Understanding Human-Autonomy Teaming through Interdependence Analysis

Matthew Johnson, Micael Vignati and Daniel Duran
40 S. Alcaniz Street
Pensacola, FL 32502
USA

mjohnson@ihmc.us, mvignati@ihmc.us, dduran@ihmc.us

ABSTRACT

As any craftsman knows, having the right tool for the job makes all the difference. Sadly, adequate formative tools are lacking for the design of human-autonomy teaming. Most existing tools focus on the taskwork, in particular physical activity. More advanced efforts extend this with consideration for cognitive activity. While these traditional task-based approaches are important, they do not sufficiently capture human-machine teaming requirements. The key to understanding human-autonomy teaming, and in fact teaming in general, is understanding interdependence. This paper proposes several design principles to help designers reorient their minds and view the problem space through the lens of interdependence. The paper then discusses a design and analysis tool called the Interdependence Analysis tool. This tool's purpose is to understand how people and automation can effectively team by providing insight into the interdependence relationships used to support one another throughout an activity. This tool helps recognize both the human factors and the technological factors that enhance or inhibit effective teaming. As such, it can be used to derive both algorithmic and interface implementation requirements. The goal of interdependence analysis is improving the guidance provided to designers building human–machine systems in order to achieve more effective teaming. As we build more intelligent machines to tackle more sophisticated challenges, designers need formative tools for creating effective teaming. The Interdependence Analysis tool meets this need, explicitly modelling the machine, the human, the work and the interplay of all three.

1.0 MOTIVATION

As any craftsman knows, having the right tool for the job makes all the difference. Sadly, adequate formative tools are lacking for the design of human-autonomy teaming. The vast majority of formative design tools are technology-centric in which the human is not even considered part of the system (e.g. MATLAB, LabVIEW). Human factors research has highlighted the need for consideration of the human, but this is often accomplished using summative assessments to evaluate existing systems at the end of development (e.g. NASA TLX). User-centered design has pushed for such evaluations to happen earlier in the development process. Apple, for example, has a reputation for early user evaluations translating into successful products. However, this remains the exception rather than the norm for most system development efforts. Moreover, while user evaluations are undoubtedly valuable, designers need to be able to account for such considerations long before there is a system to evaluate. To assist designers in accomplishing this, they need formative tools available in the design process that include the human as part of the system from the beginning.

Another major impediment to designing human-autonomy teaming is that most existing tools focus on the

taskwork, in particular the physical activity. Taskwork is only a small piece of the teaming puzzle. More advanced efforts from cognitive systems engineers extend this with consideration for cognitive activity (e.g. [1]). This is a valuable contribution, but still tends to be taskwork focused. While these traditional task-based approaches are important, they do not sufficiently capture human-machine teaming requirements. Moreover, the language, concepts, and products of those who focus on cognitive aspects are often far removed from those who design and implement working systems and do not translate "into a language that helps a product be built or coded" [2].

The key to understanding human-autonomy teaming, and in fact teaming in general, is understanding interdependence. Teaming, both human-human and human-machine, comes in many forms with varying characteristics and properties. The one concept that is consistent in every case is the importance of interdependence. This truth is invariant across all domains. This paper begins with an explanation of interdependence. To help designers reorient their minds and view the problem space through the lens of interdependence, we propose several design principles. These principles are intended to help designers shift their perspective and reframe the design challenges to focus on the core elements of teaming, specifically the interdependence needed to support it. Lastly, we discuss a formative tool called the Interdependence Analysis tool.

2.0 UNDERSTANDING INTERDEPENDENCE

In order to analyze a human-machine system's interdependence, it is necessary to understand the concept of interdependence. It is often simply equated to dependence, where one entity relies on another because it lacks some capability provided by the other. However, this definition of the concept is too simplistic to capture the nuances observed in interdependence relationships among teams engaged in joint activity. Our definition of interdependence is as follows:

> *"'Interdependence' describes the set of complementary relationships that two or more parties rely on to manage required (hard) or opportunistic (soft) dependencies in joint activity"* [3].

Interdependence is about relationships, not taskwork. It is true that the taskwork influences the potential relationships, however, designing human-machine teaming is about designing the interdependence relationships not the taskwork functions. These relationships are used to manage the interdependence in the joint activity. Relationships can be required, but a significant portion of teaming is about exploiting opportunistic relationships.

To better intuit the concept of interdependence, consider an example of playing the same sheet of music as a solo versus a duet. Although the music is the same, the processes involved are very different [4]. The difference is that the process of a duet requires ways to support the interdependence among the players. Understanding the nature of the interdependencies among team members provides insight into the kinds of coordination or teaming that will be required, such as the management of timing, tempo, and volume. Supporting these relationships through agreed signaling and exploiting these interdependence relationships at run-time is what teaming is all about. Success in a duet requires not only execution of the musical score (i.e. individual taskwork competency), but also the extra work of coordinating with others via interdependence relationships to produce effective performance. Flawless execution of the musical score individually is not enough for the duet to be successful, nor is exceptional individual competence sufficient for effective teaming.

3.0 INTERDEPENDENCE DESIGN PRINCIPLES

While having the right tools is valuable, having the right mindset or perspective on the problem is also essential to effectively leveraging the tools. This turns out to be particularly challenging for human-autonomy teaming largely because of the pervasiveness of today's function allocation-based paradigms [5]. In general, such approaches date back to Sheridan and Verplank's work on supervisory control [6]. Follow up research on dynamic and adaptive function allocation has led to numerous proposals for dynamic adjustment of autonomy. Such approaches have been variously called adjustable autonomy, dynamic task allocation, sliding autonomy, flexible autonomy, and adaptive automation. In each case, the system must decide at runtime which functions to automate and to what level of autonomy [7], thus we refer to them as function allocation-based approaches. A restrictive outcome of such approaches is that automation choices dictate interaction possibilities.

Our approach inverts this paradigm, suggesting that desired teaming interactions should shape the automation design [8]. Thus, our focus is on the interdependence relationships that enable interaction. Focusing on relationships instead of functions is challenging for most people raised under the traditional task decomposition paradigm. Task decomposition is the staple approach to tackling hard problems by breaking them down into smaller ones. Understanding interdependence requires an additional step of considering how those pieces will fit back together again when distributed between people and machines. As Peter Drucker noted long ago, "when it comes to the job itself, however, the problem is not to dissect it into parts or motions but to put together an integrated whole" [9]. While his statement was about business management, it summarizes the true challenge of human-machine teaming nicely.

To help designers reorient their minds and view the problem space through the lens of interdependence, we propose several design principles. The first principle is about the value of teaming. While there is certainly a cost to teaming, and that cost must be controlled [10], there is a value to teaming that must be balanced against the cost. Historically, traditional approaches appear almost surprised by interdependence. For example, an early task decomposition approach to planning noted that "the expansion of each node produces child nodes. Each child node contains a more detailed model of the action it represents... The individual subplan for each node will be correct, but there is as yet no guarantee that the new plan, taken as a whole, will be correct. There may be interactions between the new, detailed steps that render the overall plan invalid" [11]. Similarly, another approach noted "because the relevant constraints have been shared during the planning process, the expectation is that few, if any, conflicts will appear during plan merging. However, because of the complexity of planning dependencies, conflicts can arise" [12]. These references from early planning systems unknowingly highlighted the need to handle interdependence. Their solution was to try to supplement their task decomposition approach with capabilities such as critics used to resolve conflicts and backtracking techniques to handle cases that could not be resolved. Today's function allocation approaches are no different from these early planning approaches because they focus on what to automate and who to assign it to [7]. More recent work still notes that "'conventional wisdom' is often an over-simplification, and will be modified and sometimes reversed by a host of contextual factors" [13]. The main problem with traditional approaches was noted by Stefik long ago who observed "Subproblems interact. This observation is central to problem solving", but concluded "A key step in design is to minimize the interactions between separate subsystems" [14]. While traditional approaches view interdependence as an unexpected or unfortunate problem to be resolved, avoided or minimized, we view it as the core design element. It is something to be leveraged opportunistically. The main reason is because the value of teaming comes not from avoiding, ignoring, limiting or minimizing teaming, but from exploiting it. Thus, our first principle:

Principle 1: The value of teaming comes from exploiting it not avoiding it.

In order to exploit teaming, it is important to understand teaming. Historically, engineers have designed work to be done by an individual. Robotic algorithms generate actions for an individual robot. Intelligent planners, in general, generate a sequence of actions for an individual agent or distributed to several agents to be executed individually. In contrast, we propose that all behaviors should be designed from the beginning to be joint work [8]. This means that as any behavior or algorithm is created, designers should consider the potential teaming associated with the activity. If this is done, the single agent case is just a degenerate case and is achieved for free. However, the converse is not true. Thus, our second principle is:

> *Principle 2: All work should be designed as joint activity (coactive), with independent work being the degenerate case.*

The implication of this principle is that designers should be designing collaborative algorithms for distributed and decentralized systems. This is important because human-machine systems are inherently distributed and decentralized.

In order to achieve the second principle, we need to be able to describe joint activity. An essential theoretical understanding that is absent in existing approaches are principles needed to determine how work can be performed jointly. Currently, engineers decide where to divide work into sub-tasks or actions. This is done with limited understanding of the consequences or impact of these choices, resulting in well-known issues like the substitution myth [15]. It is not just the automation, but the design of the automation that affects performance. Because of this, there is a need for understanding the work itself and the interdependence created by decomposition and distribution. Some examples of this can be found in research on group processes and productivity which provides a taxonomy of group tasks [16]. It includes knowing if a task is divisible or not, whether the task requires maximizing or optimizing, whether the task is additive, disjunctive or conjunctive just to name a few. This leads us to our third principle:

> *Principle 3: Any work has inherent potential and limitations on joint-ness.*

This principle is about uncovering and identifying the potential interdependencies in joint work. One particularly human capability is our amazing ability to team. People are thrown into teams without training and sometimes even without domain knowledge and yet are capable of at least rudimentary teaming. Our life experiences have enabled us to understand what aspects of work require synchronization, when acknowledgement is useful, what information is relevant to share and how to request assistance. While people do make mistakes and some people are more naturally skilled than others, this generalizability across domains - regularly exhibited by people - provides the intuition that there are common patterns in joint work which support our third principle.

In order to model joint work, we must first address what counts as work. Traditional engineering and planning target physical actions that affect the world, but can ignore cognitive aspects such as sensing, perception, memory, reasoning and understanding. This oversight is demonstrated by the development of an entirely separate field known as Cognitive Engineering [17] to address issues ignored by traditional engineering. While teamwork does involve coordinating physical actions, a significant role of teamwork is coordinating cognitive activities as well [18]. Designs that do not include these aspects will not produce effective team players [10]. Thus, our fourth principle:

> *Principle 4: Teaming occurs not just on the physical level, but also on the cognitive level.*

As we get into more specific discussions on modelling and representation, we can draw upon good design

principles that cross domain boundaries. Given the complexity of teaming, our fifth principle, that is just good design practice in general and will be applied in several ways is:

Principle 5: Separate the "what" from the "how".

As an example, consider the goal of making sure a teammate does not fall into a hole while working. This could be accomplished by specifying a route for the teammate that avoids the hole, informing the teammate about the hole and letting them avoid it, positioning oneself or an object by the hole to force the teammate to avoid both the obstacle and the hole. All of these achieve the same purpose (the what) but through different means (the how).

There are many factors that go into effective teaming. These include the work itself, the team members involved in the work and the environmental factors in which the work is performed. The challenge is that all of these factors interact. In accordance with principle five, we propose our sixth principle:

Principle 6: Joint work can and should be modelled in an agent agnostic manner.

By this we mean that the description of the work should not include considerations for specific team compositions or team member capabilities. This does not mean team composition will not be accounted for, just that the initial work description is not the appropriate place to do this. This is indeed a foreign concept in robotics, as most developers strive to customize their solutions to the specific targeted hardware, though recent efforts like ROS (Robot Operating System [19]) has shown the value of abstraction in the robotics domain. This concept is less foreign in domains such as cross platform application development. In such domains, layers of abstraction are used to enable generalization. Similarly, our principle is trying to emphasize that you may not know the characteristics of the teammates ahead of time, so designing your joint work on top of a reasonable abstraction of the work itself will allow for broader applicability and reuse.

If work should be modelled as joint work, how is this different than regular work? We propose it is the inclusion of interdependence in the model that captures the potential joint-ness. Malone and Crowston define coordination as "managing dependencies between activities" [20]. Studying human teams and team effectiveness, researchers have identified team member interdependence as a critical feature defining the essence of a team [21]. From human-machine research, Feltovich et. al. propose interdependence is the essence of joint activity [22]. Interdependence focuses on how the decomposed and potentially distributed work remains interdependent. This provides exactly what is needed to understand joint work, to understand the implications of different decomposition choices, and to specify requirements based on teaming [3]. Thus, our seventh principle:

Principle 7: Interdependence provides the basis for understanding potential joint-ness.

Some interdependencies are obvious. For example, resource constraints on a shared resource or sequencing constraints on dependent tasks. These are hard requirements that must be coordinated through teaming. As hard requirements, they are unavoidable and therefore obvious. However, a significant amount of normal human teaming involves opportunistic (i.e., soft) interdependence relationships. Soft interdependence does not stem from a hard constraint or a lack of capability. It arises from recognizing opportunities to be more effective, more efficient, and/or more robust by working jointly. Soft interdependence is less obvious because it is optional and opportunistic rather than strictly required. It includes a wide range of helpful things that a participant may do to enhance team performance. Examples include progress appraisals ("I'm running late"), warnings ("Watch your step"), helpful adjuncts ("Do you want me to pick up your prescription when I go by the drug store?"), and observations about relevant unexpected events ("It has started to rain"). Many aspects of teamwork are best

described as soft interdependencies. Our observations to date suggest that good teams can often be distinguished from great ones by how well they manage soft interdependencies. Thus, our eighth principle:

> *Principle 8: Teaming involves both required (hard) and opportunistic (soft) interdependencies.*

So how does a designer identify interdependencies, particularly the less obvious soft interdependencies? Coactive Design proposed three essential interdependence relations; observability, predictability and directability [3]. Observability means making pertinent aspects of one's status, as well as one's knowledge of the team, task, and environment observable to others. Predictability means one's actions should be predictable enough that others can reasonably rely on them when considering their own actions. Directability means one's ability to influence the behavior of others and complementarily be influenced by others. There are certainly additional types of interdependence relationships, such as explainability and trust, but we view these three as foundational to the others. This leads to principle nine:

> *Principle 9: Observability, Predictability and Directability are compulsory interdependencies in teamwork.*

One characteristic of teaming that provides compelling value is the ability for teams to be robust to individual failures. This does not happen by accident or without any effort. Teams monitor and assess the state of each other in order to achieve this advantage. Underlying this skill is the understanding that failure is always an option. Human failings and limitations are well known and often the motivation for more automation. Yet automation has its own failings and limitations. This is not a problem as long as the team is attentive to the potential for failure. More directly, that the human-machine team is designed to address it. Thus, principle ten:

> *Principle 10: Failure is always an option and teaming should be designed to help the team be robust to failure of both people and machines.*

This means that designers should not only be considering if performing some task is possible, but should be considering the risks and frailties of a task with respect to all team members. This provides insight into the importance and potential necessity of different teaming options in order to prevent any single team member from being a critical point of failure. Supporting this principle involves designing appropriate observability and predictability relationships to support monitoring, backup behaviors and other teaming competencies.

Our last principle is another based on principle five — separating the what from the how. It addresses an aspect of teaming that is important, namely teaming strategy. Teaming strategy is the means by which a team chooses to exploit available interdependence options. In traditional systems that tackle multi-agent teaming, the teaming strategy is often tightly coupled to the overall system implementation, making it difficult if not impossible to change teaming strategies (e.g. [23]). It also confounds the scientific analysis of teamwork by not separating the teaming capability within the work from the teaming strategy employed by the team members. Learning to understand the teaming capability of the work separate from the teaming strategy of the team members helps designers comprehend the influence of each other on overall system performance. Thus principle eleven is:

> *Principle 11: The joint-ness of work and the teaming strategy can and should be considered separately, though they need to be designed to work together.*

Teaming strategy is not addressed by the IA tool, but it is included in the discussion because strategy is an

important part of teaming. The IA tool provides insight into the tools and options available to teaming strategies.

The purpose of these principles is to help reshape a designer's mindset in order to effectively employ the Interdependence Analysis tool. Table 1 summarizes the interdependence design principles.

Table 1. Summary of Interdependence Design Principles

Interdependence Design Principles	
1	The value of teaming comes from exploiting it not avoiding it.
2	All work should be designed as joint activity (coactive), with independent work being the degenerate case.
3	Any work has inherent potential and limitations on joint-ness.
4	Teaming occurs not just on the physical level, but also on the cognitive level.
5	Separate the "what" from the "how."
6	Joint work can and should be modelled in an agent agnostic manner.
7	Interdependence provides the basis for understanding potential joint-ness.
8	Teaming involves both required (hard) and opportunistic (soft) interdependencies.
9	Observability, Predictability and Directability are compulsory interdependencies in teamwork.
10	Failure is always an option and teaming should be designed to help the team be robust to failure of both people and machines.
11	The joint-ness of work and the teaming strategy can and should be considered separately, though they need to be designed to work together.

4.0 INTERDEPENDENCE ANALYSIS TOOL

The purpose of Interdependence Analysis (IA) is understanding how people and automation can effectively team by identifying and providing insight into the potential interdependence relationships used to support one another throughout an activity. The Interdependence Analysis tool was developed to assist with IA. The tool can be used to analyze existing systems, but one of the tool's strengths is that it can also be used formatively to guide the initial design process. The need for formative tools is consistent with Kirlik et al., who emphasize "the importance of understanding why cognitive demands are present, prior to determining a strategy for aiding the operator in meeting these demands" [24](p. 950). Understanding and designing for interdependence can provide this type of guidance. IA provides insight into when situation awareness is adequately supported and when it is not. It can inform the designer of what is and is not needed, what is critical, and what is optional. Most importantly, it can indicate how changes in capabilities affect relationships. As systems develop and improve, understanding the impact of how these changes impact human-autonomy teaming is critical to ensuring acceptance and utility of new technology.

The IA tool is in the form of a table, as shown in Figure 1, with three main sections: joint activity modelling, assessment of potential interdependence, and analysis of potential workflows.

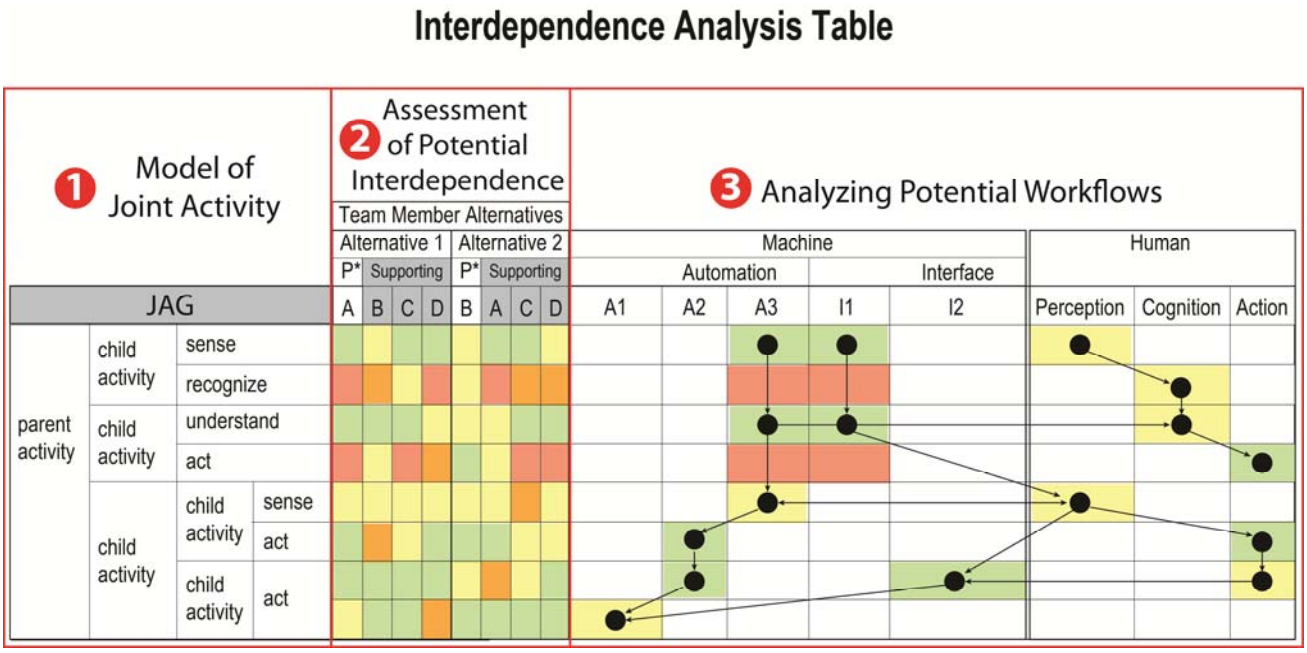

Figure 1. Generic Interdependence Analysis Table with three main section labelled. Section 1 helps designers model the joint activity, section 2 helps them identify potential interdependencies in the activity, and section 3 helps analyze the potential workflows to better understand the flexibility and risk in the human-machine system.

4.1 Modelling of Joint Activity

The first section of the IA tool focuses on modelling the joint activity. In accordance with principle 2, this modelling should model all work as joint activity. In order to accomplish this, it is important to understand what is unique and important about joint activity and how these aspects can be modelled.

4.1.1 What is Joint Activity?

So, what does it mean to model some task or function as joint activity? Our view of joint activity comes from work on joint activity theory [10], [22], a generalization of Herbert Clark's work in linguistics [4]. Joint activity has important characteristics with respect to structure, process and the potential for interaction [25].

The first distinction is in the overall structure of joint activity. Joint activity is embedded sets of actions. When viewed as a task, traditional engineering practice suggest making the given work an atomic isolated module that performs the specified function when called. The internals of the standalone module are typically hidden through encapsulation. While this is good programming practice for software development and integration, it turns out to be problematic for teaming. This is because tasks or functions are actually part of embedded sets of actions, in other words an activity. The function may have several embedded sets of actions within it, or it may itself be embedded in a set of actions or both. Though team members may be working on functions that can be represented individually, it is important, from a teaming perspective, to understand the overall team activity

context provided by the activity structure.

The second distinction is that joint activity is a process, one that extends in space and time. When viewed as a task or function, this is often overlooked, as if no time transpires and the world stands still. This has many implications from a teaming perspective. The first is that events of the past, current status within an activity and intentions for the future are all potentially relevant for effective teaming. Additionally, having a process implies that there is additional work necessary to compose the hierarchical structure. This additional work, often referred to as coordination, is also important for effective teaming.

The last distinction important to this discussion is that joint activity has the potential for interaction. While considering work as embedded sets of actions that are part of a process shifts thinking from individual tasks to activities, it is the potential for interaction that enables the activity to be considered joint. If there is not substantive interaction, then the work is parallel — not joint [25]. This need to support interaction drives what it means to model joint work. It means to decompose joint activity into a sets of embedded actions and to instrument those actions to properly support interactions needed for distributed team members to recompose the final solution. This is where designing for interdependence opposes traditional information-hiding practices. Functional encapsulation and opaque automation are often at odds with effective teaming. Understanding the nature of joint activity also plays a role in a more nuanced understanding of interaction. Since activity involves nested sets of activities, interaction can happen across a variety of levels of abstraction. Since it is a process, interaction can involve the past, the future, and ongoing progress during an activity. Defining a system as a human-machine team is defining it as joint activity, but defining it alone is not enough to make it a reality; the system must be designed and built.

4.1.2 How to Model Joint Activity – The Joint Activity Graph

While there are conceivably numerous ways to model joint activity, we propose one solution called the Joint Activity Graph (JAG). The purpose of the JAG is to capture the key elements of joint activity: structure, process and potential interactions.

The data structure underlying a JAG, is a simple graph. It is a tree of unit height — a single activity with zero or more children. More complex activity can be constructed by assembling multiple JAGs into a larger structure that is itself a JAG. Any high level JAG can be understood by recursively expanding all the children to provide a tree-like view on the overall activity. This recursive data structure has intrinsic benefits for composition and reuse. The JAG provides a description of all the embedded activities that are necessary to the proper execution of the activity it describes. As a hierarchical structure, it is similar to hierarchical task networks (HTNs) [26], [27]. As with HTNs, JAGs can be directly executable or conceptual like a goal, sometimes referred to as primitive and non-primitive tasks respectively. However, JAGs do not require explicit declaration as such. Whereas planners are built on top of existing systems with known capabilities, human-machine teaming is constantly evolving and the potential variation in team composition means rigid assumptions on the type of actions that are executable by one agent will likely be invalid for some other agent. Thus, JAGs were designed to postpone the decision on what is directly executable allowing teams flexibility to resolve this at runtime.

If structure alone was all that was needed to understand teaming, then JAGs would be unnecessary as a hierarchical tree would suffice. However, the assembly of a JAG into a more complex activity also requires a process description. JAGs contain a process model whose purpose is modelling how to assemble its immediate descendants into the parent. As with structure, flexibility is an important aspect of the teaming process. The goal of modelling the process with a JAG is not to dictate the “right way” to complete an activity, but to model the permissible ways to do so. The JAG aims to depict the roadmap of options not just one viable road. In other

words, it is not meant to define the one solution, but a solution space. The re-composition process can be individually defined for each unit JAG and is fully customizable. While these can be designed from scratch to fit specific needs, we natively support a set of common useful alternatives. These alternatives are combinations of two broad concepts: sequencing and logic. Sequencing provides for sequential or parallel execution. Logic provides the operators "And" and "Or" for specifying conjunctive and disjunctive tasks. Together these provide four useful combinations:

1. Sequential-And: All activities need to be completed successfully in sequence.
2. Sequential-Or: Activities executed in sequence until one is completed successfully.
3. Parallel-And: All activities can be executed in parallel and need to be completed successfully.
4. Parallel-Or: All activities can be executed in parallel but are racing each other until one completes successfully.

These four processes provide sufficient mechanisms to model many different activities. However, because it is unlikely that we can cover all types of processes imaginable, we left the process definition open for future extensions.

Structure and process dictate how an activity can be decomposed and distributed and how it can be recomposed to achieve the goals of the activity, however neither defines the potential for interactions that determine the joint-ness. This is accomplished by identifying the interdependence relationships that underlie and define the needed interactions which in turn enable teamwork. Some interdependencies come from the structure. For example, if a JAG's children are distributed across team members, to know that the parent JAG is successful may require information sharing. Interdependence can also stem from the process. For example, if sequential activities are distributed across team members then they will need to coordinate their sequencing. However, many of the important teamwork dimensions such as adaptability, situation awareness, performance monitoring and feedback, and decision making [28] involve interdependencies beyond those associated with structure or process. The purpose of the IA tool is to help designers identify potential interdependencies. Specifically, observability, predictability and directability (OPD) requirements explain what kind of interactions are potentially valuable to achieve effective teaming (principle 9). More importantly they depict precisely how a given activity must be instrumented to expose and ingest appropriate information.

4.2 Assessing Potential Interdependence

After modelling the joint activity, the next step is to assess the potential interdependence. This involves enumerating the viable team role alternatives, assessing the capacity to perform the work, assessing the capacity to support another team member as they perform the work, identifying potential interdependencies, and then determining the requirements to support the interdependence relationships of interest.

4.2.1 Enumerating Viable Team Role Alternatives

While the joint activity is modelled in an agent-agnostic manner, the assessment of interdependence is in general not. This is because interdependence is about relationships. Therefore, any team alternative needs at a minimum two team members, though larger teams are permitted. Having specific individual team members is not required, though it does permit greater specificity. The team alternatives section of the tool captures the fact that team composition has an impact on teaming. It also permits analysis and comparison of how changes to team composition impact teaming.

To aid interdependence analysis, the IA tool makes a distinction between a performer and supporting team

members. The performer is the individual primarily doing the given aspect of the work. The supporting team members are then viewed from the perspective of assisting the performer. It is the supporting team member columns that are key to identifying interdependencies. For a given alternative, only one entity is assigned as the performer, labelled P* in Figure 1. This is not to say others cannot do the work, but is simply a mechanism to aid the designer in considering a certain perspective.

In general, IA tables will have a minimum of two alternatives. The first with a given performer and supporter and the second with the roles reversed. This allows the designer to consider all of the joint work from both perspectives. This permutation is another key element to identifying all potential interdependencies, which in turn guides the design of effective teaming. In addition to considering any party performing any part of the work, this also forces consideration for any party assisting in any part of the work. If a team consisted of two identical team members, then having two team alternatives would be redundant since they would be identical. The main reason for a minimum of two alternatives is that people and machines are inherently different. This asymmetry needs to be understood and accounted for when designing human-autonomy teaming.

The columns in each alternative can represent specific individuals (existing or planned). If the team has more than two members then additional columns can be used, as represented by columns A, B, C and D in Figure 1. For larger teams this can become unwieldy, but categories and roles can be used to keep it manageable. For example, consider a single operator managing four unmanned aerial vehicles (UAVs) and two unmanned ground vehicles (UGVs). Assuming the vehicles are of the same type then categories can be used. The team alternative would be three columns: one human operator, one for UAVs, and one for UGVs. Multiple people are also permitted and can be simplified with roles. Consider extending the previous example to be two such units being managed by a commander. These fifteen entities can be captured by four columns: commander, operator, UAV, UGV. There is a limit to the feasibility of extending the table to large teams, but such teams are more like organizations than teams.

In general, a permutation of team members in which each is assigned the performer is how team alternatives are created. While permutations across a team seems a bit daunting and suggest visions of exponential explosion, in practice we have found that humans generally team in small numbers and larger teams use categories and roles to manage scale. Some may question whether such consideration for the team composition is necessary for effective design. Consider a simple example of providing some driver instructions on how to get to your house with some road construction blocking the obvious route. If the driver you are teaming with is someone familiar with your neighborhood, versus someone unfamiliar you probably need different coordination mechanisms. If the driver is a teenager learning to drive, or a young adult, or a senior citizen you might also imagine different teaming mechanisms. There is no “one size fits all” when designing teaming, especially when considering team composition. This is not a limitation of our approach, but merely a reflection of reality. There is a tradeoff to be made with respect to how much detail to include, but having no representation of team composition is unlikely to be effective.

4.2.2 Assessing Capacity to Perform and Capacity to Support

After the team alternatives are determined, the next step is an assessment of each column in a team alternative to each row in the joint activity model. For formative tasks, the assessment is necessarily subjective, but when assessing existing systems, it is possible to use empirical data to inform the assessment [29]. The assessment process uses a color coding scheme, as shown in Figure 2. The color scheme is dependent on the type of column being assessed.

Color Key for Team Member Role Alternative Capability Assessment	
Performer	**Supporting Team Members**
I can do it all	My assistance could improve efficiency
I can do it all but my reliability is < 100%	My assistance could improve reliability
I can contribute but need assistance	My assistance is required
I cannot do it	I cannot provide assistance
Not applicable / Not significant	Not applicable / Not significant

Figure 2. Color key for team member role alternative capability assessment.

Under the "performer" columns, the colors are used to assess the individual's capacity to perform the activity specified by the row. It complies with principle 10 — considering the potential for failure. The color green in the "performer" column indicates that the performer can do the task. For example, a robot may have the capacity to navigate around an office without any assistance. Yellow indicates less than perfect reliability. For example, a robot may not be able to reliably recognize a coffee mug all the time. Orange indicates some capacity, but not enough for the task. For example, a robot may have a 50 pound lifting capacity, but would need assistance lifting anything over 50 pounds. Another use of orange is to indicate hard-coded assumptions that limit the performance to very specific contexts. For example, a robot may be able to pick up a coffee mug from a table, but it may not be able to do so from the floor or cluttered cabinet. The color red indicates no capacity, for example, a robot may have no means to open a door.

Under the "supporting team member" columns, the colors are an assessment of that team member's potential to support the performer for the activity specified by the row. The color red indicates no potential for interdependence, thus independent operation is the only viable option for the task. Orange indicates a hard constraint, such as providing supplemental lifting capacity when objects are too heavy. Another example of orange is when a machine needs human authorization to perform the activity. Yellow is used to represent improvements to reliability. For example, a human could provide recognition assistance to a robot and increase the reliability in identifying coffee mugs. Green is used to indicate assistance that may improve efficiency. For example, a robot may be able to determine the shortest route much faster than a human or could assist in cleaning up a room.

One last note on color coding. Some relationships are not significant and so either the performer or the supporting team members' assessment may not be applicable. In these cases, use of grey is suggested to minimize the attention drawn by such cases [30].

4.2.3 Identifying Potential Interdependence Issues

Once the assessment process is finished, the color pattern can be analyzed. The color pattern characterizes the nature of the interdependence within a team for the given joint activity.

Colors other than green in the "performer" column indicate some limitation of the performer, such as potential brittleness due to reliability (yellow) or hard constraints due to lack of capacity (orange). The hard constraints in the performer column indicate a need to team to accomplish the work and are usually fairly obvious. The more interesting situation is the potential brittleness which is often less obvious. Teaming is not required in this

circumstance, but doing so can make the team more resilient.

Colors other than red in the "supporting team member" columns indicate potential required (orange) interdependence relationships between team members. Again, it is the opportunistic cases (yellow and green) that tend to be the most interesting, less obvious and contribute to resilience.

4.2.4 Determining System Requirements

With the assessment complete, the IA tool can now help designers extract clear-cut design requirements needed to enable and support specific interdependence relationships. For each relationship of interest, the design considers the compulsory interdependencies of observability, predictability and directability (OPD) in accordance with principle 9. In other words, identifying who needs to observe what from whom, who needs to be able to predict what, and how members need to be able to direct each other for a given aspect of the work. As an example, we have created a small IA table based on Fong's Collaborative Control work [18], as shown in Figure 3. In Fong's example there was one teaming alternative: the human assisting the robot. The robot was capable of performing obstacle avoidance; however, it was less than 100 percent reliable (yellow) in interpreting if an obstacle is passable. The human was capable of providing assistance, thus increasing the reliability (yellow) of the robot in this task. The yellow coloring of the human column indicates soft interdependence. Requirements can be derived from analyzing the IA table in Figure 3. The robot must be predictable in notifying the human when unsure about an obstacle because the human is not constantly watching (red). The human's ability to interpret depends on being able to sense the obstacle, so there is an observability requirement. Once the human has interpreted if the obstacle is passable, this information must have a way to alter the robot's behavior, so there is a directability requirement. These requirements define what is needed by both the algorithm and the human interface to support this interaction. Note that the IA tool's purpose is to identify what the requirements are, not how to meet them (principle 5). That is an implementation choice. These particular OPD requirements are based on the desire to support a particular interdependence relationship: the human assisting in interpretation of whether an obstacle is passable. This example demonstrates how OPD requirements derive from the role alternatives the designer chooses to support, their associated interdependence relationships, and the required capacities. This example does not include the reciprocal teaming alternative with the human assisted by the robot because this was outside the scope of the original work.

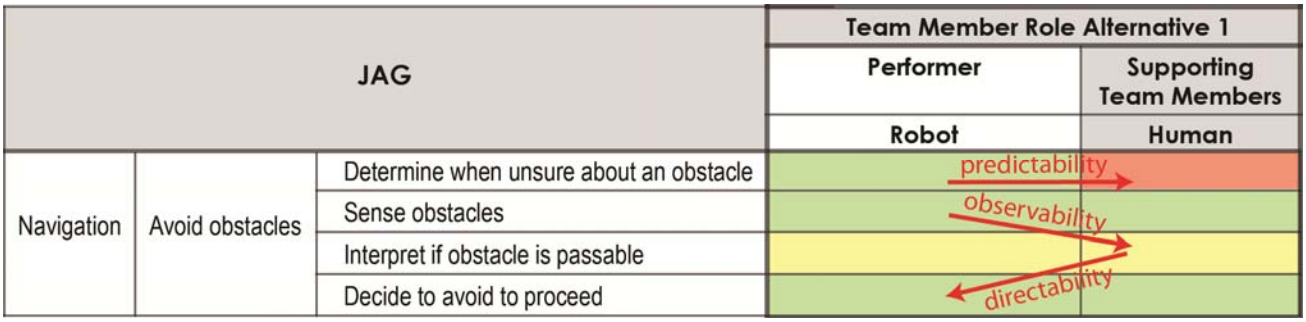

Figure 3. Interdependence analysis example from Fong's (2001) Collaborative Control work, showing observability, predictability and directability requirements based on choosing to allow the human to provide interpretation assistance to the robot during navigation.

4.3 Analyzing Potential Workflows

The original IA tool was introduced as a way to help designers understand and design for the interdependence in human-machine teams [3]. The assessment of potential interdependence provided this, but demanded a fair bit of imagination to envision the implementation alternatives. As the tool was applied in the development process

within a team of engineers, it became clear that it would be valuable to provide an improved visualization of the teaming alternatives to aid the team in developing a unified understanding [29]. Specifically, we needed a way to connect the theoretically understanding of interdependence to the physical instantiation of what had been developed thus far (i.e., the implementation) and what was planned to be developed.

4.3.1 Establishing Workflows

To achieve this, we associated leaf JAGs to particular team member's capability of supporting them. We also expanded the team members to allow more detailed mapping to specific algorithms, interface elements, or human abilities. Across the top of section 3 in Figure 1, there are column headings for each algorithm, interface element or human capability used to accomplish the task. Below each heading is a black dot to indicate where in the activity that particular component has a role. We then connect the dots with arrows to indicate potential workflows to accomplish the goal. The resulting graph structure is a visual description of all existing and potential workflows. Therefore, it is a depiction of the flexibility in the system [31]. The graph can be thought of as the adjustment options in adjustable autonomy or the initiative options in mixed-initiative interaction [32]. However, what they really are is an enumeration of the possible ways a team can complete the joint activity. This is how the IA tool provides not a solution, but a solution space.

Additionally, the graph makes it clear that discrete function allocation is not what is happening because the human is informed by automation and display elements, and automation can be assisted by the human as indicated by the numerous horizontal and diagonal lines shown in section 3 in Figure 1. Most importantly, it shows where teaming is supported and where it is not. This allows grounded debate on the value of additional teaming support versus the cost of implementation.

4.3.2 Assessing Workflows

The color coding in the workflow section of the IA tool, section 3 of Figure 1, is indicative of risk for a given pathway. The color for a given cell is determined by the performer column of the assessment section. Moving vertically the risk is compounded across different functions. However, horizontal pathways indicate potential mitigations through teaming.

The potential workflows section of the IA tool is valuable for several reasons. First it ensures the joint activity model grounds out in actionable terms and does not fall prey to wishful mnemonics [33] or suitcase words [34]. Delineating the roadmap of possibilities also helps to guide a design team in their development. They can easily see what pathways are supported and which are not, but more importantly they can be guided to potentially opportunistic pathways they may not have found otherwise. The IA tool structure is also well suited to collection of empirical data to help inform design choices throughout the design iteration process (e.g. [29]). Information about reliability, performance time and frequency of use associated with specific pathways can help shape design decisions and provide credence to implementation directions.

5.0 INTERDEPENDENCE ANALYSIS EXAMPLE

As an example of how to use the IA tool we will use it to analyze manned aircraft collision avoidance. Figure 4 depicts an abbreviated IA of this challenge. The JAG, shown in the left side of Figure 4, depicts the agent agnostic joint activity (principle 3, 5 and 6). The work includes the cognitive aspects (principle 4) to sense and interpret traffic, decide when there is a conflict and what to do about it, as well as the physical task to take the necessary actions. We assume the work is jointly shared (principle 2) by some automation working together with

a human pilot. Our hypothetical system includes automation, such as a Traffic Collision Avoidance System (TCAS) and a Detect and Avoid System (DAA), shown as headings in the workflow section of Figure 4. The figure depicts two TCAS options for discussion purposes. In teaming alternative one (TA1), assessing the automation's ability to perform, we see that TCAS can reliably sense and interpret traffic, but only cooperative traffic. TCAS has no ability to detect non-cooperative traffic (i.e. not transponder equipped). While TCAS-I only warns about traffic, TCAS-II can decide what to do and will provide vocalized instructions, but this is based on the limited sensing capability of TCAS-I. While DAA systems are not fully operationalized yet, here we assume we are designing a system to have such capability. Our envisioned DAA system leverages TCAS decisions to determine control inputs. We have assumed perfect execution (green) but left open the possibility that the automated system might misdiagnose on rare occasion (yellow) accounting for potential failure (principle 10). By considering that the human pilot has capabilities that could assist the automation (yellow) such as recognizing uncooperative traffic and making the avoidance decision we have identified potential soft interdependencies (principle 7 and 8). TA2 shows the human is capable of everything except directly executing the controls with errors most likely occurring in sensing due to attention or decision making (slow reaction time). Automation can assist the pilot in sensing cooperative traffic and making the avoidance decision (yellow), and generating control commands (green).

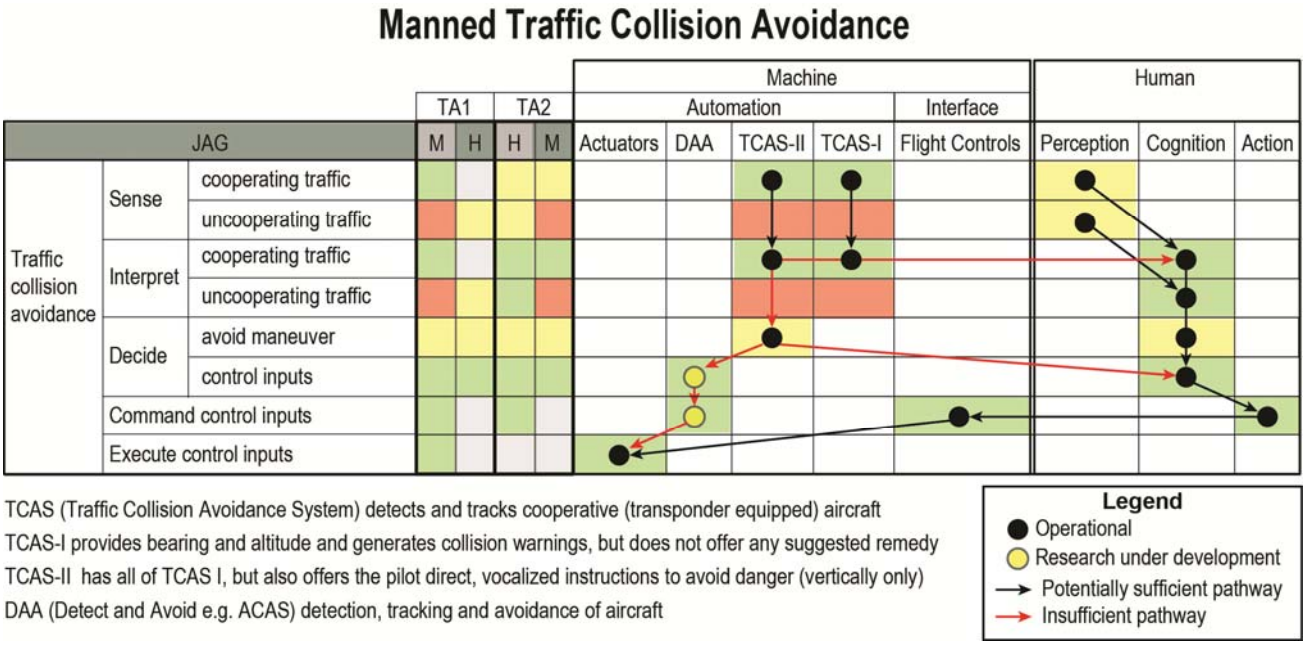

Figure 4. Interdependence Analysis of Manned Traffic Collision Avoidance.

The potential workflows are depicted in the right half of Figure 4. One can see the potential automated solution with TCAS identifying traffic and DAA commanding avoidance maneuvers. Though the final execution of commands is highly reliable by today's automation (green), it is dependent on sensing that lacks some context (red) and decision making that may not be perfect (yellow). This provides designers with an overall risk assessment for the automated solution path. The manual solution is also possible with the pilot detecting traffic, deciding what to do and taking action. There is risk in this solution, but it is different. Pilots have limited attention and may miss traffic and they may misjudge a decision. These are just two solutions, neither of which exploits teaming. By considering the supporting team member columns in each teaming alternative, designers

are guided toward teamwork options throughout the activity.

TA2 considers how the automation can help the pilot. Each option can be represented by horizontal lines that cross any boundary between teammates. Each relationship will have its own associated OPD requirements (principle 9). For example, TCAS can sense and interpret cooperating traffic for the pilot. To support this teaming pattern, the system must make potential traffic conflicts it detects observable to the pilot, in this case through a warning display. This activity can be done in parallel and thus both the pilot and the automation can contribute. The workflow includes both sense and interpret pathways because the automation is not replacing, but supplementing the pilot's ability, as most flight manuals will warn. It also means the risks of each individual pathway, missing context for automation and lack of attention for pilots, can potentially be mitigated by the other. TCAS-II can provide a suggested course of action. This requires a mechanism to direct pilot action, in this case voice instructions. Note that although the DAA has the potential to determine control inputs, it currently does not support providing such assistance directly to the pilot, only through the fully automated solution.

TA1 considers how the human can help the automation. Although the assessment column for the human in TA1 indicates the potential to help recognize uncooperative traffic or decide on a course of action, no support is provided, as shown in the workflow. If the pilot performs these activities, they are done outside of any teaming and are not sharable with the automation in the current design.

This example demonstrates the process of interdependence analysis and how it helps designers understand human-machine teaming. We connected the design principles to the analysis story to show how they relate to teaming and how the IA table guides the designer through these considerations. It should be clear that the potential for teaming and the requirements for supporting it will change as design changes are considered, such as different types of automation (e.g. TCAS-I vs. TCAS-II) or having the pilot be remote. This is how the IA tool aids designers in diagnosing the teaming implications of design choices.

5.1 Summary of IA Tool Capabilities

The IA tool is unique in its capability to aid designers in assessing interdependence in human-machine teams. Currently, it is the only design tool that specifically shows interdependence. It is also novel in that it does not try to determine which team member, the human or the machine, should be allocated to what aspects of work. Instead it focuses on capitalizing on the opportunities for synergy in the team. It enables designers to more effectively find teaming opportunities and helps them understand the requirements needed to exploit those opportunities (principle 1).

The IA tool is also novel in how it includes both people and machines in the design of a system. Many approaches do not include people as part of the system, while others assess human performance without the context of the automation and interfaces. The novel approach to modelling all work as joint work in an agent agnostic way allows for complete flexibility of teaming. The support for consideration of teaming alternatives helps designers to see the problem from multiple perspectives. Providing workflow visualization that includes people, algorithms and interfaces grounds the theory into actionable practice.

The IA tool is also unique in how it captures soft interdependencies. These types of capabilities are often overlooked, but their importance to teaming should not be. Support for such opportunistic teaming options can lead to improved flexibility and better overall system resilience [29]. All too often the purpose of human-autonomy teaming is viewed as a constraint due to automation not being able to do something. IA is about capitalizing on opportunities for synergy, not just filling required slots. It is about mutual enhancement, not

replacement or substitution.

The IA tool is also unique in providing not just a single solution, but a roadmap of design alternatives. This type of view is critical for development of advanced technologies which involve highly iterative development. Supporting multiple pathways on the roadmap provides flexibility and contributes to system resilience.

A critique of the IA process is that it is potentially time consuming to develop joint activity models and consider the different teaming alternatives. This is a valid concern, but building systems that fail or struggle to meet people's needs is also costly. While interdependence analysis will definitely take more time than simple task decomposition, it is unlikely to take more time than the first iterative development of any complex system. Our experience is that its value throughout the life of the project greatly exceeds the initial time investment.

Another concern is that IA requires a high level of expertise. In some sense this is true. The main issue is shifting away from the traditional paradigm. Shifting from a function allocation mind set is not easy, just as shifting to object oriented programming or functional programming is not easy for someone trained in procedural programming. It also requires two skill sets not often found in combination: technology expertise and human expertise. However, ignoring either technology or the human because dealing with both is difficult is not a viable answer. Future educational tracks that emphasize both could alleviate this problem. We have proposed several guiding principles to help designers reorient their perspective, but as in any field, there is no substitute for practice and experience.

6.0 CONCLUSION

The purpose of Interdependence Analysis (IA) is understanding how people and automation can effectively team by identifying and providing insight into the potential interdependence relationships used to support one another throughout an activity. These interdependence relationships enumerate the potential human-machine interactions that comprise teaming. In this way, it provides a roadmap of the opportunities afforded by a given system design. Interdependence relationships help to understand both the human factors and the technological factors that enhance or inhibit effective teaming. IA can be used to derive specific design requirements, both algorithmic and interface, that determine human-machine interactions for each alternative. The analysis also provides a risk assessment associated with each teaming option. Having alternatives provides flexibility, while understanding the risk associated with each option assists operational decision making.

As we build more intelligent machines to tackle more sophisticated challenges, designers need formative tools for creating effective human-autonomy teaming. Interdependence Analysis meets this need by explicitly modelling the machine, the human, the work and the interplay of all three. This enables developers to architect effective teaming by designing support for interdependence. Interdependence Analysis should be viewed as an approach to determining how and what automated capabilities are built such that intelligent systems are imbued with teaming competence.

7.0 REFERENCES

[1] B. Crandall and G. Klein, Working Minds: A Practitioner's Guide to Cognitive Task Analysis. Bradford Books, 2006.

[2] R. R. Hoffman and S. V. Deal, "Influencing versus Informing Design, Part 1: A Gap Analysis," IEEE Intell. Syst., vol. 23, no. 5, pp. 78–81, 2008.

[3] M. Johnson, J. M. Bradshaw, P. J. Feltovich, C. M. Jonker, B. M. van Riemsdijk, and M. Sierhuis, "Coactive Design: Designing Support for Interdependence in Joint Activity," J. Human-Robot Interact., vol. 3, no. 1, pp. 43–69, 2014.

[4] H. H. Clark, Using language. Cambridge [England] ; New York: Cambridge University Press, 1996.

[5] M. Johnson et al., "Beyond cooperative robotics: The central role of interdependence in coactive design," IEEE Intell. Syst., vol. 26, no. 3, 2011.

[6] T. B. Sheridan and W. Verplank, "Human and Computer Control of Undersea Teleoperators." Man-Machine Systems Laboratory, Department of Mechanical Engineering, MIT, Cambridge, MA, 1978.

[7] R. Parasuraman, T. Sheridan, and C. Wickens, "A model for types and levels of human interaction with automation," Syst. Man Cybern. Part A, IEEE Trans., vol. 30, no. 3, pp. 286–297, 2000.

[8] M. Johnson, J. Bradshaw, P. Feltovich, C. Jonker, B. van Riemsdijk, and M. Sierhuis, "The Fundamental Principle of Coactive Design: Interdependence Must Shape Autonomy," in Coordination, Organizations, Institutions, and Norms in Agent Systems VI, vol. 6541, M. De Vos, N. Fornara, J. Pitt, and G. Vouros, Eds. Springer Berlin / Heidelberg, 2011, pp. 172–191.

[9] P. F. Drucker, The Practice of Management. Harper Business, 1954.

[10] G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich, "Ten Challenges for Making Automation a 'Team Player' in Joint Human-Agent Activity," IEEE Intell. Syst., vol. 19, no. 6, pp. 91–95, 2004.

[11] E. D. Sacerdoti, "The Nonlinear Nature of Plans," Proc. 4th Int. Jt. Conf. Artif. Intell., vol. 1, pp. 206–214, 1975.

[12] M. desJardins and M. Wolverton, "Coordinating a Distributed Planning System," AI Mag., vol. 20, no. 4, p. 45, 1999.

[13] C. D. Wickens, H. Li, A. Santamaria, A. Sebok, and N. B. Sarter, "Stages and Levels of Automation: An Integrated Meta-analysis," Proc. Hum. Factors Ergon. Soc. Annu. Meet., vol. 54, no. 4, pp. 389–393, Sep. 2010.

[14] M. Stefik, "Planning with constraints (MOLGEN: Part 1)," Artif. Intell., vol. 16, no. 2, pp. 111–139, 1981.

[15] K. Christoffersen and D. D. Woods, "How to Make Automated Systems Team Players," Adv. Hum.

Perform. Cogn. Eng. Res., vol. 2, pp. 1–12, 2002.

[16] I. D. Steiner, Group process and productivity. Academic Press, 1972.

[17] D. a Norman, “Cognitive Engineering,” User-centered Syst. Des., pp. 31–61, 1986.

[18] T. W. Fong, “Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation,” no. CMU-RI-TR-01-34. Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2001.

[19] M. Quigley et al., “ROS: an open-source Robot Operating System,” Icra, vol. 3, no. Figure 1, p. 5, 2009.

[20] T. W. Malone and K. Crowston, “The interdisciplinary study of coordination,” ACM Comput. Surv., vol. 26, no. 1, pp. 87–119, 1994.

[21] E. Salas, M. A. Rosen, C. S. Burke, and G. F. Goodwin, “The wisdom of collectives in organizations: An update of the teamwork competencies,” in Team effectiveness in complex organizations: Cross-disciplinary perspectives and approaches, New York: Routledge/Taylor & Francis Group, 2009, pp. 39–79.

[22] P. J. Feltovich, J. M. Bradshaw, W. J. Clancey, and M. Johnson, “Toward an Ontology of Regulation: Socially-Based Support for Coordination in Human and Machine Joint Activity,” in Engineering Societies in the Agents World VII, vol. Lecture No, G. O’Hare, M. O’Grady, A. Ricci, and O. Dikenelli, Eds. Heidelberg, Germany: Springer, 2007, pp. 175–192.

[23] M. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-Based Multirobot Coordination: A Survey and Analysis,” in Proceedings of the IEEE, 2006, pp. 1257–1270.

[24] A. Kirlik, R. a Miller, and R. J. Jagacinski, “Supervisory control in a dynamic and uncertain environment II: A process model of skilled human environment interaction,” IEEE Trans. Syst. Man. Cybern., vol. 23, pp. 929–952, 1993.

[25] J. M. Bradshaw, P. J. Feltovich, and M. Johnson, Human-agent interaction. 2011.

[26] K. Erol, J. Hendler, and D. S. Nau, “HTN planning: complexity and expresivity,” AAAI, pp. 1123–1128, 1994.

[27] I. Georgievski and M. Aiello, “An Overview of Hierarchical Task Network Planning,” 2014.

[28] J. E. Driskell, G. F. Goodwin, E. Salas, and P. G. O’Shea, “What makes a good team player? Personality and team effectiveness.,” Gr. Dyn. Theory, Res. Pract., vol. 10, no. 4, pp. 249–271, 2006.

[29] M. Johnson et al., “Team IHMC’s Lessons Learned from the DARPA Robotics Challenge: Finding Data in the Rubble,” J. F. Robot., vol. 34, no. 2, pp. 241–261, Mar. 2017.

[30] C. Beierl and D. Tschirley, “Unmanned tactical autonomous control and collaboration situation awareness,” 2017.

[31] M. Johnson et al., “Team IHMC’s Lessons Learned from the DARPA Robotics Challenge Trials,” J. F. Robot., vol. 32, no. 2, pp. 192–208, Mar. 2015.

[32] J. E. Allen, C. I. Guinn, and E. Horvitz, “Mixed-Initiative Interaction,” IEEE Intell. Syst., vol. 14, no. 5, pp. 14–23, 1999.

[33] D. McDermott, “Artificial intelligence meets natural stupidity,” ACM SIGART Bull., 1976.

[34] R. Brooks, “The Seven Deadly Sins of AI Predictions,” MIT Technology Review, 2017. [Online]. Available: https://www.technologyreview.com/s/609048/the-seven-deadly-sins-of-ai-predictions/?utm_campaign=add_this&utm_source=email&utm_medium=post. [Accessed: 17-Nov-2017].