barnhoorn, J., Post, R., Sijs, J., van der Stap, N., & van der Waa, J. (2021). *Delegation in Human-Machine Teaming: Progress, Challenges and Prospects*. *January*, 10–16. https://doi.org/10.1007/978-3-030-68017-6_2

Vught, W. Van, & Veltman, K. (2021). Ontology specification RVO KMAS WP3. *TNO Report*, *June*.

Wynne, K. T., & Lyons, J. B. (2018). An integrative model of autonomous agent teammate-likeness. *Theoretical Issues in Ergonomics Science*, *19*(3), 353–374. https://doi.org/10.1080/1463922X.2016.1260181