“THEME ARTICLE”, “FEATURE ARTICLE”, or “COLUMN” goes here: The theme topic or column/department name goes after the colon.

Developing effective and resilient human-agent teamwork using team design patterns

Jurriaan van Diggelen
TNO, the Netherlands

Mark Neerincx
TNO, TU Delft, the Netherlands

Marieke Peeters
TNO, the Netherlands

Jan Maarten Schraagen
TNO, the Netherlands

Human-agent teams exhibit emergent behavior at the team level, as a result of interactions between individuals within the team. This begs the question how to design artificial team members (agents) as adequate team players that contribute to the team processes advancing team performance, resilience and learning. This paper proposes the development of a library of Team Design Patterns as a way to make dynamic team behavior at the team and individual level more explicit. Team Design Patterns serve a dual purpose: (1) In the system development phase, designers can identify desirable team patterns for the creation of artificial team members. (2) During the operational phase, team design patterns can be used by artificial team members to drive and stimulate team development, and to adaptively mitigate problems that may arise. We describe a pattern language for specifying team design patterns, discuss their use, and illustrate the concept using representative human-agent teamwork applications.

As autonomous systems are increasingly employed in various domains such as defense, logistics, and space, the need for adequate collaboration with humans remains an overarching concern. Human-agent teams (HATs) [1] consisting of humans and artificial team members are a popular paradigm to address these issues. Whereas much research has been performed to describe the requirements of an agent as a teammate, the implementation of HAT behavior in practical applications remains highly challenging. Traditional interaction design methodologies do not sufficiently address the autonomous capabilities of an agent, and often result in applications in which the human becomes a supervisor of the autonomous agent. Coactive design [2] is a useful step to develop one-to-one interaction between interdependent (human and artificial) teammates. A similar method is situated Cognitive Engineering [3] which focuses on the development of so-called ePartners that operate in human-agent groups. This paper aims at complementing these methods to capture team processes that emerge over longer periods of time with multiple humans and agents. For this purpose, we introduce the notion of team design patterns.

Team design patterns are essentially design patterns applied to the problem of developing HATs [4]. A design pattern is a generic, reusable, and proven solution to a commonly occurring problem within a given context. The idea was originally proposed in the architecture domain [5], but has been adopted in various other domains such as programming , workflow engineering [6], and interaction design [7].

A team design pattern (TDP), as introduced in this paper, can be defined as a description of generic reusable behaviors of actors for supporting effective and resilient teamwork. Just like interaction design patterns, TDPs aim to facilitate reuse of proven solutions and to stimulate uniformity of behavior across multiple applications (which is needed to obtain teammates that act in a predictable way [2]). However, teamwork contains some unique aspects that cannot be straightforwardly captured by interaction design patterns. Firstly, teamwork is performed by multiple autonomous actors cooperating to achieve a common goal [8]. Therefore, TDPs do not necessarily describe solutions that can be readily implemented at the level of individual agents, but they apply to the human-agent team as a whole, i.e. at the transaction level [9]. Secondly, human-agent teams are inherently dynamic as they continuously adapt themselves to changing environmental conditions and goals. This means that team patterns should be capable of capturing the long-term aspects of teamwork considering the development and application of TDPs by the system itself. Such patterns may not be initially present in the HAT or even deliberately designed into it.

TDPs can be used in multiple ways. By identifying desired team patterns in the system development phase, they can be used to create artificial team members with appropriate social behavior. During the operational phase, TDPs can be used by artificial team members to drive and stimulate team development, and to adaptively mitigate problems that may arise. This implies that TDPs can be further developed and adapted by HAT members to sustain adaptability.

The goal of this paper is to describe a method to specify TDPs (i.e., a pattern language) and explain their application for HAT. The method has been derived from experiences we have gained over the last decade developing and implementing HAT applications. The teamwork patterns are grounded in an ontology for HAT which helps to formalize the description of the problem and solution. We introduce multiple ways in which TDPs can be related to each other. For example, a TDP describing the behavior of an individual actor may be causally related to another TDP describing the behavior at the team level. Making these relations explicit is an important part of the proposed design method and may help to develop agents that invest in their team by exhibiting the right behaviors. We have developed the method by reverse engineering two representative projects in the space and logistic domain and used them to distill the TDP language. By phrasing these applications in terms of the pattern language, we illustrate the added value of the method and draw conclusions on how they can be used to guide design decisions.

BACKGROUND

When Christopher Alexander laid the groundwork of design patterns [5], he argued that design patterns can be distilled from practical construction efforts using a so-called pattern language, which provides a common format to generic solutions for commonly occurring problems. He stressed that design patterns only capture the very general and abstract aspects of a solution, and that the details must be filled in, given context in which the problem occurs. Furthermore, he introduced the notion of a pattern-relation, and wrote that “no pattern is an isolated entity. Each pattern can exist in the world only to the extent that it is supported by other patterns: the larger patterns in which it is embedded, the patterns of the same size that surround it, and the smaller patterns which are embedded in it.” [5]p.xiii. Each of these notions is very relevant for the architecture domain. They also turned out to be relevant for other domains, such as object-oriented programming, workflow engineering [6] and interaction design [7], [10].

We argue that design patterns are also a useful level of abstraction when dealing with the immensely complex engineering challenge of designing a set of intelligent agents that collaborate with humans in a team (i.e., a HAT). In a few recent papers the idea of a team design pattern has been put forward by ourselves [4] and colleagues [11], [12]. However, these proposals did not take the development of teams [13] into account. No team is instantaneously resilient and effective. These properties emerge over time after repeated interactions of its team members about goal-alignment, task-division and common ground [1]. This paper introduces a new pattern relation, which can be used to represent team development aspects.

The engineering methods proposed in this paper result from years of experience in designing intelligent artificial team members. The most prominent projects that contributed to the elicitation of the pattern language proposed in this paper are MECA [14] and RAILROAD [9]. Both projects are discussed in more detail in Section 4. We will first describe a classification of team design patterns, a template used for describing team design patterns, dynamic team behavior schemas, and the team ontology.

PATTERN LANGUAGE

A pattern language consists of patterns and relations among patterns. As a whole, the team design pattern language describes observable team behaviors and their expected effects. Our pattern language consists of three elements: 1) a classification of team design patterns and their relations 2) a team design pattern template, and 3) a team design pattern ontology.

A classification of team design patterns

The following distinctions are important when applying team design patterns. The first distinction is whether the pattern is constructive to the team or destructive. Constructive team patterns contribute to the team's cohesion, performance, and resilience. Destructive teamwork patterns degrade the team's cohesion, performance, and resilience. Destructive patterns (which are obviously undesirable) are an inherent aspect of team dynamics and may be caused by various reasons, such as conflicting interests, power struggles, incompetence, unexpected setbacks, or a lack of motivation. The general solution is to mitigate emerging destructive patterns with constructive team patterns. To better understand how to stimulate constructive team patterns, we introduce a second dimension: whether the pattern can be implemented on the individual team member level, or whether it is an emergent pattern at the collective team level. Individual teamwork patterns can be observed in the behavior of a single actor. These teamwork patterns can be directly translated to requirements for agent implementation, training of humans, or organizational requirements. Collective teamwork patterns can be observed in the behavior of a group of actors. These teamwork patterns are not directly implementable, but emerge from individual agent behaviors.

Using these dimensions, we can distinguish among four types of team patterns as depicted in the following table:

Table 1: Team patterns fall into one of four categories in the quadrant

	Constructive	**Destructive**
Collective	Team cohesion	Team degradation
Individual	Intervention	Resistance

- **Team cohesion patterns** are constructive patterns at the collective level. Examples of team cohesion patterns are decentralized task adoption and check-up behaviors.
- **Team degradation patterns** are destructive patterns at the collective level. Examples of team degradation patterns are the hen house pattern (i.e. talking without listening) and groupthink.
- **Intervention patterns** are constructive patterns at the individual level. Examples of intervention patterns are back-up behavior, team leader volunteering, and huddle initiation.

- **Resistance patterns** are patterns at the individual level that are destructive to the team. Examples of resistance patterns are spreading pessimism, gossiping and secrecy.

The four types of team patterns support the description of dynamic team behavior. This can be done by specifying pattern relations, which can be strengthening or weakening relations. By describing the TDPs assumed to be present in the system and their pattern relationships, we obtain a characterization of a HAT similar to a complex dynamic system. This enables individual agents to reason on how they can influence collective team patterns. For example, upon observing a pattern of team degradation, an individual agent may activate an intervention pattern to steer the collective behavior towards a pattern of team cohesion. Finally, we note that a TDP that is constructive for one context, may be destructive in another context. For example, some teams may coordinate activities by sharing information among team members to foster shared situation awareness. In such cases, frequent group meetings are expected to be constructive as a team pattern. However, in teams where individuals can conduct their work in parallel and independent of each other's work, frequent group meetings are expected to slow down the team and will be destructive as a team pattern.

Team design pattern template

Team design patterns can be described using a fixed format. This format, which is an extension of earlier work [4], is provided below:

Title of pattern	
Behavior pattern	Description of the behavior observed when this pattern is active.
Positive effect	The potential positive outcomes of this pattern in terms of for instance team performance, task performance, team cohesion, resilience.
Negative effect	The potential negative outcomes of this pattern.
Use when	Context factors that influence when the positive effects outweigh the negative effects.
Example	Describes example use case and implementation in which the potential benefits and downsides of this pattern are clarified.
Design rationale	Theoretical foundation, grounded in literature / empirical research, explaining and justifying the expected positive and negative effects of this team pattern.
Type	The type of pattern, i.e. *collective* or *individual*.

The template requires the designer to specify seven features. First, the behavior observed when the pattern is active must be described, along with the type of pattern describing whether the pattern occurs at the individual or the collective level. Then, expected positive and negative outcomes must be provided. Together, the positive and negative effects can be used by a designer to determine under which circumstances the design pattern is expected to be constructive to the team, and under which circumstances it is expected to be destructive to the team. This is described in the "use when" field. To illustrate the use of the team design pattern, an example should be provided in the form of a use case and implementation. The design rationale provides the underlying argumentation justifying the expected positive and negative effects of the team pattern.

Team ontology

The team patterns are defined with reference to a team ontology. An excerpt of this ontology is depicted below.

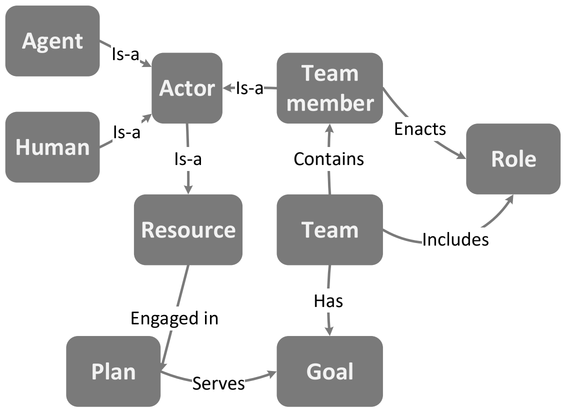


Figure 1 Team ontology

A team consists of **team members** that are **human** or **agent actors**, each of which fulfills a particular role in the team. A team has a **goal**, which may change over time. The goal can be pursued using a **plan** which requires **resources**. Plans are used in various ways within HAT [8]. A **plan** might be explicitly agreed upon beforehand (such as a drill or mission-plan), established ad hoc by the team members, or implicitly assumed. **Resources** engaged in a plan may be tools, consumables, or **team members** that are scheduled to perform part of the plan.

TEAM PATTERN EXAMPLES

We will now present two examples of team patterns that have been observed in our prior work when designing Human Agent Teams in the domains of space and rail traffic control.

Space

Human space flight is a prototypical example of a complex, dynamic, and safety-critical domain in which missions are performed by collaborative teams of humans and technical systems. In the MECA-Heart project [14], we aimed at developing a team of astronauts and agents (ground-robots and ePartners), capable of performing exploratory missions on the surface of Mars. Unforeseen conditions and events (e.g., failures of machines, environmental barriers or dysfunction of humans) are inevitably present in these space missions. An adequate solution to these problems can often not be established by one person alone, but requires knowledge, competences, and authorizations that are distributed over different team members.

For this example, we will describe the MECA system following the pattern language described in the previous section and present three TDP templates. The relationships between them is depicted below in Figure 2.

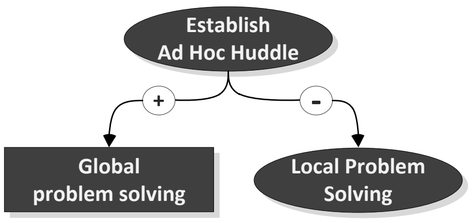


Figure 2: Team design patterns underlying the MECA Heart prototype. Ovals represent individual TDPs, and rectangles represent collective TDPs. The strengthening pattern relations are described using a + sign, and the weakening relations using a - sign.

This overview states that the collective TDP *Global problem solving* can be achieved by the individual TDP *Establish ad hoc Huddle* (a huddle is a brief exchange of relevant knowledge by participating team members, often according to a prescribed format). Because *Establish ad hoc Huddle* also has a weakening effect on TDP Local problem solving, it should only be applied when *Local problem solving* is regarded as destructive in the current context, and *Global problem solving* is constructive. The following three TDP templates aim to specify these patterns in further depth such that these trade-off decisions can be made.

Global problem solving	
Behavior pattern	**Team-members** share different solution options with each other to ensure that the **goal** of the **team** is achieved by changing current **goal**, **plans** or **resources** and choose the best viable alternative.
Positive effect	A broader range of solutions can be taken into consideration ranging from changing the **goals**, the **resource** allocations, and **plans**.
Negative effect	Collaborative problem-solving causes communication overhead.
Use when	When different **actors** have complementary expertise, situation awareness, and authorizations.
Example	An astronaut and a rover are on extra vehicular activity (EVA) on Mars' surface. The team leader is in the habitat. They collectively discuss options how to solve a problem with a malfunctioning wheel on the rover which is draining the battery. The astronaut proposes to fix the wheel, so they have enough energy supply to continue the mission. The team leader proposes to change the mission goal and set up the destination of the journey to a more nearby location. They collectively decide to change the mission goal.
Design rationale	In the usage conditions (i.e. use-when), a jointly established solution is often beneficial [18]
Type	Collective

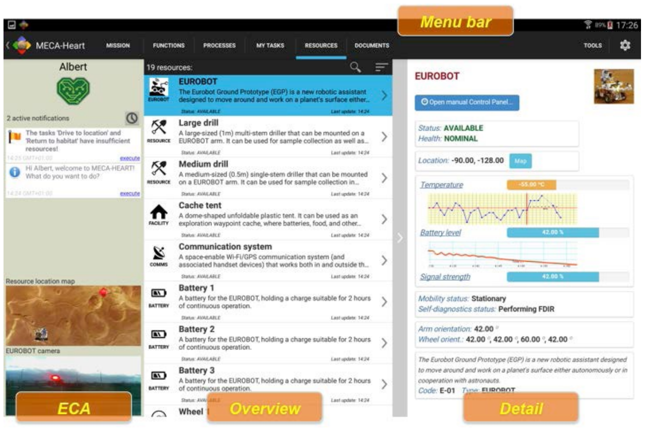
The opposite of *global problem solving* is *local problem solving*:

Local problem solving	
Behavior pattern	A **team member** changes a **goal**, **plan**, or **resource**-allocation without involving other **team members**.
Positive effect	Solution can be chosen and implemented fast without interfering with the activities of other team members.
Negative effect	Solution may be suboptimal as other team members may be able to implement better solutions.
Use when	When a team member has sufficient expertise, situation awareness, and authorizations to solve the problem alone.
Example	An astronaut on EVA fixes a problem with a malfunctioning wheel of the Rover using the repair instructions given by the MECA assistant and continues the mission.
Design rationale	When **team members** are self sufficient [17], interaction with other on a task could cause unnecessary overhead.
Type	Collective

The *global problem solving* pattern is not directly implementable (because it is collective) but requires another pattern *establish ad hoc huddle* (which is individual).

Establish ad hoc huddle	
Behavior pattern	A **team member** predicts that the **goal** of the **team** is not achievable and takes the initiative to promptly establish a joint huddle with all other **team members**.
Positive effect	Enables other team patterns that require a huddle
Negative effect	It distracts other team members from their work.
Use when	An **actor** predicts that the **goal** of the **team** is predicted as unachievable using the current **plans**, and **resources**. The **actor** believes that no **team member** is self sufficient to solve the problem.
Example	An astronaut detects a problem with a malfunctioning wheel of the rover. He establishes an ad hoc huddle using MECA. The other participants are notified about the huddle and interrupt their work to participate in the huddle.
Design rationale	Digital collaboration at a distance tools can be effectively applied for synchronous groupwork on a global problem solving task [18]
Type	Individual

The individual TDPs have been implemented in the MECA prototype to stimulate the collective TDP when appropriate. This has led to the design of the most prominent aspects in the MECA interface: a multi-layered view on the problem space (e.g., resources, tasks, and missions) which helps the detection of problems, and a collaborative chat functionality which stimulates the collaboration on problem solving. In [14], a user experiment with establish ad hoc huddle is described.



2 active notifications

The tasks 'Drive to location' and 'Return to habitat' have insufficient resources!
14:25 GMT+01:00 execute

Hi Albert, welcome to MECA-HEART! What do you want to do?
14:24 GMT+01:00 execute

19 resources:

EUROBOT
The Eurobot Ground Prototype (EGP) is a new robotic assistant designed to move around and work on a planet's surface either...
Status: AVAILABLE Last update: 14:24

Large drill
A large-sized (1m) multi-stem driller that can be mounted on a EUROBOT arm. It can be used for sample collection as well as...
Status: AVAILABLE Last update: 14:24

Medium drill
A medium-sized (0.5m) single-stem driller that can be mounted on a EUROBOT arm. It can be used for sample collection in...
Status: AVAILABLE Last update: 14:24

Cache tent
A dome-shaped unfoldable plastic tent. It can be used as an exploration waypoint cache, where batteries, food, and other...
Status: AVAILABLE Last update: 14:24

Communication system
A space-enable Wi-Fi/GPS communication system (and associated handset devices) that works both in and outside th...
Status: AVAILABLE Last update: 14:24

Battery 1
A battery for the EUROBOT, holding a charge suitable for 2 hours of continuous operation.
Status: AVAILABLE Last update: 14:24

Battery 2
A battery for the EUROBOT, holding a charge suitable for 2 hours of continuous operation.
Status: AVAILABLE Last update: 14:24

Battery 3
A battery for the EUROBOT, holding a charge suitable for 2 hours of continuous operation.
Status: AVAILABLE Last update: 14:24

Wheel

EUROBOT

Status: AVAILABLE
Health: NOMINAL
Location: -90.00, -128.00
Temperature -55.00 °C
Battery level 42.00 %
Signal strength 42.00 %
Mobility status: Stationary
Self-diagnostics status: Performing FDIR
Arm orientation: 42.00 °
Wheel orient.: 42.00 °, 42.00 °, 60.00 °, 42.00 °
The Eurobot Ground Prototype (EGP) is a new robotic assistant designed to move around and work on a planet's surface either autonomously or in cooperation with astronauts.
Code: E-01 Type: EUROBOT

Figure 3: MECA prototype supporting ad-hoc huddles and collaborative problem solving via problem space views and collaborative chat functions

Rail traffic control

Rail signalers continuously fit unplanned train movements in the real-time flow of trains. Each signaler is responsible for their own geographical area within which they monitor the traffic situation and mitigate any disturbances within the bounds of their area of control. Deviations and other irregularities are only discussed when things go wrong and need justification or explanation. Occasionally, the results of these discussions may lead to procedure updates. Most often, rail signalers focus on real-time operations; they rarely reflect on their daily experiences nor the procedures they use. Only upon request – usually after serious disturbances – are log data about the complete rail system analyzed to trace the origin of the disturbance, and review the operational procedures to prevent such disturbances in the future.

To assist a team of rail signalers in reflecting upon their operations and processes, particularly before major disturbances have occurred, the Resiliencer was developed [19]. The Resiliencer supports team reflection with respect to three boundary categories for weak resilience signals: performance, workload, and safety. When the team of rail signalers collectively reflect upon the team's objectives, strategies, and processes, the effectiveness of the team itself is improved, and organizational innovation in the context of the team's work is stimulated.

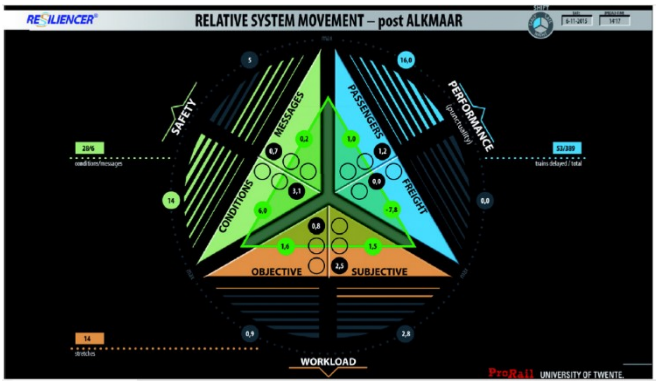


Figure 4 Resiliencer screen showing the main dimensions of team performance: workload, performance, and safety.

The Resiliencer seizes onto six main team patterns to improve the team dynamics and the resulting team performance.

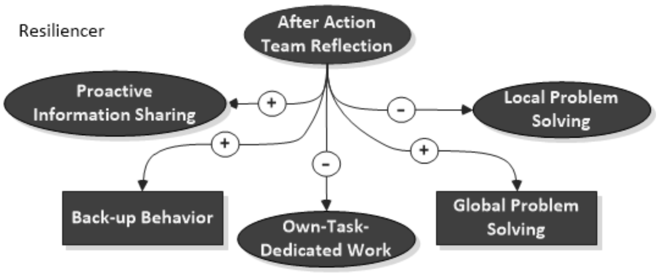


Figure 5: Complex dynamic system of team patterns in the Resiliencer use case

Due to space limitations, we will not fully describe these patterns using the TDP template and confine ourselves to a brief description. All team patterns are influenced by the pattern *After action team reflection* which occurs after each work shift and gives the participants the possibility to discuss and reflect on recent performance of the rail system as a whole. One of the team patterns that is enforced by the *after-action review* is *back-up behavior*. This is because team members become more aware of the overall task and the interdependencies between each other's tasks. As a result, they help one another by sharing information, by creating leeway for each other to make up for problems elsewhere in the team, by taking over part of their team members' workload for a given time period, or by performing other activities outside of the team member's own responsibility in support of the team recovering from events in the environment / system. The positive effect of *backup behavior* is increased resilience because small deviations or hiccups in the team are solved without too much extra coordination effort.

Using a similar line of reasoning, we identified other team patterns that can be enforced or discouraged during the *after-action team reflection*. These are *local* and *global problem solving* (as discussed in the previous section), *pro-active information sharing*, and *own task dedicated work*.

CONCLUSION

We have proposed a method for specifying reusable team design patterns that can be used to shape the design of human-agent teams. We have applied the method to design human-agent teams that invest in team development to become a better team in the long term.

Such high-level design methods that take this longitudinal dimension into account are crucial when designing intelligent adaptive systems. Our proposal is a first step to make the intended patterns of team development explicit.

We identify several directions for future research. Firstly, we intend to develop evaluation methods for the team patterns and their relations that can be used to close the cycle of development, prototyping, and testing. Secondly, we intend to develop artificial intelligence techniques that can be used to develop agents that recognize team development patterns themselves and use these to guide their social behavior. Thirdly, we intend to develop a pattern library that contains the most relevant team patterns at this level of abstraction and use this as an evolving knowledge base for enhancing human-agent teamwork, entailing progress in its theoretical and empirical foundation.

Acknowledgment

This work was supported by the Dutch Ministry of Defense's exploratory research program Human-Machine Teaming.

References

[1] Klein, G., Woods, D. D., Bradshaw, J. M., Hoffman, R. R., & Feltovich, P. J. (2004). Ten challenges for making automation a" team player" in joint human-agent activity. IEEE Intelligent Systems, 19(6), 91-95.

[2] Johnson, M., Bradshaw, J. M., Feltovich, P. J., Van Riemsdijk, M. B., Jonker, C. M., & Sierhuis, M. (2014). Coactive design: Designing support for interdependence in joint activity. Journal of Human-Robot Interaction, 3 (1), 2014.

[3] Neerincx, M. A. (2011). Situated cognitive engineering for crew support in space. Personal and Ubiquitous Computing, 15(5), 445-456

[4] Neerincx, M. A., van Diggelen, J., & van Breda, L. (2016, July). Interaction design patterns for adaptive human-agent-robot teamwork in high-risk domains. In International Conference on Engineering Psychology and Cognitive Ergonomics (pp. 211-220). Springer International Publishing.

[5] Alexander, C., Ishikawa, S., Silverstein, M., i Ramió, J. R., Jacobson, M., & Fiksdahl-King, I. (1977). A pattern language (pp. 311-314). Gustavo Gili.

[6] van Der Aalst, W. M., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow patterns. Distributed and parallel databases, 14(1), 5-51.

[7] Borchers, J. O. (2000, August). A pattern approach to interaction design. In Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques (pp. 369-378). ACM.

[8] Tambe, M. (1997). Towards flexible teamwork. Journal of artificial intelligence research, 7, 83-124.

[9] Schraagen, J.M.C. (2017). Beyond Macrocognition: The Transaction Level. In J. Gore & P. Ward (Eds.), NDM13 Naturalistic Decision Making and Uncertainty, Proceedings of the 13th bi-annual international conference on Naturalistic Decision Making, Bath, 20-23 June 2017 (pp. 182-188). Bath: The University of Bath.

[10] Mioch, T., Ledegang, W., Paulissen, R., Neerincx, M.A. and van Diggelen, J. "Interaction design patterns for coherent and re-usable shape specifications of human-robot

collaboration," in Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems. ACM, 2014, pp. 75–83.

[11] Schulte, A., Donath, D., & Lange, D. S. (2016, July). Design patterns for human-cognitive agent teaming. In International Conference on Engineering Psychology and Cognitive Ergonomics (pp. 231-243). Springer International Publishing.

[12] Kahn, P. H., Freier, N. G., Kanda, T., Ishiguro, H., Ruckert, J. H., Severson, R. L., & Kane, S. K. (2008, March). Design patterns for sociality in human-robot interaction. In Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction (pp. 97-104). ACM.

[13] van Diggelen, J., Looije, R., van der Waa, J., & Neerincx, M. (2017, July). Human Robot Team Development: An Operational and Technical Perspective. In International Conference on Applied Human Factors and Ergonomics (pp. 293-302). Springer, Cham.

[14] Bosse, T., Breebaart, L., Diggelen, J. V., Neerincx, M. A., Rosa, J., & Smets, N. J. (2017). Developing ePartners for human-robot teams in space based on ontologies and formal abstraction hierarchies. International Journal of Agent-Oriented Software Engineering, 5(4), 366-398.

[15] Bradshaw, J. M., Hoffman, R. R., Woods, D. D., & Johnson, M. (2013). The Seven Deadly Myths of" Autonomous Systems". IEEE Intelligent Systems, (3), 54-61.

[16] Hollnagel, E., Woods, D. D., & Leveson, N. (2007). Resilience engineering: Concepts and precepts. Ashgate Publishing, Ltd..

[17] McFarland, D., & Spier, E. (1997). Basic cycles, utility and opportunism in self-sufficient robots. Robotics and Autonomous Systems, 20(2-4), 179-190.

[18] Roschelle, J., & Teasley, S. D. (1995, August). The construction of shared knowledge in collaborative problem solving. In Computer-supported collaborative learning (Vol. 128, pp. 69-197).

[19] Siegel, A.W., & Schraagen, J.M.C. (2017). Team reflection makes resilience-related knowledge explicit through collaborative sensemaking: Observation study at a rail post. Cognition, Technology & Work, 19(1), 127-142.

[20] Tuckman, B. W. (1965). Developmental sequence in small groups. Psychological bulletin, 63(6), 384