TYPE Original Research
PUBLISHED 07 December 2022
DOI 10.3389/fphy.2022.969544

Interdependence design principles in practice

Micael Vignati*, Matthew Johnson, Larry Bunch, John Carff and Daniel Duran

Institute for Human and Machine Cognition, Pensacola, FL, United States

OPEN ACCESS

EDITED BY
William Frere Lawless,
Paine College, United States

REVIEWED BY
Bonnie Johnson,
Naval Postgraduate School,
United States
Mito Akiyoshi,
Senshu University, Japan

*CORRESPONDENCE
Micael Vignati,
mvignati@ihmc.org

SPECIALTY SECTION
This article was submitted to Interdisciplinary Physics, a section of the journal Frontiers in Physics

RECEIVED 15 June 2022
ACCEPTED 23 November 2022
PUBLISHED 07 December 2022

CITATION
Vignati M, Johnson M, Bunch L, Carff J and Duran D (2022), Interdependence design principles in practice. *Front. Phys.* 10:969544. doi: 10.3389/fphy.2022.969544

Adaptability lies at the heart of effective teams and it is through management of interdependence that teams are able to adapt. This makes interdependence a critical factor of human-machine teams. Nevertheless, engineers building human-machine systems still rely on the same tools and techniques used to build individual behaviors which were never designed to address the complexity that stems from interdependence in joint activity. Many engineering approaches lack any systematic rigor and formal method for identifying, managing and exploiting interdependence, which forces ad hoc solutions or workarounds. This gap between theories of interdependence and operable tooling leaves designers blind to the issues and consequences of failing to adequately address interdependence within human-machine teams. In this article, we propose an approach to operationalizing core concepts needed to address interdependence in support of adaptive teamwork. We describe a formalized structure, joint activity graphs, built on interdependence design principles to capture the essence of joint activity. We describe the runtime requirements needed to dynamically exploit joint activity graphs and to support intelligent coordination during execution. We demonstrate the effectiveness of such a structure at supporting adaptability using the Capture-the-Flag domain with heterogeneous teams of unmanned aerial vehicles and unmanned ground systems. In this dynamic adversarial domain, we show how agents can make use of the information provided by joint activity graphs to generally and pragmatically react and adapt to perturbations in the joint activity, the environment, or the team and explicitly manage and exploit interdependence to produce effective teamwork. In doing so, we demonstrate how flexible and adaptive teamwork can be achieved through formally guided design that supports effective management of interdependence.

KEYWORDS

human-machine teams, joint activity, interdependence, adaptability, joint activity graph, heterogeneous multi-agent systems

1 Introduction

Teaming is a dynamic activity that comes to life as agents[1] work together towards a common goal. Understanding the factors of teaming is critical to produce effective human-machine teams. Approaches to understanding teaming are as numerous as they are multidisciplinary, involving human sciences (sociology [1], linguistics [2], psychology, etc.), engineering sciences (distributed artificial intelligence, constraint satisfaction problems, planning [3], synchrony [4]) and human machine cognition often trying to bridge the two (theory of mind, common ground [5], communication, trust [6]). On one hand, research in theoretical models for understanding teaming is extensive and cohesive, with some empirically deriving principles of cooperation (e.g., in cognition enabled multi-agent systems [1]). On the other hand, the applied aspect of teaming and its understanding has been assessed to be far less consistent across the research community [7]. As systems become more sophisticated and take on increasingly complex roles, the need for tools which help researchers, designers and engineers consider and exploit all dimensions of teaming are key for both effectiveness and acceptance.

According to the concept of bounded rationality [8], agents are required to work together in any system of substantial complexity. Whether due to the distribution of skills, capabilities and knowledge or simply to improve aspects of performance, teaming is unavoidable and makes coordination between agents a pragmatic requirement. Malone and Crowston define coordination as "*the act of managing interdependencies between activities*" [9]. Johnson and Bradshaw make the case that "*the understanding of interdependence is key to characterizing human-machine teamwork in an understandable, actionable, and generalizable manner*" [10]. Interdependence is sometimes characterized as interference (which can be positive or negative) [1, 11] or as dependence [1]. A few types of interdependence and their impact on task quality and maximum duration were formalized by Decker as non-local-effects [12]. It is clear from the research community that interdependence is a key component to teams [9, 13, 14] and makes the need for a theory of interdependence fundamental [10, 15]. However, interdependence is complex and for that reason it has historically been avoided.

Adaptability is a central aspect of teaming. The capacity to adapt is often correlated with team performance. There is a significant body of research on *adaptive autonomy* [16–18] which followed the definition of levels of automation, and defined adaptive autonomy as switching between levels of automation. This work demonstrated the impact of automation design choices on human-machine performance. Other work has focused more broadly on *adaptability* [19], not limiting it to levels of automation. Adaptability is about allowing agents to understand and react to change cognitively or physically. Adapting allows agents to mitigate adverse effects and opportunistically take advantage of situations to improve performance along specified dimensions of teaming. We posit that adaptability is the ability to negotiate and manage the interdependencies within the team to yield behavior classified as *good teamwork*.

Given the importance of human-machine systems, there are surprisingly few tools available to support designing, building, execution and interaction with human-machine teams. Existing tools typically target a single aspect of human-machine systems, such as distributed communications, or task allocation. Our desire is to develop a more comprehensive approach based on the extensive body of research on teaming. In a previous paper, *Understanding Human-Autonomy Teaming through Interdependence Analysis* [20], we provided principles to identify and understand interdependencies within a human-machine-system, helping designers design effective teaming systems. We now take the next step of providing tools to operationalize these principles in practice.

In this paper, we present joint activity graphs (JAGs), a formalism providing a systematic method to capture the essential elements necessary to describe and execute joint activity. We also introduce the JAG Engine as a runtime JAG interpreter and a means to execute multi-agent behavior. This engine handles the aspects of teaming that must be dynamically determined, such as team composition, task participation/allocation, and communication. The JAG Engine leverages the JAG formalism to support runtime teamwork through management of interdependence. Together, these tools provide designers and builders a practical and systematic approach to creating joint activity behaviors that support coordination processes within a team. Because JAGs are grounded in teamwork theory, they also enable a highly adaptive system. To demonstrate the range of adaption possible, we provide examples grounded in a *Capture-the-Flag* (CTF) domain. CTF is a fast-paced adversarial domain requiring quick and responsive adaptation within the team to be successful. This systematic approach, combined with the formalism described in this paper, help define and expose the different types of interdependence common to a broad range of activities and their associated coordination requirements, which in turn supports *good teamwork*.

2 Background

Providing agents with a computational means to determine their own behavior is necessary for agents to be useful. Many approaches have been developed over the years, such as planning systems [21], and reactive behaviors [22]. However, these are

1 In this paper agent indistinguishably refers to human or machine.

behavior architectures, not guidance on how to develop specific behaviors suitable for joint activity. We next discuss some representational techniques important to the design of joint activity.

2.1 Hierarchical task networks

Per Erol, in 1994, "*most of the practical work on AI planning systems during the last 15 years has been based on Hierarchical Task Network (HTN) decomposition*" [23], and while HTNs now share the scene with machine learning, this statement mostly still holds true [3]. HTNs introduced the concept of compound tasks which were lacking in classical automated planning systems such as STRIPS [21] and PDDL, dramatically reducing the search space at the expense of domain knowledge. HTNs provide declarative goals and a rich constraint language on intermediate states that can express a large space of interactions. A key challenge with HTNs is their lack of support for parallel activity, which is critical in joint activity. Classical HTN planning does not explicitly preclude parallel behavior (in that an agent can execute multiple plans in parallel) but unfortunately does not concern itself with the complexity and relationships that may arise from parallel activities. We will see in Section 4.2.2.2 that we take the opposite approach and assume that all activities can be executed in parallel unless otherwise constrained.

Work based on hierarchical models is often concerned with task decomposition but seldom with *answer synthesis* (how to re-compose subtasks back together to fulfill the goal they decompose and their interactions) [24]. Duarte proposes a hybrid controller [25] as a solution to the *answer synthesis* problem [24] as presented by Smith and demonstrates that controllers can be synthesized hierarchically by applying it to the swarm multi-agent domain [26]. With regards to distributed artificial intelligence, Durfee compared the agent coordination to a search in hierarchical space which very cleverly approaches the problem of synthesis by grouping and abstracting behaviors at multiple levels [27] fully taking advantage of the composition capabilities of hierarchical task networks.

Decomposition and synthesis are critical components of designing joint activity. Choices made will enable or hinder exploitation of associated interdependencies and will have a substantial impact on teamwork. We build on the strengths of HTNs and extend it to include an understanding of interdependence (supported by the *4S framework for understanding teamwork* [28] and *interdependence analysis* [29, 30]), as well as a generalization of synthesis.

2.2 Behavior trees

Behavior trees are a form of hierarchical decomposition of agent behavior that has its genesis in video games. They were first proposed as good engineering practice to handle the complexity of large systems and allow behaviors to be more reactive to changes in requirements (such as behaviors of non player entities in video games).

Behavior tree use in robotics and artificial intelligence has been steadily growing in last decade [31]. They are a hierarchical decomposition of agent behavior, grounded in execution, data driven, and reusable which makes them a go-to model when designing agent behaviors. In these respects, they are similar to the joint activity formalism that we present in Section 3. However, behavior trees are significantly different in other aspects. They were initially designed with single agent behavior in mind (with sparse and disconnected attempts to be augmented to support multi-agents). Thus, they hide interdependencies, resulting in ad hoc solutions when trying to apply them to build joint activity (multi-agent behavior). They are re-evaluated often, typically multiple times per second, and do not hold state. Behavior trees stateless nature and their lack of data flow make data usage within a behavior pragmatically hidden and require the use of back channels for information sharing, such as a blackboard. Like HTNs, parallel execution is not precluded (behavior tree formalism has been augmented with an explicit parallel node) but there is no context support for the resulting interdependencies.

In contrast, a joint activity graph is always designed from a multi-agent perspective, with single agent behavior being the degenerate case. JAGs are event driven, as opposed to being reevaluated at regular intervals, providing observability into teamwork processes, enabling causality tracing and adaptation explanation. In stark contrast to a behavior tree's lack of state and blackboard back channel, data flow (i.e. inputs, outputs and bindings) is a central part of the JAG model. This makes tying data to joint activities possible and in turn enables and simplifies the process of identifying relevance of information (see Section 6.3). JAG inputs further specify its behavior and as such are part of the context necessary to make decisions. In that respect inputs satisfy coactive design interdependence requirements: observability, predictability and directability (OPD) [29].

We build on the practicality of behavior trees (composability, grounded in execution and data driven) to augment our framework with established practices and adapt it to the domain of human machine teaming.

2.3 TAEMS

In his inspirational thesis, Decker describes a generalization of Durfee's Partial Global Planning [27] called TÆMS or Generalized Partial Global Planning [12]. TÆMS is described as a "*domain-independent coordination framework for small agent groups*" [32]. It expands on the domain specific limitation of Partial Global Planning by including a more abstract and hierarchical representation of the joint activity allowing a generalized identification and management of coordination relationships (interdependencies).

Our work heavily builds upon and extends this work, both in the structure (task decomposition and quality) and interdependencies (non local-effects). Decker formalises hierarchical synthesis in the form of quality accrual functions (e.g., min, max, average) making it consistent with task interdependence (see Section 4.2.1). Decker also formalises a significant set of interdependencies (e.g., *enables*, *facilitates*) and their measurable effect on tasks' quality and duration.

Joint activity graph expands this work to also include OPD requirements as team interdependencies [29]. Most, if not all, concepts described in TÆMS have direct overlap or are generalized in the joint activity graph formalism that we propose.

3 Joint activity graphs

Joint Activity Graphs (JAGs) are a new method to describe *joint* activity in a way that is executable. Our goal with JAGs is to provide a rigorous and systematic method for defining joint activity that can be run in a distributed manner and achieve behavior that would be described as *good teamwork*. A key design principle when employing JAGs is that all work should be designed as joint work [20], meaning the JAG should be designed with an understanding that multiple agents will be involved in performing the work. This is a dramatic shift from the typical single-agent behavior mindset.

JAGs describe the solution space of joint behaviors. They capture the goals and actions necessary, as well as the options and contingencies available. Because of this, JAGs are not simply a plan, but a description of the set of alternatives available to the team.

A major challenge in defining a JAG is understanding the interdependencies within the joint work. Teamwork is complex and involves the interplay of dimensions such as team goals, task work, team composition, execution strategies and interdependencies as discussed in 4. Malone and Crowston stated that "*one of the most intriguing possibilities for coordination theory is to identify and systematically analyze a wide variety of dependencies and their associated coordination processes*" [14]. The JAG structure is defined to provide a framework for capturing the interdependence systematically. It provides a common structure onto which teaming information within a joint activity is captured. This structure allows designers to systematically consider a broader range of teamwork aspects at design time than commonly supported by current tools and techniques. The JAG definition includes the hierarchical work, similar to HTNs. It also includes synthesis functions in a more generic manner than found in behavior trees. Lastly the JAG includes the necessary information for capturing data flows.

Formally, a jag $\gimel$ is defined as the tuple

$$\gimel = \langle J_\gimel, s_\gimel, I_\gimel, O_\gimel, B_\gimel \rangle$$

$J_\gimel$ is the set of joint activity graph children of $\gimel$

$$J_\gimel = \{\gimel_1, \gimel_2, \ldots\}$$

$s_\gimel$ is a synthesis function over its own inputs and $J_\gimel$'s outputs

$$\left(\bigcup_{n=1}^{|J_\gimel|} O_{\gimel_n} \cup I_\gimel \right) \overset{s_\gimel}{\mapsto} O_\gimel$$

$I_\gimel$ is the set of $\gimel$'s input parameters

$$I_\gimel = \{i_1, i_2, \ldots\}$$

$O_\gimel$ is the set of $\gimel$'s output parameters

$$O_\gimel = \{o_1, o_2, \ldots\}$$

$B_\gimel$ is the set of bindings representing the output-input data flow within $\gimel$

$$B_\gimel = \{b_1, b_2, \ldots\}$$

where

$$b_k \in \left(\bigcup_{n=1}^{|J_\gimel|} O_{\gimel_n} \cup I_\gimel \right) \times \left(\bigcup_{n=1}^{|J_\gimel|} I_{\gimel_n} \cup O_\gimel \right)$$

These features, represented in a JAG definition, capture the essential elements needed to interpret interdependencies within joint activity. An example of a generic JAG definition, such as the jag pictured in Figure 1 would be defined as follows:

$$\begin{aligned}
\gimel &= \langle \{\gimel_a, \gimel_b\}, s_\gimel, \varnothing, \varnothing, \varnothing \rangle \\
\gimel_a &= \langle \{\gimel_{a,1}, \gimel_{a,2}\}, s_{\gimel_a}, \varnothing, \varnothing, \{(o_1^{a,1}, i_1^{a,2})\} \rangle \\
\gimel_{a,1} &= \langle \varnothing, s_{\gimel_{a,1}}, \varnothing, \{o_1^{a,1}\}, \varnothing \rangle \\
\gimel_{a,2} &= \langle \varnothing, s_{\gimel_{a,2}}, \{i_1^{a,2}\}, \varnothing, \varnothing \rangle \\
\gimel_b &= \langle \varnothing, s_{\gimel_b}, \varnothing, \varnothing, \varnothing \rangle
\end{aligned}$$

It should be noted that the JAG formalism intentionally does not describe the team or the strategy. This is consistent with the interdependence design principles [20], appropriately separating these concerns. The formalism, as we will show, does work with both at runtime.

This JAG formalism is beneficial in a variety of ways. First, it provides a framework for tracking the information necessary for understanding the teaming context within an activity. Additional information about team context, such as team composition, task allocation, and task progress, while not defined in the JAG, can be tracked through the JAG. This enables individual agents to make effective single agent behavior choices that are consistent with good teamwork decisions. Second, the framework provides agent coordination mechanisms to facilitate appropriate team interactions at runtime based on that team context reasoning (see Section 4). In other words, the formalism

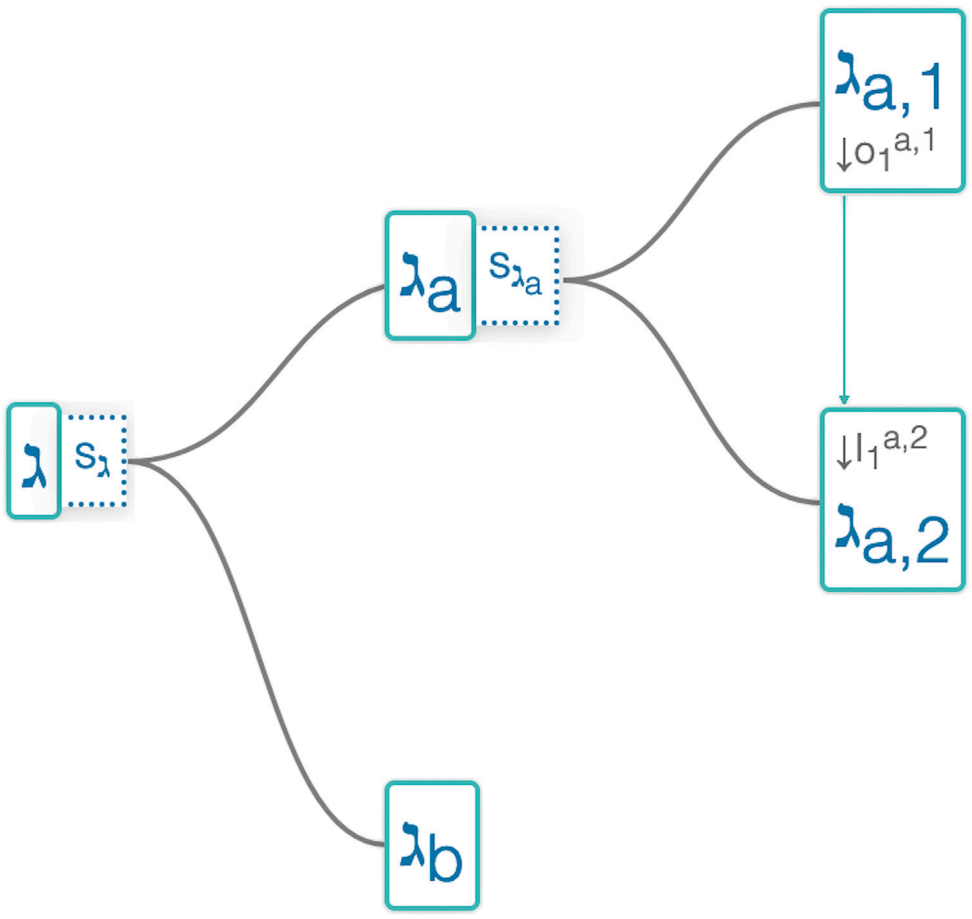

FIGURE 1
Example of joint activity graph representation showing decomposition, synthesis and data binding between two siblings.

provides the minimal situation awareness necessary for collaborative contexts.

Before explaining how the JAG formalism helps address interdependence, we will first expand on the broad range of sources of interdependence that occur within joint activity. To do so, we will reference the 4S interdependence framework for understanding teamwork [28].

4 Teamwork challenges

One of the main reasons understanding teamwork is challenging is because teamwork involves a wide range of interdependencies. It should not be a surprise that different kinds of teamwork can be distinguished according to the types of interdependence involved. For example, lifting a couch together involves different interdependencies than sharing a hammer. Each type of interdependence can involve different coordination mechanisms necessary to manage it. For example, lifting a couch might require agreeing and reacting to a start signal ("lift on 3"), while sharing a hammer could require verbal notification of completion or even simple observation of availability of the hammer. As such, operationalizing teamwork requires developing support for managing a range of interdependent relationships using a range of coordination mechanisms and techniques. Johnson et. al [28], proposed a framework for organizing many of the important concepts associated with teaming based on the interdependencies at play. The framework is organized on four facets: state, structure, skills, and strategy. Here we expand on this framework.

4.1 State interdependence

State interdependence refers to interdependence resulting from the need to coordinate and share resources across team members.

We propose expanding this category with two common state types that generate interdependence constraints on the team: information and resources.

4.1.1 Information interdependence

Information creates interdependence based on each agent's need-to-know. Teamwork is built on common ground [2], and so it should be no surprise that team members would need to share information to operate effectively as a team. This need generates information interdependence as each team member experiences their own view of the activity. This type of interdependence is often referred to as a need for situation awareness [33], common ground [2] or shared mental models. Regardless of the phrasing, each implies that discrepant knowledge between agents can lead to poor team performance while consistent knowledge would result in improved team performance. Examples of the type of information teammates depend on are task assignment (who is working on what), task commencement (what has been started), task completion (what has been finished), task outputs, including status (successful or failure) and results (values or decisions).

A key challenge for information interdependence is determining information relevance. As with most aspects of teaming, there are two sides to the issue. The first is an agent recognizing information it receives as relevant and understanding how that information might impact their own understanding of past, present or future decisions and adapting based on the new information. The other side of the issue is an agent understanding when new information it discovers might be relevant to others. This involves being able to identify who is dependent on what information and when. This is made more difficult in fluid teams without fixed roles. Even with fixed roles, dynamic activity means that some information will likely become irrelevant with time and effective teammates should recognize this.

4.1.2 Resource interdependence

Resources create interdependence by constraining what can be done. A person can only carry so much and a robot can only drive to one location at a time. Resource constraints are probably one of the most studied types of interdependence. It is well known that if two activities require the same resource, one can block the other, creating a sequential interdependence constraint [34]. Resources can be things in the environment, like a printer, but the agents themselves can be viewed as a resource as well. For example, person A can help person B carry something, and person A can help person C carry something, but it is unlikely person A can help both person B and C simultaneously, thus creating a sequential interdependence constraint. A key challenge with resource constraints is identifying them and being able to coordinate them effectively as a team.

Information and resources share very similar coordination requirements. One substantial difference is that information can be replicated, usually at low cost compared to physical resources. Once replicated information can then be used in parallel as if there were two of the same resource available. This is one of the many ways to address state interdependence.

4.2 Structural interdependence

Structural interdependence refers to types of interdependence caused by the structure or organization of the work. It comes from two main sources: the taskwork and the team organization [28]. Interactions in highly complex and tightly coupled systems can be difficult to predict. Different level of abstractions are needed at different levels of operation with no holistic understanding of its interdependence. In high risk systems this may lead to catastrophic consequences [35]. Understanding the interdependence resulting from the system itself and its organization is key in identifying potential critical paths and address them adequately.

4.2.1 Task structure interdependence

Taskwork generates interdependence in both the decomposition process and in the synthesis process.

4.2.1.1 Task decomposition

Decomposition of the joint activity generates taskwork interdependence. Structural interdependence is determined by the decomposition boundary of the activity. This boundary is often a design or engineering driven decision. Arguably, tasks can always be further decomposed into sub-tasks but eventually the level of decomposition becomes unwieldy or even absurd. The decision of where the boundary lies often varies with the domain and agents under consideration. The coordination mechanisms involved in a command and control situation are different than those required in a mechanical repair situation and so are the abstractions at play. Goals and requirements may also dynamically change at run time and adaptive teams should be able to adjust task boundaries to provide flexibility in their plans. The process of decomposition itself is not a challenge, it is understanding the implications of how those changes impact interdependence that is difficult.

4.2.1.2 Task synthesis

When tasks are decomposed they must eventually be recomposed, creating interdependence. In distributed problem solving, answer synthesis and behavior composability are critical abstractions of complex distributed systems [24]. Synthesis provides practical mechanisms that address structural interdependence. The key challenge for synthesis to be operationalized is defining how tasks' outputs are generically combined given the range of possibilities.

When decomposing a joint activity, there is a requirement to explicitly define synthesis functions capturing how the output/state of the joint activity is derived from the output/state of the sub-tasks. This allows designers and reasoning engines to understand and exploit decomposition related interdependencies. For example, Boolean logic could be used to define synthesis functions. Consider the activity of going to lunch. It can be decomposed into eating lunch and paying for lunch. Here, the success of going to lunch depends on the success of both children. This type of synthesis can be captured with a Boolean operator such as and; both children have to succeed for the parent to be considered successful. Changing the success synthesis function to *or*, would completely change the meaning of the activity and associated types of interdependence.

4.2.2 Team organization interdependence

4.2.2.1 Team decomposition (roles)

Another way to generate interdependence is through organizational choices. Similar to task decomposition, one's choices about the team structure can create boundaries and interdependence. Distribution of work is another reason why coordination is necessary [13]. For example, having an engineering department and a purchasing department will require the engineering department to go through purchasing for parts, creating a sequential interdependence. Effective organization design typically involves designing roles to reduce the degree of interdependence to allow roles their maximum freedom.

4.2.2.2 Team participation

Some organizations have fixed predefined roles continuously performed by the same individuals, making participation constant and predetermined. Teamwork in general is more fluid, allowing flexible roles and intermittent participation to allow the team to adapt. This is particularly important for teams that do not have the resources to cover all work and may need to choose what is attended to.

Participation in joint activity represents joint commitment, a requirement for teamwork [5, 13]. Participation is often assumed or ignored in system design, but it is an important dimension that plays a critical role in interpreting interdependence. An implementation that is unable to account for participation is blind to key information necessary for effective teaming.

Participation must also account for interdependence in the form of task constraints. The structure and task decomposition choices may limit team composition and participation. In his book *Group Processes and Productivity*, Steiner [36] presented a categorization of joint activity (*group tasks*) along three dimensions, one of which was whether the task was divisible or unitary (*component*). Divisible means the task can be divided and distributed to individuals. The example Steiner gives is a multi-question test, where each question could be given to a different student. Unitary means the task cannot be divided. Steiner uses a test with a unique single question as an example. He posits that, because the question cannot be broken down into sub-questions, this makes this task unitary and that "*the group would be required to* ***work together*** *to discuss and determine the correct answer [. . .]*". A limitation of the unitary category is it does not differentiate tasks that can only be done by a single person. For example, giving a group a single pill that must be swallowed. Only one person can do it and no others can contribute. This is a different type of interdependence than the single question, in which all team members could contribute to the answer.

4.2.2.3 Team synthesis

Simply distributing work creates a need for a synthesis function, similar to task decomposition. The synthesis strategy used is related to the second dimension proposed by Steiner [36], which he characterized as the *interdependence* characteristic of the joint activity. Steiner proposes the categories of, additive (all team members' work contributes to the task - shoveling snow), compensatory (group averaging—averaging weight estimates), disjunctive (single decision—answer to a math problem), conjunctive (all team members must contribute - climbing a mountain as a group) and discretionary which is a combination of any of the previous ones. Each of these synthesis strategies involves a different type of interdependence and different coordination mechanisms.

Practically, team synthesis is different from task synthesis in that it is not about reasoning over children's outputs but rather over multiple outputs for a given joint activity instance. In Steiner's "single question test" example, each student participates in the same joint activity generating multiple, potentially different, outputs to the question. These outputs must be reconciled to produce the unified joint activity output. For example, by using *team operators* (a particular type of team synthesis) on a hierarchical decomposition of joint activity, Tambe demonstrated how selective and efficient communication could be achieved in a distributed environment [37].

Another aspect of team synthesis is the understanding of participation status in joint activities. For example, if a goal is conjunctive, meaning it must be completed by all members of the team, recognizing when a teammate is not capable of participating in the goal (e.g., due to capability requirements or resource constraints) will allow a reasoning process to understand that this sub goal should not be undertaken by anyone or should trigger early failure if it had been started by some agents already.

Team synthesis is challenging because it often involves awareness of several other interdependencies. For example, to accomplish team synthesis effectively, an agent may need to be aware of state information, task structure, and team participation.

4.3 Skill interdependence

While state and structure interdependencies are about being able to identify and understand interdependence, skill is about having the supporting coordination mechanisms to address them. For example, if one is dependent on knowing when their teammate has finished using the hammer, one could employ several mechanisms to coordinate. One could actively observe the teammate with the hammer to see when they put it down (i.e., monitoring). Alternatively, one could ask the teammate to provide a notification when complete. Both mechanisms are effective each with their own advantages and constraints. Each mechanism requires specific skills or abilities to be successful. For example, monitoring only requires effort by the person doing the monitoring and alleviates the burden from the one being monitored. The disadvantage is that it can require significant attention, possibly reducing team productivity. It also creates a single point of failure. Notification requires more coordination effort by both parties, but frees each from the monitoring burden, potentially allowing better use of time.

While there are potentially an endless number of coordination mechanisms, many can be categorized as being able to recognize the existence (or lack of) interdependence between one or more parties, understand the communication or behavior pattern necessary to manage that interdependence, and the means to execute it. This means recognizing when observed changes in the environment are relevant to others on the team and sharing them (information), recognizing someone is constrained to doing one task at a time and providing assistance (resource), recognizing that tasks have sequential interdependence and providing the waiting party notification of completion (decomposition), sharing task results (synthesis), and notifying only those relevant to the activity (participation). These pattern generalizations are how people can leverage teamwork skills in new situations.

4.4 Strategy interdependence

Teaming strategy is about having the competency to discern how and when to engage a coordination skill to impact a state or structural interdependence in order to improve some quality within the team. Effective teamwork involves trying to improve behavior qualities. This aligns with Steiner's third category of coordination challenges: *focus* [36]. Steiner provided only two discrete categories: maximizing (improving throughput) or optimizing (improving quality). Decker [12] generalized this concept by introducing a *quality* to tasks as an abstract representation of the task's *focus* as well as a task's *duration* as one of its prime characteristics. A key challenge with focus, and strategy in general, is that it often varies based on circumstances and rarely can be set in stone *a priori*, hence it is not part of the JAG formalism, but a runtime consideration of that formalism.

5 Evaluation domain

We desired to have an evaluation domain that exercised the broad range of types of interdependence described in Section 4. As part of the Defense Advanced Research Projects Agency (DARPA) program called CREATE (Context Reasoning for Autonomous Teaming), we developed a new evaluation domain. The goal of CREATE was to investigate new decentralized teaming approaches for physically distributed groups of agents. The program's focus was on solutions that demonstrated "context reasoning", enabling agents to be resilient to uncertainty and adapt to unexpected events in the absence of centralized control. This provided a perfect test case for operationalizing interdependence design principles. We chose to base our evaluation domain on Capture-the-Flag (CTF). CTF is a dynamic adversarial game that has many of the desired characteristics that demand complex teaming, in particular many of those discussed in Section 4.

One limitation of traditional CTF is that it is mainly disjunctive activity (e.g., shooting, carrying the flag). It was desirable to have an evaluation domain that exercises a broader range of activity types. We looked to enhance the CTF domain leveraging Steiner's interdependence categories [36]. Some domains only provide additive tasks (e.g., foraging, search), others only provide disjunctive tasks (e.g., image recognition, decision making), while others are solely conjunctive (e.g., carrying a large table together).

Our new version of CTF has unique rules that foster a wider variety of teaming activities to better exercise different interdependence requirements. It consists of adversarial teams composed of heterogeneous agents: unmanned aerial vehicles (UAV) and unmanned ground systems (UGS). The objective is to find the enemy's flag and bring it back to your team's base (color coded endzones in Figure 2). A UAV can find and pick up the enemy flag, and deliver it to their base (disjunctive task). UAVs can also pick up and move UGS, which cannot move on their own. UGS are non-mobile smart mines that can suppress the enemy UAVs, sending them back to their base. UAVs can deploy UGS (additive task) as a defensive tactic. Instead of shooting each other as in traditional CTF (disjunctive task), the UAVs can temporarily suppress one another, sending them back to their base. UAVs achieve this by outnumbering the enemy players (conjunctive task). The visual range of the UAVs and UGSs were restricted to increase the value of sharing information between team members. This combination of activities required teams address a broader range of interesting teaming challenges than traditional CTF.

Specifically, our modified CTF domain exercises all of the types of interdependence described in section 4. For instance, the addition of the mine laying task created information interdependence (Section 4.1.1) with regard to where to lay mines and resource interdependence (Section 4.1.2) to coordinate who would lay each mine. There is variety in the activity decomposition (Section 4.2.1.1), as the main activities (retrieving flag and laying mines) can be completed in parallel,

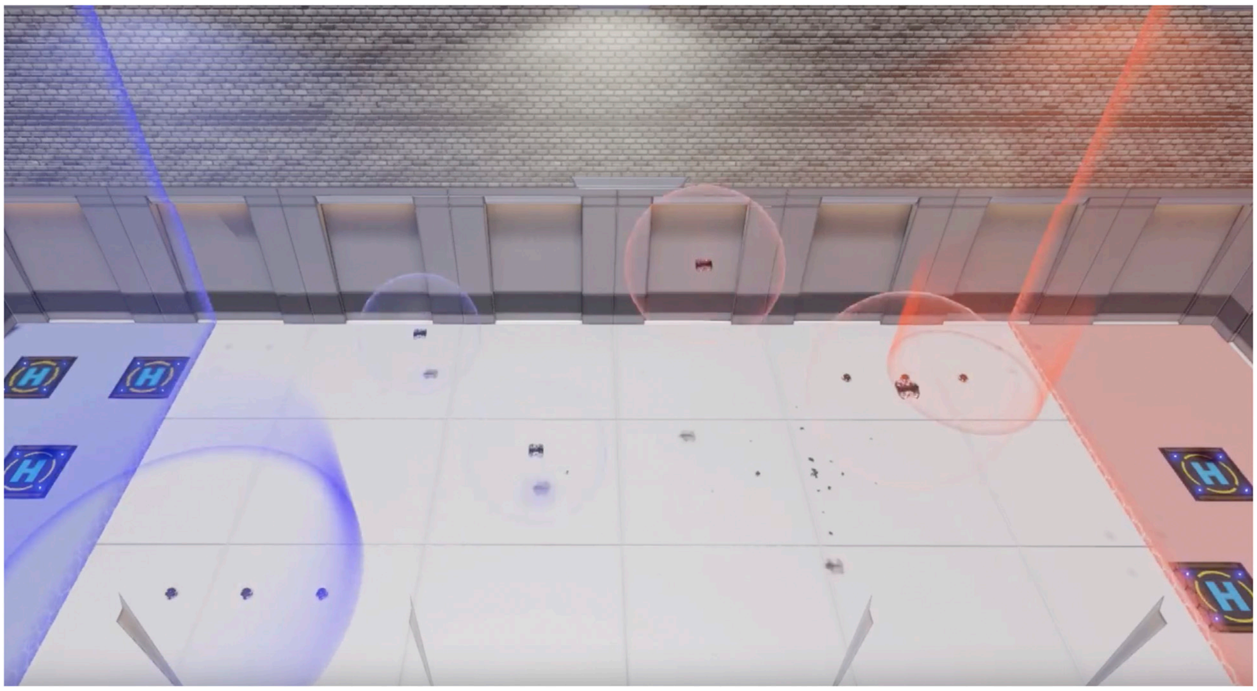

FIGURE 2
Live game of Capture-the-Flag in the simulated lab arena. This game shows a 3v3 (technically a (3 + 3)v (3 + 3) with 3 UAVs and 3 UGSs per team) with the blue team endzone on the left and the red team endzone on the right. At this point in the game, 1 UAV in each team has been disabled.

while the sub-activities of each have sequential dependence (e.g. pickup before delivery). The task synthesis requirements (Section 4.2.1.2) vary as some tasks can be done asynchronously, like mine laying, while other tasks must be done conjunctively, like UAV suppression of the enemy. The team must chose how to balance offensive and defensive strategies (Section 4.4), which directly impacts participation (Section 4.2.2.2). Team members can dynamically change roles creating team organizational interdependence (Section 4.2.2.1). The team's strategy must account for how the combined efforts of each individual result in effective behavior (Section 4.2.2.3). These are only a few of the many instance of interdependence that must be managed to produce effective coordination by the team. Each requires possessing the coordination skill (Section 4.3) and an understanding of information relevance (Section 6.3) to support *good teamwork*.

To exercise our framework, we developed hardware agents (see Figure 3) as well as a virtual twin simulator in unity (see Figure 2) that allowed the development and validation of joint activity graphs both in simulation and hardware in a fully distributed environment.

As a dynamic and uncertain evaluation environment, our modified CTF fosters a broad range of interdependence demanding a rich understanding of team context to produce effective team performance. Although CTF is a very active domain that involves fast-paced physical work, it also requires a large amount of sophisticated cognitive reasoning over team context. This reasoning is complicated by the fact that it happens within each individual agent in a distributed manner. These independent decisions must be synthesized and coordinated across the team, providing an excellent evaluation domain for assessing our interdependence design principles in practice.

6 Addressing interdependence

The JAG formalism was developed to help designers think through the considerations necessary when designing joint activity. It directly supports addressing state and structural interdependence. It also provides the teaming context needed to address skill and strategy interdependence within a team. This is accomplished by the JAG Engine reasoning over the JAG formalism to make coordination and strategy decisions, discussed further in Section 7. As conveyed in 4, the various types of interdependence relate to one another in many ways, so there is not a one-to-one-mapping to the JAG formalism. Instead, the elements of the JAG formalism combine in different ways to help address all of the interdependecies in 4.

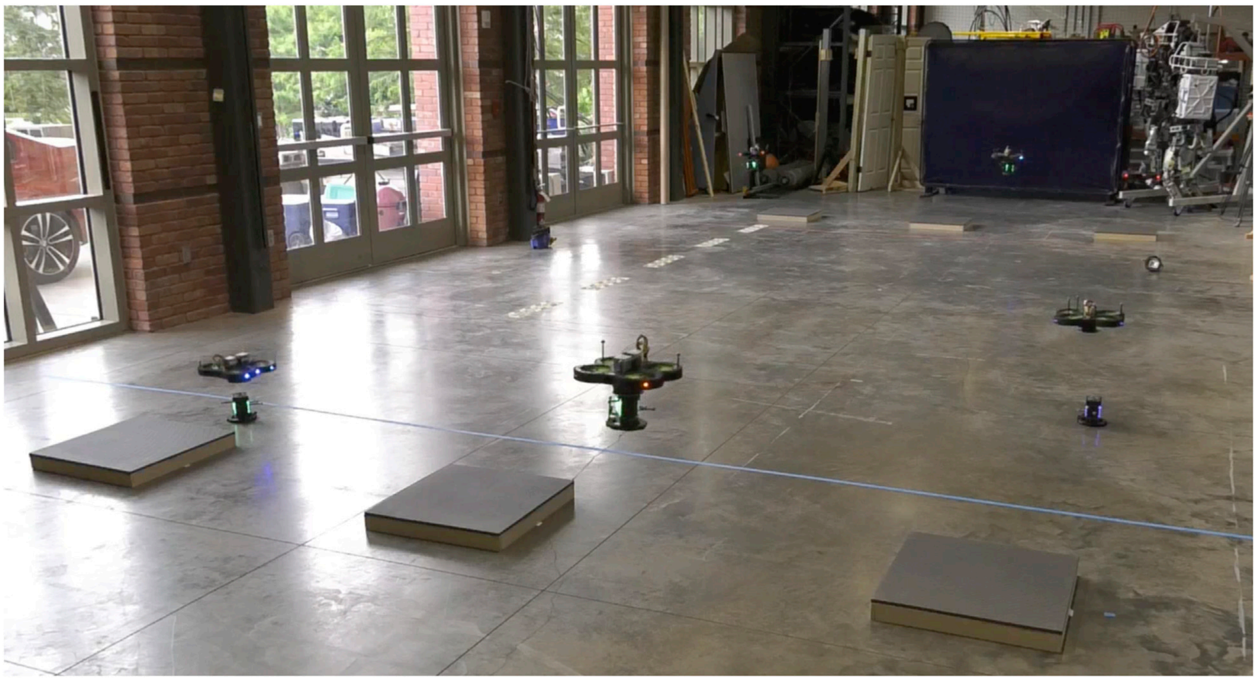

FIGURE 3
Live game of Capture-the-Flag in the lab arena.

6.1 Decomposition

JAG's task work component, $\gimel$, is a hierarchical decomposition of the joint activity. It defines the activity search space for the agents and is consistent with Durfee's distributed goal search [27] and Smith's synthesis requirements [24]. The main purpose of hierarchical decomposition is to understand task work context.

As an example, Figure 1 shows two different levels of decomposition: $\gimel$ decomposes into $\gimel_a$ and $\gimel_b$. $\gimel_b$ has no further decomposition whereas $\gimel_a$ is further decomposed into $\gimel_{a,1}$ and $\gimel_{a,2}$. This decomposition has implications both in terms of participation and interaction.

Agents select what to do next through an understanding of the activity space as defined by the decomposition. Additionally, activity decomposition provides a structural skeleton for tracking participation of the entire team. Each agent's participation in joint activity can be tracked at the individual hierarchy level. This helps scope interactions enabling level specific coordination mechanisms. For instance, communications about sub-tasks do not need to be broadcast to the entire team but only to the agents participating at that level of the hierarchy (see Section 6.3). Similarly roles and responsibilities can be defined at each individual level of decomposition.

Decomposition also allows designers and agents alike to define and act at different abstraction boundaries. Consider a grab behavior defined as $\gimel_{grab}$. On one hand, a human could undertake the $\gimel_{grab}$ behavior as a 'primitive' limiting observability, predictability or directability into the task. This would prevent team members from interacting with the different parts of the process involved in the grab behavior. A machine, on the other hand, may decompose its $\gimel_{grab}$ behavior further to allow team members to interact, contribute and/or support the different sub processes at play within the machine during the activity.

There might be practical reasons for relying on higher abstraction levels. For instance, humans can grab things pretty reliably whereas current machines may need more support throughout the whole process such as finding the location of the object or determining the best approach trajectory. Pragmatically, the level of decomposition drives the abstraction boundary of the behavior and in turn the type of interdependence and the capabilities needed to manage it. The JAG approach allows both design time and runtime flexibility for such boundaries, facilitating human-machine joint activity.

Decomposition has other intrinsic benefits common to all similar approaches, such as the creation of modular behaviors and the promotion of reuse of existing designs. Since JAG designs have interdependence considerations defined with the decomposition, those consideration transfer with reuse.

By using a hierarchical structure, JAGs support the ***task decomposition*** (Section 4.2.1.1) like similar approaches (see

Sections 2.1 and Section 2.2). However, JAGs go further and support ***structural interdependence***, specifically ***team decomposition boundary*** (Section 4.2.1.1) and ***team participation*** (Section 4.2.2.2), the importance of which will be further discussed below.

6.2 Synthesis function

"*Our ability to decompose a problem into parts depends directly on our ability to glue solutions together*".

- John Hughes, Why Functional Programming Matters [38].

Synthesis defines the process of recomposing the task decomposition and its results. The synthesis function $s_{ℷ}$ can take the form of any mathematical function. Examples include quality functions min, *mean* or max [12, 36] (dealing with conjuntive, disjunctive or additive aspect of tasks) as well as Boolean operator [37] such as *and* or dealing with team goal requirements. As such, joint activity graph synthesis can benefit from contributions from a wide variety of fields such as sensor fusion and organizational theory.

Even though this synthesis function can be arbitrarily complex, a broad range of activity can be covered by a reasonably small set of reusable joint activity patterns. We expect each domain will favor specific sets of functions with significant overlap. For example, in our modified CTF domain, all but one operators were standard Boolean operators.

Importantly, the synthesis function also acts coherently with leaf nodes, also called primitives [13, 23, 25, 32] or methods [39]. A leaf node is a jag ℷ whose set of children $J_{ℷ}$ is empty. As $\bigcup_{n=1}^{|J_{ℷ}|} O_{ℷ_n} = \varnothing$, its synthesis function $s_{ℷ}$ is then reduced to:

$$I_{ℷ} \overset{s_{ℷ}}{\mapsto} O_{ℷ}$$

A leaf node's synthesis function $s_{ℷ}$ essentially acts on its own inputs, then outputs a result and potentially generates non-local effects as defined by Decker [39]. This is essentially a function call to a machine or human interface. This synthesis function coherency is an important distinction from classical planning and behavior modeling. It allows joint activity designers and exploiters to consider and interact with all levels of abstraction in the same manner.

Synthesis function definitions allow JAGs to formally capture the processes needed to manage the ***synthesis interdependencies*** described in Sections 4.2.1.2 and Section 4.2.2.3. A surprising number of tools and techniques ignore synthesis, even though it is critical to teaming. JAGs provide a general and extensible solution to address synthesis interdependence.

6.3 I/O and information relevance

The inputs $I_{ℷ}$ to a joint activity ℷ provide the necessary information needed by the activity. They are a common way to parameterize activities.

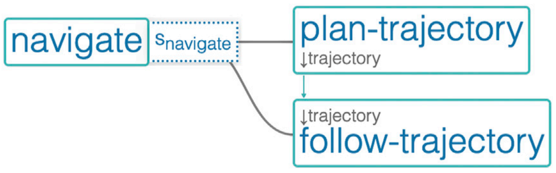

FIGURE 4
A simple jag behavior with input and outputs.

The outputs $O_{ℷ}$ of a joint activity ℷ are derived from ℷ's inputs and the outputs of ℷ's children *via* ℷ's synthesis function $s_{ℷ}$. The synthesis function can be a simple pass-through, return one or more child outputs, or can be an arbitrarily more complex function returning a derived result from one or more child outputs. This is consistent with and supports concepts such as Decker's sub-task quality accrual functions (min, max average) [12], but a more general extension.

Bindings, $B_{ℷ}$, define the information flow within an activity. Bindings uniquely identify a data provider and a data consumer. Inputs for an activity can be passed down and consumed by (bound to) any child joint activity. Sibling outputs can also be consumed as inputs by other siblings. For example, Figure 1 shows $ℷ_{a,1}$'s output $o_1^{a,1}$ bound to $ℷ_{a,2}$'s input $i_1^{a,2}$.

This creates an implicit sequential interdependence requirement [34]; $ℷ_{a,2}$ cannot be started before $ℷ_{a,1}$ has completed and generated its output $o_1^{a,1}$. Input and output flow is completely defined, in practice, through bindings.

I/O plays a key role in identifying information relevance. Team performance monitoring is one of the *Big Five* components of team effectiveness [40] and is crucial in enabling adaptability. It is common to use monitoring functions to observe, prevent failure, and repair plans through continuous planning [41]. However, monitoring functions have to be manually defined and managed which can be cumbersome. We propose that we can make the process more observable and systematic with parameterization of behaviors and explicit data flow to address ***resource and information state interdependences***. The data used by joint activities is inherently relevant to that activity. If the input changes the output may change as well. Hence, data flow identifies what portion of the world is relevant at different levels of the joint activity. In turn, team processes addressing ***information interdependence*** can be executed based on this flow. These processes help agents identify to which teammate a new piece of information is relevant. They also help agents assess if a received piece of information is relevant to their own ongoing activities.

Similar to good software engineering practices, we have found that the amount of behavior parameterization is directly proportional to the adaptability the team with regard to that behavior.

For instance the behavior $ℷ_{navigate}$ in Figure 4 behavior can be implemented very specifically as to only be able to navigate to a predefined location. Without input, this behavior cannot react to information updates. There is also no observability into the information necessary to execute navigate. Two independent executions will behave the same, and make managing certain interdependencies impossible, and by extension, teamwork that much worse. A slightly more common implementation would be to parameterize $\gimel_{navigate}$ with a location which would be consumed by $\gimel_{plan-trajectory}$. Updates about the location would now be known to have an impact on $\gimel_{navigate}$. Similarly, navigation is likely to include a list of obstacle in its planning. If that list of obstacles is a parameter (an input), then $\gimel_{navigate}$ can now react to new information about obstacle location (see Section 8.2). Relevance of new information, such as the information about the destination and obstacles, is now systematically tied to the navigate behavior which leads to smart and informed reactions to world changes. Relevance can now be defined more specifically:

Information *p* is relevant to a behavior *b if b is active* and *if p matches any input from b or from a behavior whose output is recursively consumed by b.*

For example, $\gimel_{follow-trajectory}$ consumes $\gimel_{plan-trajectory}$'s output, $o^{plan-trajectory}_{trajectory}$ which was generated using $i^{plan-trajectory}_{obstacles}$. Any change to an obstacle concept would therefore be relevant to the $\gimel_{follow-trajectory}$ joint activity.

Because there is a systemic link between data and the behavior that uses this data, the more a behavior can be parameterized the more it can be reactive to changes in its parameters. This awareness of information relevance can facilitate better team adaptation. This applies generically throughout the joint activity as defined by its data bindings.

Although outside the JAG formalism, the concept of matching information was an important part of building an agent knowledge base. The process of matching should be left to the system designer to decide but it may be useful for the reader to understand how we designed information and implemented concept matching in our agents. Our approach was soft property matching. Concepts (or pieces of information) are a bundle of arbitrary property value pairs. If all properties of a concept *c*1 exist in another concept *c*2, and both values satisfying equality for their type then *c*1 matches *c*2, however the inverse is not true. For instance, an agent referring to a *blue mine* would match the generic friendly unarmed mine concept in listing 1 and the more specific mine instance in listing 2. However, if an agent refers to a specific blue UGS, that agent is not referring to just any blue UGS. This is analogous to looking for one's favorite blue pen that was gifted when graduating as opposed to looking for any blue pen.

Listing 1. Friendly unarmed mine concept.

```
{
"type": "agent:mine"
"team": "blue"
"armed": false
}
```

Listing 2. Specific mine instance concept.

```
{
"type": "agent:mine"
"team": "blue"
"id": "542ce2b1-c00e-47ff-8d7f-8db0fc118b13"
"name": "blue-mine-3"
"armed": false
"location": (0.0, 0.15, -1.5)
}
```

7 Dynamic team context reasoning

While the JAG formalism helps designers consider interdependence *a priori*, other types of interdependencies only manifest themselves during joint activity execution. The JAG Engine provides reasoning over the JAG formalism to make coordination and strategy decisions.

7.1 JAG engine

In order to operationalize the management of interdependence we developed an additional tool called a JAG Engine. A JAG Engine is an execution environment that interprets and executes joint activity graphs. It is able to use the information captured by the formalism described in Section 3 to drive the behavior of an agent in support of teamwork. It exposes interdependencies and provides processes to manage them. The JAG Engine interfaces with the agents being supported *via* traditional application programming interfaces for artificial agents and user interfaces for human agents. This engine uses user defined strategies to drive the behavior of agents following the process models defined in joint activity graphs under execution. The engine understands the intrinsic interdependencies in JAGs, such as the fact that multiple agents can participate in additive tasks, or that new information may be relevant to specific agents based on their joint activity participation. It is worth noting that in the case of human agents, the processes can be exposed *via* user interfaces allowing human agents to interact with the proposed courses of action in the same way artificial agents would (e.g., accept, reject or counter propose).

The JAG Engine, combined with the JAG formalization provides a unique capability for system control, enabling flexible and even dynamic shifts in control. JAG Engines have a 1 to n relationship with agents. They can be distributed (one engine per agent), centralized (one engine for all agents) or anything in between (*k* engines for *n* agents where $1 \leq k \leq n$). The engine specification also provides an abstraction layer for communication with built-in communication options relevant to teaming processes (e.g. participation in a JAG, completion of a

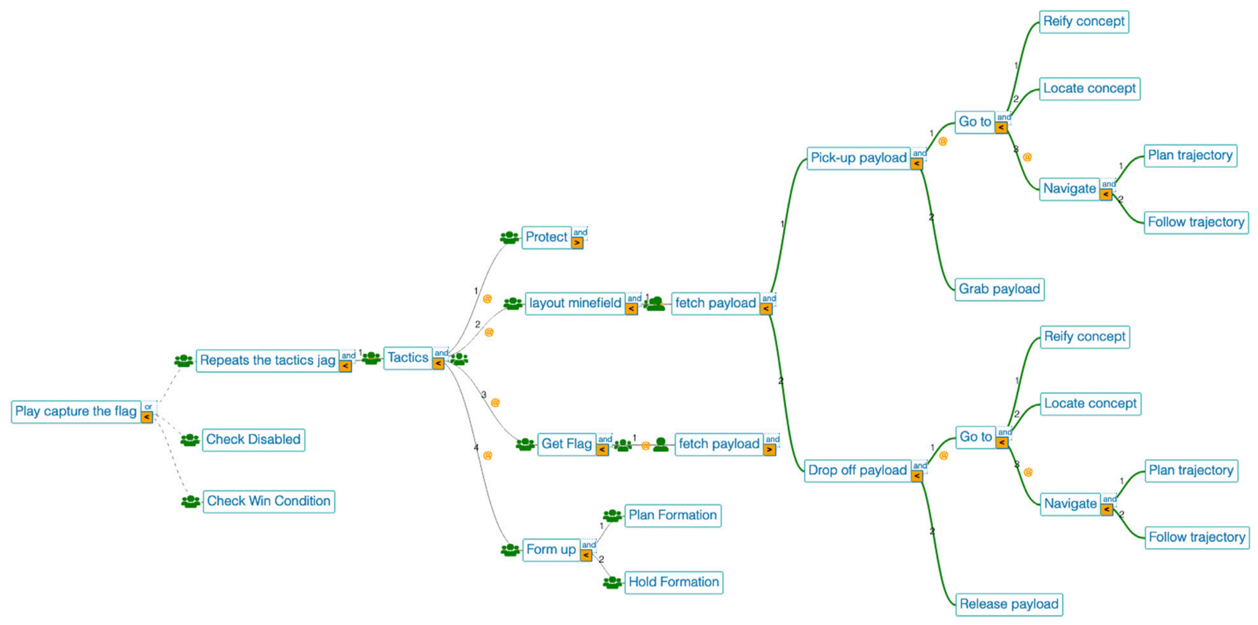

FIGURE 5
Joint activity graph for the Capture-the-Flag scenario - semi expanded.

JAG, negotiations relevant to the strategy in use, etc.). The specification allows for flexible strategy implementations allowing system designers and subject matter experts to create domain appropriate decision-making systems that can dynamically ingest and act on the run-time teaming context.

A JAG engine implementation provides the necessary framework to interpret JAGs, drive team behavior using strategies, comply with and expose interdependencies requirements and opportunities, and interface with a team of heterogeneous agents. The JAG Engine tracks and coordinates participation, as well as enabling strategy to be informed by the teaming context of the JAG. Figure 5 shows a semi expanded version of a Capture-the-Flag joint activity graph. Such graph is a view on joint activity definitions and is directly executable by a JAG Engine.

7.2 Participation

Participation in a joint activity plays a key role in managing communication backed coordination mechanisms such as information sharing, as well as decisions about task allocation. Yet common techniques often ignore participation (see Section 2.1 and Section 2.2). Without rigidly defined roles, it is unclear how proper teamwork can be achieved without an understanding of participation.

For example, consider the JAG defined in Figure 1. $\gimel_{a,2}$ requires input $i_1^{a,2}$ from the output $o_1^{a,1}$ of $\gimel_{a,1}$. The agent participation in $\gimel_a$ will contribute to restricting sharing of $\gimel_{a,1}$'s result with only agents participating in jag $\gimel_a$ and not with agent participating in $\gimel_b$. The information interdependence exists locally and no higher than $\gimel_a$, thus agents not involved in this subspace of the joint activity probably do not need to know about $\gimel_{a,1}$'s result.

One key nuance we have encountered, which is lacking on most approaches, is joint activity instance tracking. Let's consider the joint activity $\gimel_{deploy-ugs}$; Two agents can participate in the same joint activity instance, (both are working together to deploy ugs on the left side of the field) or they can work on two separate instances of the same joint activity (agent A deploys UGS on the left, and agent B deploys UGS in the center of the arena). Instance tracking together with participation proved to be key in differentiating intent and ***team organization interdependence***. This has ramifications for strategy and information sharing as well.

7.3 Strategy

In most real world problems that involve teamwork, there are usually multiple ways to tackle the problem, each with different costs and benefits. For teams to be successful, they must have some goal alignment to ensure the team members are utilizing compatible strategies (see Steiner [36] and Decker [39]). This is another aspect of interdependence that manifests itself at run time: team *focus* Section 4.4.

We consider that the focus of a activity cannot be statically set for all teamwork scenarios and thus designed joint activity graphs to support a multidimensional representation of the activity's *focus* (or foci). It is of note that an activity's foci is not defined in the taskwork but rather defined outside of taskwork as a strategy parameter that can dynamically change. For instance, a team may decide to focus on speed while another on quality. Often teams will have multiple competing foci and balance them at runtime. This is essential for reusable team behaviors across different strategic approaches. This multidimensional abstraction of the focus represents the agents interests in task quality (e.g., speed, accuracy, quantity, etc.) and can inherently be specified at individual levels in the task work and per agent. This allows our structure to support more recent work on preferences, [42] and consider concepts such as commitment [13] as an agreement on the work to be done (taskwork) and the foci to work towards.

Team participation, a type of team organization interdependence, is crucial to information relevance as well as task and role allocation. We know that agents often work together with the goal of improving some joint activity focus (e.g., speed, accuracy, resource consumption). Strategy interdependencies help understand and address the varying, potentially conflicting or synergistic, foci at play during execution of the joint activity, thereby addressing ***strategy interdependence***.

8 JAG supported adaptation

The Capture-the-Flag domain described in section 5 allows us to exercise a broad range of teaming challenges and operationalize interdependence design principles to show adaptability to the environment, the team and the joint activity.

We ran teams of heterogeneous agents from size 5 (2 UAVs and 3 UGSs) to size 23 (20 UAVs and 3 UGSs) against each other in our virtual environment. Due to safety and space constraints, we only ran 5v5 and 6v6 games on hardware. All these games were run using the exact same JAG shown in Figure 5. Teams were able to adapt and coordinate independent of scale (see video *ctf-scale* in Supplemental Video S1) addressing interdependencies described in Section 4.

Our tools enable systematic identification and management of interdependence through its formalism. ***Decomposition interdependencies*** are handled by the joint activity natural hierarchical structure through jag children. ***Resource and information state interdependencies*** are captured by joint activity data flow definition in combination with participation awareness. ***Task and team synthesis interdependencies*** are reflected through each joint activity synthesis definition also in combination with participation awareness. ***Skill and strategy*** are exposed and addressed at run time by the JAG Engine and user defined strategies.

8.1 Structural adaptation

Agents were able to reason over task allocation using decomposition and strategy interdependence, and participation status.

For instance, agents would dynamically re-prioritize their behavior to go after the enemy flag if and when they realized there was no agent currently participating in that section of the joint activity. This would happen when offensive agents would get suppressed on their way to the enemy flag.

Agents would also understand whether they were participating in the same joint activity instance (such as A and B laying down UGS on the right side together) or in different instance of the same joint activity (such as A laying down UGS on the right and B also laying down UGS but in the center).

8.2 State adaptation

Agents were able to resolve resource constraints using information and participation interdependencies. For example, two agents would often try to deploy the same UGS. Using participation status they were able to identify the need for negotiation which would lead one of the agents to reevaluate its activity to go after a different UGS. In our strategy, we used first come first serve and distance based costs as negotiation processes. However, it is important to note that the specific negotiation process is less important than the identification of the need for negotiation within context. Agents were able to quickly adjust to new information whether it was a new location of the flag or the enemy (which would automatically trigger planning of a new path) or the fact that one's own flag had been grabbed (leading to re-prioritization of behavior to intercept the enemy with the flag). In a dynamic and information rich environment such as the CTF domain, information sharing and observation are a significant source of knowledge update.

Early in design we were confronted with the "artificial" dichotomy of information provenance. There were two distinct but similar pathways for an agent to ingest information depending on whether it was observed by local sensors (vision) or received through communication by team members. We realized that the source of information could instead be a characteristic of the piece of information received and that there was no need to distinguish them in processing. Team members can be thought of as sensors, and the information received can be characterized accordingly based on the sensor (teammate) and transport medium characteristics. This makes dealing with cognitive activities such as reifying information simpler, robust to failure and often elegantly handled.

By associating characteristics to each information provider (sensors, teammates) such as latency, accuracy, reliability, an abstraction can be made over the reception of information which does not need to distinguish local vs. remote information, and makes handling reaction to change simpler, more consistent and elegant. Often, assumptions about local sensors are made which may hide characteristics of the transport medium and source, and leads to unnecessary special handling. In Capture-the-Flag's agent design, we successfully removed special information processing based on the source or transport medium in favor of information characterized along the dimension of interest. As such, reacting and adapting to new information behaved completely independently of its provenance and transport which allowed, for example, agents to re-plan their trajectory around enemies that were not in their vision range but in range of a teammate (UGS or UAV) somewhere else on the field. Team members know what information may be relevant, enemy location in this example, because data flow and participation indicates what information is in use at any given time (see Section 4.2.2.2 and Section 6.3). This happened **without us, designers, having to make any specific behaviors or adjustments to existing behaviors**.

Accessing agent knowledge is part of the activity and allows situations to fail gracefully. For instance, if getting the location of a resource is a joint activity, one agent can fail to complete the activity which can then be completed by another agent without special consideration. The activity of generating the location for a resource can be completed by all members of the team and synthesis of the answers can applied to that activity the same way they are applied to *physical* activities. It also ties knowledge use to activities which in turns enables adaptability (described in Section 6.3) independent of the knowledge provenance.

8.3 Strategy adaptation

Reacting to new information often means re-evaluating activities under execution. Whether it is because they are no longer relevant, or because they need to be restarted with a different parametrization, tasks need to be interrupted. That said, not all tasks can be abruptly interrupted without further considerations. Designing interruption as a first class system within our framework proved to be an important requirement.

Two main concepts need to be considered: partial results and interruption procedures.

With regards to partial results, there already exists a substantial body of research, of which we were able to take advantage: namely anytime algorithm and its derivatives [43, 44]. Being able to produce partial results is an important consideration when designing adaptable joint activities. Partial results, may influence characteristics of the results (e.g. accuracy) and as such can be processed by synthesis without special consideration.

Some tasks may need to execute a clean up procedure before they can interrupt a behavior (such as release a constraint on a resource). The most blatant example of a need for interruption procedures was delivery of UGS. Initially naively defined, the transport of objects (UGS or flag) proved to be an interesting scenario demonstrating how failing to handle interruption clean up may lead to failure. While in the middle of deploying a UGS, the suppression of the UAV attempting to retrieve the flag, triggered another UAV to re-evaluate current priorities of active task and switch to go after the flag. Still carrying a UGS, the UAV was unable to successfully grab the enemy flag but kept trying without knowing how to "clean up" the previous behavior. This type of situation can be really insidious and can be a common engineering problem in structures such as behavior trees that get constantly reevaluated. Conversely, the clean up can be a requirement of starting a task as well and in that aspect is consistent with pre-conditions and post-conditions in classical planning. For example, if one fails to release a piece of tape, the clean up may be about succeeding at the failed task (failure to release the tape and must try again before moving on - post-condition) or the clean up may be about having a "hand" free to grab something else as part of the subsequent task (need to grab something different and must succeed before undertaking the next task). Interruption is an inherent part of adaptability in unpredictable environments and even more so in human machine teams when opportunities and conflicts have a tendency to arise. In that respect joint activity graphs enable event driven interruption to understand and use contextual information (such as current participation, data flow, interruption's partial result) about the activity at hand to clean up adequately.

In an execution driven environment such as CTF (where the joint activity graph drives the behavior of the agents), agents default to participation in all nodes. Capability, task type, or resource constraints may restrict participation in joint activities. For instance, dropping of a UGS is reserved to the agent carrying said UGS. Note that the trajectory planning section of the drop off joint activity would not be restricted and as such, all agents, including UGS, can contribute a trajectory result (which is valuable as they may have information that other agents would not have - which indeed happened when UGS were used as scouts). An instance of capability restriction would be activating the grab mechanisms, which is restricted to agents capable of grabbing (not UGS).

9 Future work

In this paper, we provided a joint activity graph formalism (Section 3) to capture the key design elements necessary for effective teaming. We also described the JAG Engine (Section 7.1) as a reasoning engine to interpret the JAG formalism and drive the behavior of individual distributed agents working together as a team. In other words, JAGs provide an understanding of team context that enables generation of cooperative team behavior. However, that understanding could be utilized in others ways. Two alternatives include using the JAG representation to support inference and prediction of human team behavior and using JAG representations to support inference and prediction of adversarial team behavior.

In an ongoing project called DARPA ASIST (Artificial Social Intelligence for Successful Teams), we have had some initial success using JAGs to build a mental model of a human team's joint activity in the search and rescue domain. The goal is to use that modeling to support an artificial social intelligence observer to use signals from the team members to build a partial mental model of each participant. Through this JAG-based dynamic model of the team, we aim to enable the artificial social intelligence to generate prediction and potentially intervene to prevent errors, help repair common ground, or simply improve team processes and performance. In a similar way, JAG engines can be used operationally as a real-time C2 decision-aid or for real-time monitoring of the multi-agent behavior.

We are also investigating using the same approach used during the DARPA CREATE program in support of cooperative teams to explore its potential with adversarial teams. Still in the context of Capture-the-Flag, we are working on integrating a way for agents to dynamically track the joint activity of the enemy team using observations of the enemy's actions as signals. This is similar to process of using JAGs to model team behavior, as described for ASIST. However, ASIST is using a single agent, and this work needs to perform the assessment across distributed agents. The challenge is enabling a distributed team of agents to build a pragmatic mental model of the enemy's joint activity and then use that model to make predictions about their intentions. This would allow the team to deploy counter-measures or take other actions to opportunistically gain an advantage.

This future work aims to show that joint activity graphs are an effective structure to capture and track active contribution to tasks, helping to predict team behavior, assess team efficiency, identify team breakdowns, and generate interventions to improve team performance.

10 Conclusion

In this paper we have presented a new formalism, joint activity graphs, as a tool to design joint activity. We have also introduced the JAG Engine as a tool to interpret JAGs at runtime, driving agent behavior. Together these tools enable human-machine systems to manage and exploit the interdependence within the team through the systematic use of joint activity graphs. By providing support for understanding teaming context, JAGs provide an rigorous and systematic approach to effective human-machine team performance.

In Section 3 we presented a formal structure, joint activity graphs, that systematically guides the architecture of human-machine team systems to address these challenges. JAGs assist designers in the design of joint activities and provide shared contextual information at run-time that supports coordination processes, enabling team members to manage their interdependencies with teammates. Specifically, it provides structures to capture hierarchical decomposition, handling of task interdependence, agent participation, sequencing processes, data flow and the synthesis necessary for activity recomposition.

In Section 4, we describe the broad range of teaming challenges in terms of types of interdependence necessary for adaptive teamwork.

In Section 7.1, we introduce the JAG Engine. Its purpose is to ingest and execute joint activity graphs providing the context necessary to recognize when interdependencies arise and their nature: operationalizing their management through adequate coordination processes.

In Section 6 and Section 7.1 we describe how the two tools provide support for the broad range of interdependencies in Section 4.

In Section 8, we described how these principles of joint activity design were applied in the concrete domain of Capture-the-Flag to provide highly adaptive team behavior. Teams of distributed agents were able to do at least as well as fully centralized teams and were more resilient to breakdowns in communication, agent failures and dynamic team re-composition (such as the loss of a member). This demonstrates that adequate identification and management of interdependence allows teams to better understand information relevance [5], handle and recover from coordination surprise, and continuously repair common ground. The result of this adaptability is effective team performance that can be described as *good teamwork*.

We hope the formally guided approach to human-machine team design presented here proves useful to others working toward complex adaptable teams. The approach is supported by principles, guidelines and tools that can help designers develop systems that support effective management of interdependence in order to achieve flexible and adaptable teamwork in human-machine systems.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

MV and MJ designed and developed the framework presented in this paper and contributed equally to the writing of the manuscript. All authors contributed to the design and development of the experimental testbed and system implementation. All authors contributed to manuscript revision, read, and approved the submitted version.

Funding

The work in this paper was partially supported by the following grants Title—ASIST Predicting Effective Performance in Teams (PEPT) TA2 Funding Agency—DARPA ARO Agreement #—HR001120C0037 Title—JACKPOT: Joint Activity and Contextual KAOS Policies for Operational Teaming Funding Agency—DARPA Agreement #—HR00112090024. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contracts No. HR00112090024 and No. HR001120C0037. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphy.2022.969544/full#supplementary-material

SUPPLEMENTAL VIDEO S1
Capture-the-Flag at scale.

References

1. Castelfranchi C. Modelling social action for AI agents. *Artif Intelligence* (1998) 103:157–82. doi:10.1016/S0004-3702(98)00056-3

2. Clark HH. *Using language*. Cambridge [England]New York: Cambridge University Press (1996).

3. JC Beck, O Buffet, J Hoffmann, E Karpas, S Sohrabi, editors. *Proceedings of the thirtieth international conference on automated planning and scheduling*, 30. France: AAAI Press (2020).

4. Demir M, McNeese NJ, Cooke NJ. Team synchrony in human-autonomy teaming. In: J Chen, editor. *Advances in human factors in robots and unmanned systems*, 595. Cham: Springer International Publishing)Advances in Intelligent Systems and Computing (2018). p. 303–12. doi:10.1007/978-3-319-60384-1_29

5. Klein G, Feltovich PJ, Bradshaw JM, Woods DD. Common ground and coordination in joint activity. In: WB Rouse KR Boff, editors. *Organizational simulation*. Hoboken, NJ, USA: John Wiley & Sons (2005). p. 139–84. doi:10.1002/0471739448.ch6

6. Johnson M, Bradshaw JM. The role of interdependence in trust. In: *Trust in human-robot interaction*. Elsevier (2021). p. 379–403. doi:10.1016/B978-0-12-819472-0.00016-2

7. Ilgen DR, Hollenbeck JR, Johnson M, Jundt D. Teams in organizations: From input-process-output models to IMOI models. *Annu Rev Psychol* (2005) 56:517–43. doi:10.1146/annurev.psych.56.091103.070250

8. March JG, Simon HA. *Organizations (cambridge, mass*. 2nd ed. edn. USA: Blackwell (1993).

9. Malone TW, Crowston K, Toward an interdisciplinary theory of coordination. In: *Center for Coordination Science, Sloan School of Management, Massachusetts Institute of Technology*. Tech Rep CCS (1991).

10. Johnson M, Bradshaw JM. How interdependence explains the world of teamwork. In: WF Lawless, J Llinas, DA Sofge, R Mittu, editors. *Engineering artificially intelligent systems*, 13000. Cham: Springer International Publishing)Series Title: Lecture Notes in Computer Science (2021). p. 122–46. doi:10.1007/978-3-030-89385-9_8

11. Hoc JM. Towards a cognitive approach to human–machine cooperation in dynamic situations. *Int J Human-Computer Stud* (2001) 54:509–40. doi:10.1006/ijhc.2000.0454

12. Decker KS. *Environment centered analysis and design of coordination mechanisms*. Amherst, MA: Doctoral dissertation, University of Massachusetts Amherst (1995).

13. Jennings NR. Coordination techniques for distributed artificial intelligence. In: GM O'Hare NR Jennings, editors. *Foundations of distributed artificial intelligence*. Wiley (1996). p. 187.

14. Malone TW, Crowston K. The interdisciplinary study of coordination. *ACM Comput Surv* (1994) 26:87–119. doi:10.1145/174666.174668

15. Lawless WF. Towards an epistemology of interdependence among the orthogonal roles in human–machine teams. *Found Sci* (2021) 26:129–42. doi:10.1007/s10699-019-09632-5

16. Kaber DB, Endsley MR. The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theor Issues Ergon Sci* (2004) 5:113–53. doi:10.1080/1463922021000054335

17. Kaber DB, Riley JM, Tan KW, Endsley MR. On the design of adaptive automation for complex systems. *Int J Cogn Ergon* (2001) 5:37–57. doi:10.1207/S15327566IJCE0501_3

18. Entin EE, Serfaty D. Adaptive team coordination. *Hum Factors* (1999) 41:312–25. doi:10.1518/001872099779591196

19. Burke CS, Stagl KC, Salas E, Pierce L, Kendall D. Understanding team adaptation: A conceptual analysis and model. *J Appl Psychol* (2006) 91:1189–207. doi:10.1037/0021-9010.91.6.1189

20. Johnson M, Vignati M, Duran D. Understanding human-autonomy teaming through interdependence analysis. In: *Symposium on human autonomy teaming* Southsea, United Kingdom: NATO/STO (2018). p. 20.

21. Fikes RE, Nilsson NJ. Strips: A new approach to the application of theorem proving to problem solving. *Artif Intelligence* (1971) 2:189–208. doi:10.1016/0004-3702(71)90010-5

22. Brooks RA. Intelligence without representation. *Artif Intelligence* (1991) 47:139–59. doi:10.1016/0004-3702(91)90053-M

23. Erol K, Hendler J, Nau DS. HTN planning: Complexity and expressivity. *AAAI* (1994) 94:1123

24. Smith RG, Davis R. Frameworks for cooperation in distributed problem solving. *IEEE Trans Syst Man Cybern* (1981) 11:61–70. doi:10.1109/TSMC.1981.4308579

25. Duarte M, Oliveira SM, Christensen AL. Evolution of hybrid robotic controllers for complex tasks. *J Intell Robot Syst* (2015) 78:463–84. doi:10.1007/s10846-014-0086-x

26. Duarte M, Gomes J, Costa V, Oliveira SM, Christensen AL. Hybrid control for a real swarm robotics system in an intruder detection task. In: G Squillero P Burelli, editors. *Applications of evolutionary computation*, 9598. Cham: Springer International Publishing)Series Title: Lecture Notes in Computer Science. (2016). p. 213–30. doi:10.1007/978-3-319-31153-1_15

27. Durfee E, Montgomery T. Coordination as distributed search in a hierarchical behavior space. *IEEE Trans Syst Man Cybern* (1991) 21:1363–78. doi:10.1109/21.135682

28. Johnson M, Vera A. No AI is an island: The case for teaming intelligence. *AI Mag* (2019) 40:16–28. doi:10.1609/aimag.v40i1.2842

29. Johnson M, Bradshaw JM, Feltovich PJ, Jonker CM, Van Riemsdijk MB, Sierhuis M, Coactive design: Designing support for interdependence in joint activity. *J Human-Robot Interaction* (2014) 3:43. doi:10.5898/jhri.3.1.johnson

30. Johnson M, Shrewsbury B, Bertrand S, Calvert D, Wu T, Duran D, et al. Team IHMC's lessons learned from the DARPA robotics challenge: Finding data in the rubble. *J Field Robot* (2017) 34:241–61. doi:10.1002/rob.21674

31. Iovino M, Scukins E, Styrud J, Ögren P, Smith C. A survey of Behavior Trees in robotics and AI. *Robotics Autonomous Syst* (2022) 154:104096. doi:10.1016/j.robot.2022.104096

32. Lesser V, Decker K, Wagner T, Carver N, Garvey A, Horling B, et al. Evolution of the GPGP/TÆMS domain-independent coordination framework. *Autonomous Agents Multi-Agent Syst* (2004) 9:87–143. doi:10.1023/BAGNT.0000019690.28073.04

33. Endsley MR. Toward a theory of situation awareness in dynamic systems. *Hum Factors* (1995) 37:32–64. doi:10.1518/001872095779049543

34. Thompson JD *Organizations in action: Social science bases of administrative theory*, 10. New York, NY: McGraw-Hill College (1967).

35. Perrow C. *Normal accidents: Living with high risk technologies*. Princeton, NJ: Princeton university press (1999).

36. Steiner ID. Group process and productivity. *Social psychology*. New York: Academic Press (1972).

37. Tambe M. Towards flexible teamwork. *J Artif Intell Res* (1997) 7:83–124. doi:10.1613/jair.433

38. Hughes J. Why functional programming matters. *Comput J* (1989) 32:98–107. doi:10.1093/comjnl/32.2.98

39. Decker KS, Lesser VR. Generalizing the partial global planning algorithm. *Int J Coop Inf Syst* (1992) 01:319–46. doi:10.1142/S0218215792000222

40. Salas E, Sims DE, Burke CS. Is there a "Big five" in teamwork? *Small Group Res* (2005) 36:555–99. doi:10.1177/1046496405277134

41. Myers KL. Cpef: A continuous planning and execution framework. *AI Mag* (1999) 20:63

42. Rossi F, Venable KB, Walsh T. Preferences in constraint satisfaction and optimization. *AI Mag* (2008) 29:58. doi:10.1609/aimag.v29i4.2202

43. Dean TL, Boddy MS. An analysis of time-dependent planning. *AAAI* (1988) 88:49

44. Zilberstein S. Using anytime algorithms in intelligent systems. *AI Mag* (1996) 17:73