Int J Artif Intell Educ (2018) 28:286–313
DOI 10.1007/s40593-017-0151-2

ARTICLE

Creating a Team Tutor Using GIFT

Stephen B. Gilbert[1] · **Anna Slavina**[1] · **Michael C. Dorneich**[1] · **Anne M. Sinatra**[2] · **Desmond Bonner**[1] · **Joan Johnston**[2] · **Joseph Holub**[1] · **Anastacia MacAllister**[1] · **Eliot Winer**[1]

Published online: 12 September 2017

Abstract With the movement in education towards collaborative learning, it is becoming more important that learners be able to work together in groups and teams. Intelligent tutoring systems (ITSs) have been used successfully to teach individuals, but so far only a few ITSs have been used for the purpose of training teams. This is due to the difficulty of creating such systems. An ITS for teams must be able to assess complex interactions between team members (team skills) as well as the way they interact with the system itself (task skills). Assessing team skills can be difficult because they contain social components such as communication and coordination that are not readily quantifiable. This article addresses these difficulties by developing a framework to guide the authoring process for team tutors. The framework is demonstrated using a case study about a particular team tutor that was developed using a military surveillance scenario for teams of two. The Generalized Intelligent Framework for Tutoring (GIFT) software provided the team tutoring infrastructure for this task. A new software architecture required to support the team tutor is described. This theoretical framework and the lessons learned from its implementation offer conceptual scaffolding for future authors of ITSs.

Keywords Team tutoring · GIFT · Architecture · Evaluation

✉ Stephen B. Gilbert
gilbert@iastate.edu

[1] Iowa State University, Ames, IA, USA

[2] Army Research Laboratory, Orlando, FL, USA

Introduction

For decades, a wide variety of ITSs have been used to train individuals on specific tasks (Capuano et al. 2000; Corbett et al. 1997; Koedinger et al. 1997). ITSs exist on a spectrum of computer aided instruction (CAI) (Shute and Psotka 1994). The most rudimentary CAI system has a basic input-output structure wherein the learner provides an answer to a problem posed by the computer, and it provides feedback. On the other end of the spectrum are the more advanced "intelligent" systems, which tailor feedback to the learner based on his or her knowledge acquisition and performance in real time. With increasing complexity, ITSs become more difficult to author and require multiple interacting components. Generally, four modules make up the basis of an ITS: a domain model, a learner model, a pedagogical model, and an interface model (Buche et al. 2004). One might conjecture that at least one new module would be required for a team tutor. The difficulties of authoring an ITS for an individual to perform a specific task have been well documented. This process includes creating several modules that work together to monitor learner progress, assess errors, and provide suggestions relevant to these errors in a manner that is not confusing to the learner (Gilbert et al. 2015b; Murray 1999; Murray et al. 2003). Thus, it does not come as a surprise that far less success has been documented in creating ITSs for training teams (e.g. Gilbert et al. 2015a).

The Generalized Intelligent Framework for Tutoring (GIFT) allows authoring of an ITS for individuals that can train learners in a variety of domains and environments (Sottilare et al. 2012). GIFT includes the components that are traditional in an ITS, as well as a few additional modules: a sensor module for interfacing with various biometric sensors, and a gateway module that allows GIFT to communicate with third-party programs. In this article, these components were adapted and modified to create a team tutor. The case study that follows includes a description of team tutor terminology adopted to ease communication internally about team tutoring, a description of the team tutor task itself, details on adaptations to GIFT's architecture to support teams, and a generalized framework for creating team tutors based on these experiences.

The goals of this paper are 1) to layout the general process of developing a team tutor and the difficulties that arise during the design process, 2) to describe the application of this process in the form of a case study of the creation of a team tutor authored in GIFT, and 3) to recommend future improvements to GIFT to support team tutoring further. While a formal evaluation of the team tutor described in the case study is ongoing, this exploratory research contributes terminology and an approach to team tutors that establishes a foundation for conceptualizing team tutors during the authoring process, and it should be helpful for those interested in developing any team tutors, as well as offer specific guidance if the tutor is implemented in GIFT.

Background

The Difficulties Posed by Team Tutoring

One of the difficulties posed by team tutoring arises from a team being more than just a group of people working on the same task. A group, like a team, is comprised of many

members, but unlike a team, as a group increases in size, the contributions of each individual decrease (Whatley 2004). A team is usually smaller than a group, and each individual member contributes however he or she can towards a shared objective. How the members interact with each other within a team is important. A team is a group of people who work together to accomplish something that would be accomplished less effectively, if at all, by any individual working alone (Yin et al. 2000). While the focus on individual training can be on the outcome or product of the training, the focus in team training is often on the process, which includes how team members communicate and make decisions together (Cannon-Bowers and Salas 1997).

Another difficulty in designing a team tutor is the number of variables that must be assessed. A team tutor must assess not only overall team achievement, but also individual contributions, sub-group interactions, team cohesion, and social constructs that are present in a team context but may be absent from individual or group work (Hughes et al. 2016). While an individual tutor may be able to estimate the mental model that a single trainee is working from, a team tutor can ideally infer the shared mental models present among multiple team members. A shared mental model is defined as a shared understanding among team members of the task along with expectations for how the task is to be completed. The shared model could be used by individual team members to predict the behavior of fellow team members, which facilitates coordination between them (Jonker et al. 2011). Team knowledge (Cooke et al. 2000) is a related construct. Team knowledge, as the team practices a task, may emerge separately from team members' individual knowledge. There is also a distinction between task work and teamwork (Schaafstal et al. 2001). Task work is described as the tasks completed by the team as part of an overall goal, while teamwork is the process by which this is achieved. All of these must be taken into account when considering how to model a team within the intelligent tutoring context.

A team ITS has to not only train individuals in completing a task, but must also teach team skills related to the "9 C's" noted by Salas, Shuffler, Thayer, Bedwell, and Lazzara (2014), which are absent from individual ITSs: cooperation, coordination, cognition, conflict, coaching, communication, context, composition, and culture. An individual tutor must keep track of each step involved in a task, the progress of the individual in completing the steps, the order in which the steps must be completed, and any causal relationships between steps. A team tutor must additionally indicate which team member is responsible for each step in the process and take into account the interdependency between teammates (Rickel and Johnson 1998). As Jones (1974) noted, team members' team skills can be quite dynamic. When measured, team skills can vary significantly based on the team task at hand. As shown in Fig. 1 (bottom), creating a team tutor requires at least three new modules: an expert model for team skills, a pedagogical model that can decide when to give feedback to specific members vs. the whole team, and a team model to record the current team skills of the team. Also, there will be a new individual learner model for each individual.

Both task work and teamwork are present during a team task. This fact implies that each behavior by a team member can have two or more traceable meanings for a model-tracing ITS, one for the task skills model, and one for the team skills model. For example, if Alice said to Bob, "Enemy spotted!" then that action would qualify both as successful enemy identification (a task skill) and communication (a team skill). If Alice took this action even though it was Bob's normal role to spot enemies, but Alice

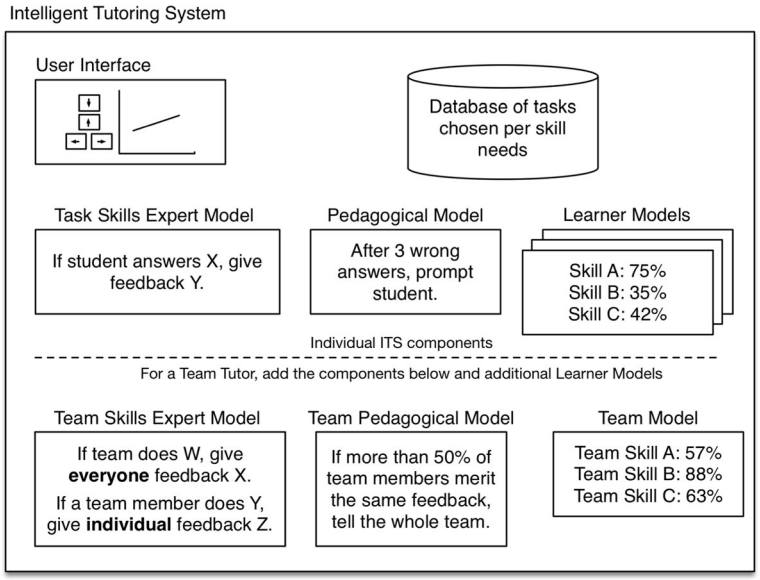

Fig. 1 The basic components of an ITS (top) and the additional components required for a team tutor, inspired by (Sottilare et al. 2011). The boxes contain example content

noticed Bob was overtasked, then Alice's action would also qualify as backup behavior, the team skill that allows team members to take over the responsibilities of other team members when necessary (Porter et al. 2003; Salas et al. 2014). While this one-to-many mapping of learner behaviors to ITS model elements is not unique to team tutors, we suggest that it is dramatically more frequent with team tutors, and adds to the complexity of their design. In terms of software architecture, the representation chosen for that action needs to work smoothly with both the task and team models. That is, if Alice's action were coded simply as a "communication," then it would have been noted by the team model but perhaps not the task model. If it were coded as "enemy identification," then the reverse could be true. If it were dually coded as "communication: enemy identification," then both models would have an element of the action to process.

Before a team tutor can be created, a team task must be chosen that 1) requires learners to use skills that are tutorable and 2) enables learners to be assessed. Because of the limits of technology in assessing subtle social interactions, it can be difficult to locate such as task. In the case of an individual tutor, the task only needs to include interactions between the individual and the task environment, which the tutor can measure. While characterizing these tasks and their associated grading schemes can be complex, once they are completed, evaluating the performance of an individual is fairly straightforward. Each action taken by the learner corresponds to an occurrence within the environment independent of any other actions. In a team task, however, actions are interdependent and the task has to reflect this interdependency.

Another challenge arises from the cognitive complexity of team tutoring: team members may simply ignore the tutor. In a group tutoring study led by Kumar et al. (2010), in which members were less interdependent than in a true team, group members that used a conversational virtual tutor often ignored the tutor. When the tutor had to

compete for attention with other group members, it could lose. The group task did not require participants to take immediate action in response to a problem. Therefore, it is expected that in a team task, which may demand immediate attention and higher independency with team members, members may also ignore the tutor.

Another major difficulty in team tutoring is determining when feedback should be given. While this difficulty may exist in all forms of training (Hattie and Timperley 2007), it can be exacerbated when both team and task skills are being trained. A team task should involve team members interacting in real time so that they coordinate their actions. A virtual environment in which learners were presented events to which they had to coordinate a response (Schaafstal et al. 2001) provided an example of the difficulty in assessing which skills were actually trained over the course of the task. The feedback that was delivered to the learners occurred after the task was completed which made it difficult to determine whether the desired skills were obtained over the course of training with the team tutor.

Previous Team Tutors

A small number of team tutors have been created for a variety of tasks and with various configurations, with some featuring groups rather than teams. In the 1990s, the Advanced Embedded Training System (AETS) was designed to facilitate team-based training a Naval Air Defense Team (Zachary et al. 1999). AETS featured instruction for individual operators based on tracking keystrokes, speech, and eye movements and a comparison of operator behavior with expected behavior. The AETS also monitored individual and team performance and provided a dashboard of relevant information to human trainers, but it did not have an architecture in place to offer automated team feedback.

The following examples feature ITSs that involve multiple learners, though they may not be a "team" in the usual sense. In one instance, a group tutor was created that interacts with multiple learners within a collaborative learning environment (Kumar et al. 2010). The human learners communicated with each other and their tutor in a chat environment and were tasked with completing a collaborative design for a competition. The tutor in this case functioned as a conversational agent that spoke with the learners and interacted with them in a social form rather than just as a pedagogical agent. This type of tutor allows learners to receive feedback quickly, but might not be suitable for a situation in which the learner is engaged in a task that is continuous. Within a military context, such as the one in which our tutor was designed, it is important for learners to be able to receive feedback without interrupting what they are doing, so a social tutor would not fit.

In another case, a similarly socially interactive tutor was created that was actually part of the learner's team (Traum et al. 2003). The tutoring agent behaved as both a teammate and a tutor within a virtual environment in which the learner was trained in completing a peacekeeping scenario. By embedding the tutor as a teammate within the scenario, this approach to team tutoring created artificial constraints on the interactions between team members. While human team members can freely interact with each other, build rapport, and establish social connections, a virtual tutor has a limited social ability to react to its teammates. This puts a limitation on the types of team interactions that could occur with such a tutor and may make it more difficult to transfer the skills

and behaviors acquired with the tutor teammate to fully human teams. A similar tutor/teammate system was created to train individuals to complete complex tasks that required a hands-on learning experience (Rickel and Johnson 1998). This tutor took the basic model of an individual tutor and added a layer of complexity by introducing different roles for the trainees, which indicated to the tutor the tasks each individual needed to complete and how these tasks might be interdependent with others. The focus of the tutor was not purely on interactions between team members, but rather on the completion of tasks. A similar study focused on the roles that learners took on within their teams (Singley et al. 1999). A team tutor designed by a subset of the current authors (Walton et al. 2015a; b) focused on a task that required teams of three learners to navigate a virtual shopping mall and coordinate their purchases according to a set of rules. The tutor in this case provided feedback in a non-invasive fashion during the task itself so that the feedback could be acted upon immediately. Unfortunately, the feedback provided by this tutor was unable to capture the participants' attention, which was held by the task itself.

While several examples of team tutoring systems exist, few research papers actually describe the metrics used to evaluate the team skills being trained. Yin et al. (2000) focused on describing how team interactions can be modeled using a tutoring agent, but did not describe how the outcomes of their practice-based tutor could be measured. Similarly, Schaafstal et al. (2001) described how team training should be conducted in the realm of emergency management, but did not go into detail about how the primary skills being trained could be measured. They described using observable behaviors to create after-action reviews for providing feedback on trainee performance after completion of a task, but did not indicate how these behaviors were measured or by whom. Before a tutor can be considered, a team task must be created with measurable outcomes. While much of the previous literature focuses predominantly on the architecture of the tutor (Piramuthu 2005; Rickel and Johnson 1998; Schaafstal et al. 2001; Traum et al. 2003), it is important to carefully consider the task that is being tutored, as its structure impacts everything from how the learners interact with the virtual environment to the metrics being used by the tutor to assess these interactions. Therefore, it is important to determine the measures and how they will be operationalized as soon as possible in the team tutoring development process.

In the case of developing a team tutoring framework for use with GIFT, it is also important to incorporate flexibility in the measurement of performance for individual team members, as well as the overall team, since performance measures may differ greatly between domains. GIFT is a platform created to simplify the design and implementation of ITSs by removing the need to recreate the entire ITS when switching to a new task or a different program. GIFT accomplishes this through independent modules that can be reused in a variety of domains due to the domain independence of these modules (Sottilare 2014). Only GIFT's domain module contains content specific to the skills being trained and the scenario within which the training occurs. Theoretically, the information in the other modules could be reused in other tutors.

Feedback

Several key issues in the design of a team tutor relate to the design of the feedback. For instance, an ITS designer must decide when team members should receive the feedback

(Walton et al. 2014). Mid-task feedback may distract them, but after action reviews may not have the desired skill building effect. The influence of feedback is highly tempered by the type of feedback and how it is given (Hattie and Timperley 2007). The primary purpose of feedback is to try to reconcile the actual learner state with the desired learner state suggested by the goals of the tutor. In order for feedback to effectively bridge the gap between learner state and desired outcome, the feedback must give the learner some form of guidance towards that outcome. The most effective feedback, according to the model Hattie and Timperley (2007) propose, lets the learner know 1) what the goals are that he or she is meant to achieve, 2) how he or she is doing in achieving those goals, and 3) what he or she can do to make progress towards them. Feedback has also been shown to be effective in group tasks, provided that the content of the feedback is followed by at least one member of the group (Nadler 1979). One of the key facets of designing an ITS for teams is making sure that at least one team member is attending to the feedback. This is also one of the primary challenges.

Feedback content is also an important consideration in designing an ITS and can differ based on the context, the type of learner, and the goal of the tutor. Sometimes feedback can prevent learning, particularly if it takes the place of the internal thought and problem solving that the learner needs to acquire to complete a task (Salomon and Globerson 1987). Feedback must not replace cognition, but can be used as a reminder of how to complete a task or as an indicator that an error has been committed. When feedback is given during a test-like event, which could include any sort of binary evaluation (correct or incorrect), the feedback serves primarily as a way to correct errors (Bangert-Drowns et al. 1991).

The way that feedback is delivered to the learner is also important, because depending on how it is delivered, it can influence whether or not the participant notices it at all. While completing a complex visual task, visual stimuli that are not relevant to task might be easily ignored (Von Mühlenen et al. 2005). It might be useful to use auditory feedback that entails telling the learner what he or she is doing right or wrong. However, any type of auditory stimulus, including distracting information, might capture a learner's attention, because it is difficult to ignore auditory distractors when engaged in a difficult visual task (Tellinghuisen and Nowak 2003). This result indicates that auditory feedback must be carefully timed and worded so as to deliver useful information without needlessly distracting the learner. Auditory delivery of feedback may be sufficient if the learner is receiving it relatively infrequently, so that messages do not overlap. However, auditory feedback can become difficult to use when it is being delivered in real time and the learner is being evaluated on his or her performance of multiple tasks. This approach may cause overlap in the audio messages that are being presented to the learner, and they may be unable to determine which message corresponds to which action that has been performed. In this situation, it may be prudent to deliver the feedback visually, which requires being able to direct the learner's attention to that feedback.

When someone is experiencing high cognitive load, a uniquely colored stimulus can capture his or her attention (Burnham 2010). Color change could capture attention when it is surprising (Horstmann, 2002), and new objects can capture attention when the brightness of the objects changes relative to the background (Franconeri et al. 2005). Therefore, in order to capture visual attention when a learner is experiencing high cognitive load, feedback must be delivered in a color and brightness that are distinct from that task.

These studies all demonstrate different approaches to addressing parts of the team tutoring problem, while being grounded in a specific domain of interest. The difficulties that each of these approaches encountered are further magnified when generating a flexible team tutoring framework in GIFT. As GIFT allows for multiple domains and approaches to be used within it, more flexibility needs to exist in the ways that tutoring is authored, so that it can support reuse. While being domain-independent is a strength of GIFT, it also makes authoring more difficult when creating a team tutor.

A two-person team task was created as the domain content for a team tutor developed within the GIFT architecture. The GIFT software architecture was expanded to accommodate the needs of a team tutor beyond the existing modules that supported individual tutors. Next, terminology is introduced that describes various facets of team tutoring that may aid in interpreting the description of the case study. The terminology may also be helpful in future endeavors with authoring a team tutor. The steps of authoring a team tutor are outlined along with a description of the iterative approach taken to create the tutor.

Case Study: The Surveillance Scenario Team Tutor

A case study was developed to provide a framework for team tutors. Specifically, a military team task tutor involving teams of two people was created based on Army surveillance training as a demonstration task for team tutoring using GIFT. The military nature of the task arose out of the collaboration between our research team and the Army Research Laboratory in developing both the task and the tutor. Our goals for the creation of this team task tutor included:

1. Define terminology to describe the various components of team tutors
2. Develop a military-relevant task scenario that was scalable in difficulty
3. Develop GIFT Team Tutoring Architecture software requirements
4. Generalize a framework for creating team tutors based on this case experience and creating other team tutors.

Goal 1: Team Tutor Framework & Terminology

The wide variety of tutors described above demonstrate that it can be difficult to compare team tutors based on different terminology and differing development approaches. What follows is a suggested set of terms that may be useful in describing team tutors to distinguish them from individual tutors. Standardizing terminology makes it easier to share ideas across platforms and collaborate on the creation of new tutors. The following is a proposed set of terms that can be used to define specific aspects of a team tutor and to describe it. The terms are based on conceptualizing the team tutor as an ontology, analogous to an object-oriented hierarchy in programming context. For example, if a team has a member, who has a role, which has a task, which has a skill requirement, then that team will possess all the skill requirements of its corresponding members' tasks. These terms might

be useful in the initial planning of a tutor when mapping out its purpose as well as how it is meant to function.

- **Team**: a team consists of **Members**, with one or more **Team Tasks**, and one or more **Team Skills**.
- **Member:** a member has one or more **Roles** and one or more **Skills**.
- **Role**: a role has one or more **Tasks**.
- **Task:** a set of actions that an individual role must perform. Generally a task consists of steps that must be completed often in a particular order. A task has one or more **Skill Requirements**, and optional **Team Skill Requirements**.
- **Team Task:** a task the team is responsible for. A team task is similar to a task but also includes interdependency between team members. A team task has one or more **Team Skill Requirements**.
- **Action:** a behavior by a Member that can be assessed, typically a substep of a Task.
- **Skill:** a member's level of performance on a particular behavior that is required for a task.
- **Skill Requirement:** a skill performance threshold that is required by a task to perform it appropriately.
- **Team Skill Requirement**: a team skill performance threshold that is required by a task or a team task to perform it appropriately.

Another set of terms is useful for distinguishing the assessment of a team or member's recent action (action performance), their current accumulated performance (performance state), and their overall task performance when the task is completed. Each of these performance assessments might use a different performance metric.

- **Action Performance:** the assessment of a member's action that just happened.
- **Performance state**: the current cumulative level of performance for a team member at a specific point in time, typically before task completion.
- **Task Performance:** a measure of a team's or member's performance on a task or team task, typically after completion.
- **Performance metric:** a formula used to measure performance on an individual or team task or action.

Goal 2: The Surveillance Scenario

Goal 2 above was to develop a military-relevant task scenario that was scalable in difficulty. The Surveillance Scenario involved an ongoing series of events designed to train small teams in surveillance. Because communication is a critical team skill, and communications are relatively easy to measure if implemented using structured language or another formal method, a simple task requiring only observation and communication was chosen. The task scenario was designed to be scalable in difficulty and agnostic of the tutoring approach, so that it could serve as an open-ended research platform for team tutoring. The Surveillance Scenario was situated in a virtual environment within Virtual Battlespace 2 (VBS2), which is software that facilitates the creation of custom military training scenarios in a 3D virtual space. It used a model similar to the

Event-Based Approach to Training (EBAT) approach (Fowlkes et al. 1998; Schaafstal et al. 2001) in which simulated events are created for teams to practice specific skills.

The Surveillance Scenario requires a team of two. Each member stands atop a building and is responsible for surveillance of a 180-degree zone (see Fig. 2 and Fig. 3). At the start of the scenario, enemies (referred to in the tutor feedback by the military term "OPFOR," which derives from "opposing force" and is a term for "enemy") emerge from behind walls, dart from place to place, and sometimes cross zone boundaries at the single green pole or double green poles. Each team member is responsible for alerting the other member if an enemy is about to enter the other team member's zone. The task was designed to expand to four players with 90-degree zones if needed. Each team member used the mouse to scan for enemies in his or her zone, looking back and forth because it was not possible to see the entire zone at one glance. Because of the extent of the building rooftop, each member could see only slightly into the other member's zone.

As a way of facilitating the creation of shared mental models among team members, the environment and scenario are structured symmetrically such that each team member is doing the exact same task as the other. A shared mental model within this environment makes it much easier for one team member to predict the behavior of the other team member based on mutual experience. For instance, if Bob sees many enemies approaching and has the experience of having to keep track of all of them to inform Alice, he would have a better understanding of what Alice is experiencing when she has many enemies approaching from her zone.

To be more specific, using the terminology recommended above; the team consisted of two team members. The tasks were:

- **Transfer**: Each team member had to alert the other member whenever an enemy was about to cross into the other member's zone. This had to be done verbally as well as by keypress.

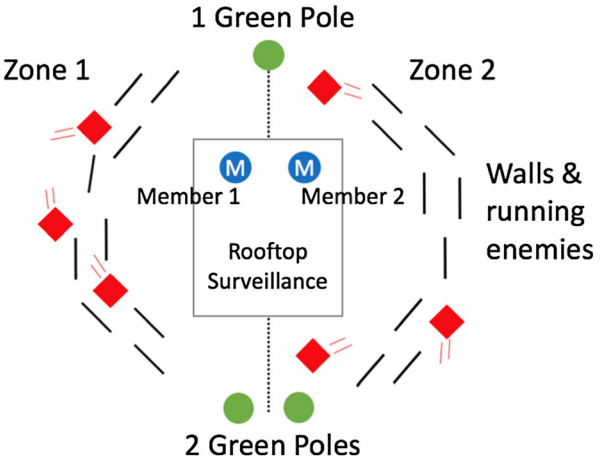

Fig. 2 Aerial view of Surveillance Scenario. Each team member (blue circle with M) was responsible for a 180-degree zone around a building and had to 1) alert the other member if an enemy (red diamond) was about to transfer into that member's zone, 2) acknowledge alerts, and 3) identify enemies that were transferred by the other team member

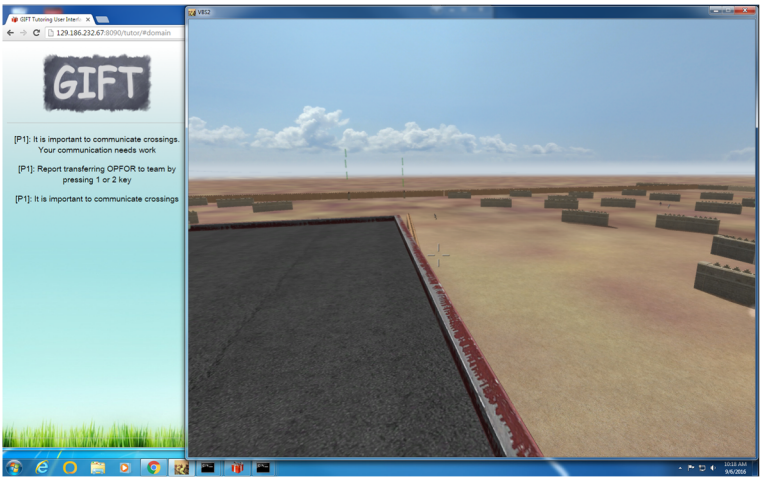

Fig. 3 This screenshot of the Surveillance Scenario Tutor shows Team Member 1 looking toward the boundary with two green poles. Enemies emerge from behind walls. GIFT provides feedback at left

- **Acknowledge**: Each team member had to acknowledge that a communication from the other team member had been received. This had to be done by keypress and accompanied with a verbal acknowledgment.
- **Identify**: After an enemy had been transferred from one team member to the other, the teammate receiving the enemy had to identify the enemy as soon as it entered his or her zone. If an enemy enters a team member's zone and the team member identifies it without the member's partner having announced the transfer, this was considered backup behavior.

A sample of the task mechanics follows. If the two team members were Alice (assigned to Zone 1) and Bob (assigned to Zone 2), then as an enemy in Zone 1 approached the zone crossing point with one green pole, Alice would press the 1 key to indicate person crossing at the 1-pole zone and say something like, "Person crossing at the 1." Bob would then press the E key to acknowledge the enemy, say something like, "Got it!" and then press the spacebar when he sees and identifies the enemy who just crossed. The closer the enemy is to the boundary at the moment of transfer, the better the performance (a premature transfer might lead the other team member to look to the zone too early and waste time watching for the enemy to arrive). If an enemy crossed at the two-pole zone, the member pressed the 2 key. These four keys, 1, 2, space, and E, were the only keys used, and were chosen so that team members could type them all with their left hands while using their right hands on the mouse to scan visually back and forth in their respective zones. The language spoken by team members was chosen by them. In early implementations of this task, the members were required to use specific military phrases designated by the research personnel, but this requirement was sufficiently onerous to prevent some teams from reaching consistent performance levels, even after multiple trials. Additional details on the creation and military basis of this task are described in detail in Bonner et al. (2015b), Bonner et al. (2016a), and Bonner et al. (2016b).

While the task context was relatively simple, the Surveillance Scenario became a powerful research testbed because of the ability to explore numerous team tutor research questions using the same task, such as how cognitive load affects ability to perceive feedback, whether knowing one's team member previously affects performance, how consecutive trials form a learning curve, how that learning curve is affected by different forms of feedback, and many others. Table 1 illustrates different dimensions of feedback that can be explored. For example, immediate delivery of feedback has been shown to be effective during tasks that imposed a higher cognitive load (Kulik and Kulik 1988), but feedback at too high a frequency could be overwhelming and result in the feedback being ignored or missed (Lam et al. 2011). The Surveillance Scenario could be used to explore how much feedback is too much, and the answer may depend on the team members themselves.

The task load on players can be adjusted by changing the number of enemies that appear, and players can be cognitively tasked quite heavily. The task is difficult primarily because a member must continuously scan the entire 180-degree zone while watching for enemies appearing (visual search), visually tracking enemies leaving the zone, and listening for transfer alerts. The cognitive complexity arising from this relatively simple scenario was surprising to many participants, who perceived the task at first as quite difficult, if not completely overwhelming. In future work, if additional teamwork tasks were needed, the authors have considered asking members to collaborate in keeping track of the total number of enemies in the entire field.

The Surveillance Scenario Tutor & Moving Average Assessment

Having created a task scenario that required team skills and that was scalable in difficulty, a tutor for this scenario was required. More specifically, a tutor was needed to increase team members' performance with each of the Surveillance Scenario tasks:

Table 1 Dimensions of feedback that can be explored with the Surveillance Scenario as a research testbed (Bonner et al. 2015b)

Dimension	Levels	How realized in Surveillance Task Testbed
Subject	*Individual, Team*	Tutor provides feedback about an individual team member or entire team
Target	*Public, Private*	Tutor provides feedback to one person (private) or whole team (public)
Timing	*immediate, after, omitted*	Feedback occurs based on patterns or task effectiveness during the task, or after the overall grade or rating is given. Feedback is omitted when an error is committed, but is not sufficiently important to interrupt training to provide immediate feedback or to be included in the After Action Review.
Type	*Proactive, reactive*	Proactive: Feedback before a learner makes error Reactive: Feedback after a learner makes an error
Specificity	*Generic, specific*	Generic: "*Good job, Soldier*" Specific: "*You missed an OPFOR located at 7 o'clock*"
Tone	*Positive, negative*	Positive: "*...you might want to try...*" Negative: "*...your poor performance is hurting the team*"
Style	*Collaborative, Competitive*	Collaborative: "Slow down scanning to help the team…" Competitive: "Your performance is worse than Joe's."

Transfer, Acknowledge, and Identify. The essence of creating a tutor is 1) defining the patterns of behavior that the tutor will watch for (assessment), 2) creating feedback to give based on these assessments, and 3) defining the exact rules or conditions that will trigger the feedback. Creating the Surveillance Scenario Tutor required the specification of performance metrics for every action taken (keystrokes types) and different performance metrics to translate those actions into task performance for each task. This section describes the approach to giving feedback within the Surveillance Scenario, and the following section describes how it was implemented using GIFT.

The GIFT software architecture formalizes performance assessment into three categories: Above-Expectation, At-Expectation, or Below-Expectation (Hoffman and Ragusa 2015; Sottilare et al. 2013). Thus, the approach to designing the performance metrics took a similar path, seeking to define performance on each task with analogous categories.

It is worth noting that because the Surveillance Scenario tasks require continuous ongoing repetitive performance, the assessment approach must evaluate not only each individual team member's actions (e.g., identifying an enemy), but also an ongoing pattern of performance that has accumulated to the current performance state. If Alice and Bob, for example, both identify 75% of the enemies, but Alice's missed enemies occurred sporadically through the past five minutes, while Bob's missed enemies occurred all in the past 10 s, Alice and Bob should probably not have the same accumulated performance state. To address this issue, an approach similar to a moving average was implemented. The tutor used the past five actions to identify the member's current performance state. As a result, feedback for behavior was given only after at least five actions had occurred, not every time a team member did something wrong, as is more typical in ITSs. To determine the window size of five actions, feedback was initially given after every action and the tutor was tested on pilot participants. Using eye-tracking and post-task interviews, it was confirmed that participants received much more feedback than they could process. Based on gaze movement, as well as discussions with the participants, the number of actions required for feedback was set at five.

While these details may seem esoteric or specific to the Surveillance Scenario Tutor, this moving average approach to performance assessment requires a technology architecture to support it, and this requirement drove some of GIFT architecture described below. In addition, this difference between action performance and performance state highlights the fact that a team tutor author may want to establish performance metrics that serve as triggers for giving feedback, but use completely separate performance metrics for actually measuring task performance.

The feedback implemented in GIFT for the Surveillance Scenario was dependent on the member's and team's performance. Team members who were performing above expectations did not need as much feedback, and it was reinforcing rather than corrective. Members performing at a low level needed to be given reminders of the task goals and how to accomplish them, lest they had forgotten some component of the overall task. This type of feedback, which can be considered formative (Shute 2008), occurred frequently as members learned the task and figured out timing. As such, some variations in feedback statements were created so as not to provide redundancy.

In our particular implementation with GIFT, feedback statements appeared in real-time during the task via text that appeared on the left side of the screen in a bright yellow flashing box. The text was black to contrast it from the background (see Fig. 3).

Accompanying the appearance of the feedback was a high-pitched beep that indicated to the learner, if the visual changes were not sufficient to capture attention, that there was a new feedback message.

Assessing the Identify Task

Each of the performance states for the Identify task was mapped to a feedback statement. The overall assessment and feedback is shown in Table 2. Feedback was given when there was a performance state change (e.g., from Above-Expectation to At-Expectation, or At-Expectation to Below-Expectation) or when the state had been maintained for the duration of the moving window. I.e., if a member identified 10 enemies, all within seven seconds after appearance, the member would see the At-Expectation feedback twice, once after the first five At-Expectation actions, and a second time after the next five actions. Using the moving average approach, feedback was given less frequently, but the tutor still updated the member with feedback on performance. Note that all participant actions were logged in GIFT for post hoc performance analysis, not just every fifth action.

Assessing Transfer & Acknowledge Tasks

Team members were assessed on the basis of the timing between when a transfer was initiated by one member and when an acknowledgment occurred from the other member. Transfer and Acknowledge were evaluated separately for individual assessment but as Transfer-Acknowledge pairs as a team measure (ideally, all Transfers by one team member would be paired with Acknowledges by the other team member). Overall assessment and feedback for the Transfer and Acknowledge tasks are shown in Table 3.

Collaborative Complexities

The tasks described may seem like an oversimplification of real-world teamwork that decontextualizes each action. However, this result is more a product of the method of analysis employed in GIFT rather than how teamwork was conceptualized by the authors. While beyond the scope of this paper, contextualized team behavior was

Table 2 Performance metrics and feedback for the Identify task. Timing thresholds are used to evaluate actions. Performance state is assessed using a moving window of past five actions

Identify Action Assessment	Identify Performance State Assessment	Corresponding Individual Feedback to Members
Above-Expectation: <= 5 s	Above-Expectation: If majority of past 5 actions are Above-Expectation with no Below-Expectation.	Excellent work identifying OPFOR.
At-Expectation: 5–10 s	At- Expectation: If majority of past 5 actions are At-Expectation	It's important to identify OPFOR as quickly as possible.
Below-Expectation: > 10 s	Below-Expectation: If majority of past 5 actions are Below-Expectation	1st time: Identify OPFOR immediately. 2nd + time: Remember to identify all incoming OPFOR.

Table 3 Performance metrics and feedback for the Transfer and Acknowledge individual tasks, and for the Transfer-Acknowledge team task

Transfer Action Assessment	Transfer Performance State Assessment	Individual Feedback to Members
Above-Expectation: N/A (task was binary)	Above-Expectation: Transfer occurred when OPFOR is at zone boundary	Good alerts about crossings.
At-Expectation: Transfer is announced for a crossing OPFOR.	At-Expectation: Transfer occurred shortly before OPFOR arrives at zone boundary	It is important to communicate crossings.
Below-Expectation: Transfer is not announced for a crossing OPFOR.	Below-Expectation: OPFOR passes into other team member's zone without a transfer occurring.	1st: Make sure your partner always knows when an OPFOR is about to cross. 2nd+: It is important report crossing OPFOR.
Acknowledge Action Assessment	Acknowledge Performance State Assessment	Individual Feedback to Members
Above-Expectation: N/A (task was binary)	Above-Expectation: acknowledge time – transfer time < = 1 s	Good acknowledgment of transfers.
At-Expectation: Member acknowledges a transfer.	At-Expectation: acknowledge time – transfer time > 1 s	Please confirm all transfers.
Below-Expectation: Member does not acknowledge a transfer.	Below-Expectation: acknowledge time – transfer time > 2 s	You need to confirm all transfers. Your communication needs work.
Transfer-Acknowledge Team Action Assessment	Transfer-Acknowledge Team Performance State Assessment	Team Feedback to All Members
Above-Expectation: N/A (task was binary)	Above-Expectation: acknowledge time – transfer time < = 1 s	Successful handoffs!
At-Expectation: For the last transfer from either member, there was an acknowledge from the other member.	At-Expectation: acknowledge time – transfer time < = 2 s and >1 s	It is important to alert each other about crossings and acknowledge them.
Below-Expectation: For the last transfer, there was no acknowledge.	Below-Expectation: acknowledge time – transfer time > 2 s	1st: Your team communication needs to improve. 2nd: Team, please keep up the communication. 3rd+: Work on communication.

analyzed as part of the ongoing analysis of the effectiveness of the Surveillance Tutor. This framing includes the concept of backup behavior, which was operationalized as an Identification occurring in the absence of a Transfer from a team member. If Bob is overwhelmed by incoming OPFOR and fails to alert Alice about a crossing, Alice can support him by identifying the OPFOR that Bob neglected to transfer (backup behavior). However, that Identification could also have just been a fortuitous action by Alice without her realizing that Bob was overloaded. The current tutor was not developed to use contextual markers to distinguish between true backup behavior and a fortuitous identification. The ongoing evaluation of the Surveillance Tutor's effectiveness, to be discussed in future work, will attempt to include this sort of analysis as a measure of team performance.

Goal 3: The GIFT Architecture that Supported the Surveillance Tutor

There are various tutor authoring tools that one could use to create a team tutor. It would also be possible albeit tedious to author a tutor from scratch. When deciding which tool to use, considerations should be made regarding the flexibility of the tool. Authoring tools can make tutor creation easier by providing an interface that does not require much coding to make something usable. For instance, a visual programming interface may simplify the creation of assessment conditions. One of the advantages of using GIFT to author a team tutor is its modularity. The pedagogical module, which delivers feedback to the learner, can be modified separately from the leaner module to adjust feedback frequency to accommodate the cognitive load imposed by the task. Previous team tutors have used agent-based models to delegate the various components of the tutoring process (Kumar et al. 2010; Piramuthu 2005; Rickel and Johnson 1998; Traum et al. 2003;). The agents used in agent-based ITSs are self-contained processes that continuously communicate with each other (e.g. Piramuthu 2005). While GIFT does not explicitly use agents, the modules that it employs can be considered a similar architecture.

Figure 4 and its caption describes the overall GIFT architecture and its particular implementation for the Surveillance Tutor. In order to display the feedback intended for each user, GIFT's domain modules are coupled with the VBS2 simulation software by utilizing an asynchronous messaging system, ActiveMQ. The domain module takes information from the scenario being trained, evaluates performance, and provides feedback to the learner. Further, GIFT's domain module is the only module that contains domain-specific information. Therefore, it is theoretically possible to reuse developed tutoring procedures in new contexts and domains by switching out the materials and feedback contained in the domain module.

The domain module knowledge is stored in Domain Knowledge Files (DKFs) that contain the feedback and performance assessment metrics encoded using XML. To accommodate the assessment of multiple individuals as well as a team, a customized version of GIFT was created that enabled team tutoring. The following section describes some of the customizations of this Team Architecture. Additional details, for example, regarding synchronization of multiuser startup of the scenario, can be found in Gilbert et al. (2015a). The logic within the DKF can be characterized as, "For each concept to be learned, use an If-Then condition to assess the learner's action and update the learner's performance state (Above, At, or Below Expectation). If the learner's

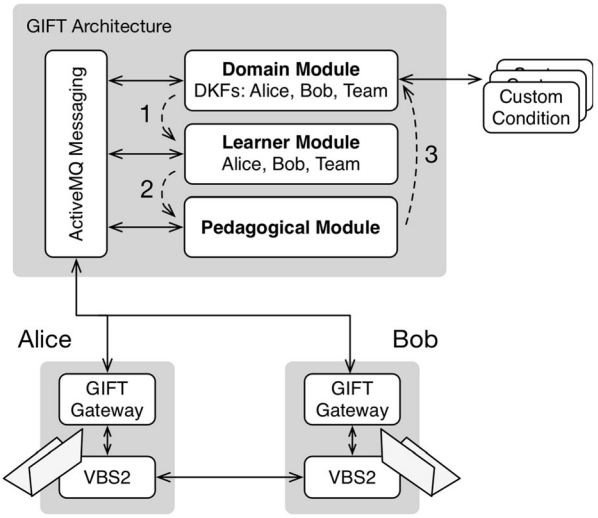

Fig. 4 Abstract representation of GIFT architecture generally and its particular implementation for the Surveillance Scenario Tutor. Alice and Bob's actions on their laptops in VBS2 are sent to the Domain Module via GIFT's VBS2 Gateway modules. The Domain Module (1) assesses them using domain knowledge files (DKFs) and passes the results to the Learner Module. The Learner Module (2) notes whether the learner's performance state has changed and passes the results to the Pedagogical Module. The Pedagogical Module (3) recommends an approach to feedback, and the Domain Module's DKFs, which contain the feedback statements corresponding to each approach, send the feedback to Alice and/or Bob. Custom condition files (top right) are programmed as Java classes by tutor authors to enable the DKFs to assess scenario-specific actions. The Surveillance Tutor's custom conditions include Identify, Transfer, and Acknowledge

performance state has changed, give appropriate feedback." Note the key distinction here between assessing a real-time action and updating the cumulative performance state based on that action.

The DKF If-Then logic contains several constraints that were inherent to the original GIFT architecture. One constraint of this approach is that the granularity of the evaluation is fixed at the three categories of Above, At, or Below Expectation. However, a tutor author can achieve a greater granularity of assessment by breaking down concepts into sub-concepts. Another constraint of this approach is that the DKF only gives feedback when a concept or sub-concept performance state has changed. Thus, it is not typically possible to give feedback confirming ongoing good performance ("Good job…"), since the state of Above Expectation has not changed, or ongoing poor performance ("You still need to work on…") since the Below Expectation state has not changed. A third constraint is that default DKF conditions focus on updating the performance state based on the most recent action; there is no history of learner actions stored. Thus, it is not possible in the default GIFT architecture to have conditions based on historical patterns of behavior.

New GIFT Approach: Smart Custom Conditions to Assess Patterns of Behavior

While some conditions are included with GIFT, e.g., to assess whether an entity is at a specific position within the scenario, a tutor author must typically program additional custom conditions as Java classes for a particular training scenario. To implement the

desired Surveillance Tutor described above, Java custom condition classes were created that were fairly complex, what one might call "smart conditions," in that they overcame several of the constraints described above. Conditions were developed for evaluating the tasks of Transfer, Acknowledge, and Identify. For Transfer, for example, the condition would evaluate whether a player entered the 1 or 2 keystroke at the correct time as an enemy approached a zone boundary to cross it. Feedback could be given, then, if the key was entered too early or too late (see Table 3). The condition for Acknowledge did not concern itself with the position of the enemy, but assessed how much time had passed since the Transfer from the other player (also see Table 3). Note that because the Transfer and Acknowledge tasks are supposed to be done in pairs by both learners (one transfers and the other acknowledges), this custom condition embodied some of the team dynamics within itself, watching for one learner's transfer and then recording the time until the other player's acknowledgment arrived. The Identify condition used the moving average approach which was discussed above; technical details follow below.

To avoid giving feedback on a per-action basis (the first column of Table 2 and Table 3), the conditions used a custom formula based on a history of actions to assess the ongoing performance state (the second column of those tables), adding a layer of abstraction to the representation of performance (see the discussion of the moving average assessment above). In addition, inside the customer condition classes, the default performance state for every action was maintained as "Unknown" (rather than Above, At, or Below Expectation) to allow triggering feedback by changing the Unknown performance state to Above, At, or Below (a state change). This approach allowed the custom condition classes to 1) maintain a history of the user's performance (e.g., the last five events) and 2) tell GIFT to provide feedback for ongoing good or bad performance even though a member's or team's official GIFT state performance state (outside the custom conditions) had not changed. This method was critical for enabling the management of the overall frequency of feedback experienced by learners. This approach could be used not only with team tutors, but also with traditional ITSs for individual learners.

Using smart custom conditions like these to augment the standard DKF approach has the advantage of specialized customization, but the disadvantage of storing some of the intelligence of the ITS outside GIFT, which reduces the modularity of the tutor and the possibility for re-use in other tutors. As more team tutors are created, it is possible that frequently needed smart conditions concepts will be incorporated into the pedagogical module in the GIFT team architecture. The team pedagogical module may be adapted to manage overall levels of feedback, possibly even customizing them based on members' individual cognitive load during the task.

Domain Module Customization: Multiple Simultaneous DKFs

The domain module created for the Surveillance Scenario consisted of three DKFs, two representing the two individual team members, and one representing the team. Each team member's DKF enabled the evaluation of how he or she performed on the tasks of identification, transferring, and acknowledgment at an individual level, and contained the individual feedback statements. The team DKF evaluated the team's performance and contained team feedback statements. The original GIFT architecture did not support

multiple simultaneous DKFs within a single session. However, the GIFT team architecture simulated three tutors running simultaneously during a session, one for each person and one for the team. Although it is tempting to envision these DKFs as hierarchically arranged, with the team DKF "above" the individual DKFs, in the software architecture they are not. The DKFs are differentiated simply in the kinds of data they process – the individual DKFs evaluate individual performance metrics, and the team DKF evaluates team metrics, which may or may not include individual performance metrics as components of team performance.

It is worth noting that this new team architecture faces a scalability challenge in two ways. With a team of three members X, Y, Z, if it were desired to give every combinatorial subset of team members specific feedback, one would need six separate DKFs for X, Y, Z, X&Y, X&Z, Y&Z, and X&Y&Z, each with redundant XML code when repeated feedback statements are needed. This is a scalability challenge because of the quickly growing number of DKFs. Second, because the DKFs in the current architecture are independent, it is not easy to create **compound conditions**, i.e., performance state assessments that depend the actions or performance state of multiple team members simultaneously, (e.g., "If X is doing well as transferring, but Y is not, then…"). Theoretically, such compound conditions would be feasible within the Team DKF, but we had difficulty getting this additional complexity to work within the extended GIFT architecture, and our Surveillance Tutor did not require them.

With no compound conditions, the reader might wonder why a team DKF was needed at all in the Surveillance Scenario. There are two characteristics of a team DKF. The first is compound conditions (which depend on multiple team members), and the second is team feedback, in which multiple team members receive the same feedback. In the Surveillance Scenario, the team DKF assessed using only simple conditions, but offered team feedback (see the bottom third of Table 3).

The tutors based on each DKF do not know currently about each other, and therefore each individual tutor's performance state assessments cannot be shared amongst all tutors. Similarly, because the DKFs for members and the team are independent, a team member may receive feedback messages as a result of both the individual member's DKF and the team's DKF, as if two human tutors were coaching each person simultaneously without consulting with one another. While the messages' content may not conflict, it would be better for the team architecture have a single "executive" module that would oversee the overall feedback experience of each member and provide filters for other modules.

Goal 4: Team Tutor Framework - Steps of Authoring a Team Tutor

Based on process of developing the Surveillance Team Tutor, along with experience creating the aforementioned shopping mall team tutor (Walton et al. 2015a, b) and a team tutor for a three-person search and destroy mission (currently in progress), a generalized framework is offered for creating team tutors. While this framework is based on authoring a team tutor in GIFT, it is intended to aid future team tutor authors in creating all necessary coordinating components for a team tutor in any software tool. The first element of the framework is the terminology defined above. The other elements are the ten steps for authoring a team tutor described below and in Fig. 5, which have been adapted and expanded from the eight steps described in Bonner et al. (2016a).

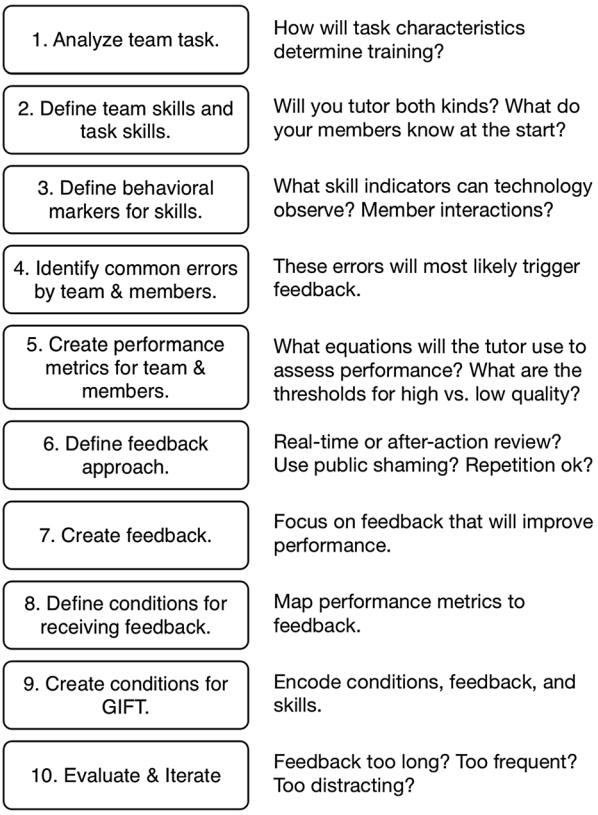

Fig. 5 The steps of authoring a team tutor

Developing a team tutor involves some of the same general steps as authoring an individual tutor, but also includes additional layers of complexity to accommodate the team aspects.

Analyze Team Task First, an existing team task must be chosen and analyzed to define the characteristics of it that will be adapted for tutoring or team training. This step can be one of the most difficult, because existing team tasks are often accompanied by procedural documentation that describe the goals and duties of each team member, but rarely are the required team skills documented. Also, some team tasks are flexible, e.g., a task could be done by a team of 3–5 people, in a variety of settings. When building a team tutor, usually, some of that flexibility will be sacrificed. The tutor author must settle on one scenario, one team size, and other specific parameters that will allow the tutor to be a representative sample of the real-world task, narrowing the potential scope of the learning, but allowing more specific assessment of learning outcomes.

Various properties of the team task must be noted to facilitate choosing the tutor's approach. Does this team task have a leader, or not? Do all members of the team have the same role or are there be specific tasks that each member is responsible for? How much interdependency do the team members have? In Bonner et al. (2015a), a

taxonomy of team structures offers a more detailed list of characteristics, e.g., the form of leadership, format of communication, roles, and relative locations of team members (co-located or distributed), member skill characteristics, members' skills, and the degree of learning culture on the team. All of these characteristics can affect the design of the team tutor. The most critical question at this juncture in authoring is whether tutoring or coaching provided by a computer, either in real-time or in after-action review, could actually help the team perform better.

Define Team Skills and Task Skills An underlying assumption of the idea of "team skills" is that they are applicable across domains. I.e., a team with excellent team skills will perform well on new novel task X more quickly than a team with fewer team skills. Ellis et al. (2005) found generic teamwork skills training showed higher cognitive and skill based outcomes within small team tasks. Similarly, Prichard et al. (2006) noted that generic teamwork training produced better performance than no teamwork training, and that groups that both trained and completed their task together performed higher than those who received generic teamwork training but were reassigned to new teams to complete the task. Thus, we can reasonably argue that a team may benefit from team skill training even in training contexts not directly related to their duties.

In 2014, Eduardo Salas et al. (2014) published a guide to teamwork that offers a meta-analysis of multiple team articles and distills the results to suggest that team performance is influenced by the 9 C's: cooperation, coordination, cognition, conflict, coaching, communication, context, composition, and culture. Sottilare et al. (2017) subsequently published a meta-analysis of team tasks that offered specific behavioral markers for many of the C's. However, operationalizing the measurement of these team constructs remains a challenge that varies based on the team and the team task. How communication, for example, will be measured in a particular team task may depend on the instructional designer and the task itself. A team tutor author will need to choose which team constructs among these and others, such as trust and backup behavior, are most germane to the specific domain of the tutor. Then, the author will need to define performance metrics for each team skill.

While domain knowledge in GIFT is currently encoded in a DKF, it may be that a future GIFT architecture designed more specifically for teams would feature a team knowledge file (TKF), which would store team work knowledge alongside the task work DKFs.

Define Behavioral Markers for Skills The critical challenge with any skill definition is establishing the behavioral markers that can be used as evidence for that skill. The method of interaction with the user interface must be determined such that the aspects of behavior relevant to accomplishing the goal can be detected, measured, and graded by the tutor (Traum et al. 2003). For team constructs such as the 9 C's, researchers have proposed principles that can be used to choose behavioral markers (Rosen et al. 2011), as well as specific examples (Sottilare et al. 2017). When the mapping between a behavioral marker and the skill is uncertain, ITS authors have frequently used a Bayesian approach to estimate the probability of an action representing a lucky guess or an accidental slip (Corbett and Anderson 1995). This approach can be combined with groups of behavioral markers for higher accuracy.

Identify Common Errors by Team and Members One of the key elements of the knowledge elicitation process when capturing the knowledge of an expert in an ITS is noting the common errors made by participants. Common methods for this process include the Critical Decision Method, the Knowledge Audit, and direct observation (Crandall and Hoffman 2013). However the common errors are identified, having a team rather than just an individual learner multiplies the number of ways that things can go wrong, because there can be errors in both task skills and team skills.

Create Performance Metrics for Team and Members While Step 3 defines the behavioral markers that might indicate the presence of a skill, this step requires using those markers, other knowledge of teamwork, and the task domain to define a performance assessment for each skill that needs to be learned. It may be that multiple metrics are required to fully assess task performance, such as, in our Surveillance Scenario, we assessed a team member's communication skills both with a metric based on the number of Transfers announced and with a metric based on the number of Acknowledgments given.

Define Feedback Approach Significant decisions must be made here, many of which are highlighted in Table 1 in the discussion of feedback design. Will feedback arrive in real-time or after the performance? Will feedback be text, audio alerts, pre-recorded audio, or in what form? Will feedback be repetitive? Will feedback be addressed to individuals or to the entire team, or a mix? Will the feedback intended for individuals appear only to those specific individuals or to the entire team? How frequently will feedback occur? Will the answer to any of these questions depend on an individual member's performance? These decisions will have a widespread impact on the team tutor, and then must be carefully considered.

Create Feedback This step goes hand in hand with Step 6, but the words of the feedback messages themselves must be carefully crafted. Messages that are simply too long can be ignored by members under heavy cognitive load. As described above, Hattie and Timperley's (2007) review of feedback suggests that good feedback will remind the learner of his or her goals. In the case of a team tutor that includes team skills, the feedback must remind the members of their goal of performing smoothly as a team as well.

Define Conditions for Receiving Feedback This step combines the performance metrics (Step 5) with the feedback (Steps 6 and 7). A performance metric typically offers a mathematical approach to scoring action performance, performance state, or task performance, but this step requires defining the metrics that will be used to trigger feedback, which may or may not be tied to task performance or even performance state. A more complex scenario might involve cascading conditions or multiple metrics. For example, if metric X is greater than 50 and metric Y is less than 0.5, then given feedback F.

Encode Conditions in the Tutor In this step, the tutor author must take encode the information from Step 8 (which is based on Steps 5–7) into an actual software platform. This stage of tutor authoring typically requires computational thinking

(Blessing et al. 2009). Assuming one is using GIFT, the body of design work completed above is encoded into GIFT. Conditions, feedback, and performance metrics are added to DKFs. Custom conditions are programmed. In the future, this step will not be at the end of a waterfall development process, but carried out along the way in synchrony with the above steps via integrated agile design with well-designed team tutor authoring tools.

Evaluate and Iterate The resulting tutor will now run and can be pilot tested. From our experience, the first implementation will need to be tuned significantly. The timing dynamics and cognitive load experienced by participants is very difficult to estimate before a working prototype has arrived. Plan on extensive testing and further iteration.

While these steps may appear linear, the process of authoring a team tutor can be iterative at any point along the way. For instance, analyzing a team task may lay a foundation for the tutor, but the task may not be a perfect fit. In defining conditions for delivering feedback, one might discover that some tasks are too nebulous to evaluate with the existing technology. Similarly, while identifying common errors, one may discover that members perceive the scenario in a way that was not intended by the authors, which may lead to a reevaluation of the behavioral markers associated with the skills being taught.

Conclusions & Future Work

It was noted in the Introduction that team tutoring was difficult for several reasons and that the goals of this research were 1) to offer a generalized framework for conceptualizing and authoring team tutors, 2) to contribute guidance on authoring a team tutor using GIFT specifically, based on the case study of the Surveillance Tutor created using GIFT, and 3) to recommend future features for GIFT or other team tutoring platforms that would ease the creation of team tutors. While the evaluation of the Surveillance Tutor itself was beyond the scope of this paper (the evaluation is still underway), it is worth examining whether the research described herein met these three goals.

First, we suggest that the terminology and authoring process presented above addresses some of the difficulties presented initially. One difficulty noted was the increased complexity due to additional performance metrics to assess, since team tutors must cover both team skills and task skills. The framework described distinguishes these types of skills in the authoring process and uses a specific terminology to address assessment. Also, the framework carefully distinguishes between assessments of individual player actions, cumulative performance state, and final task performance. In addition, the case study illustrates that assessment for feedback triggers can be different than assessment for task performance. Another difficulty mentioned was the design of feedback. While the presented framework does not offer a direct answer to the best approach to feedback, the research outlines many of the feedback design decisions to be made, which supports team tutor authors by making their choices explicit.

Regarding Goals 2 and 3, one of the difficulties of developing team tutors is rooted in software architecture. The case study of the Surveillance Scenario Tutor described briefly the technical innovations that were made to GIFT so that it could support team

tutoring, e.g., the moving average approach within custom conditions and multiple simultaneous DKFs. Also, the scenario led to recommendations for an improved team architecture, such as an executive feedback module that could monitor the amount of feedback that each team member receives. GIFT is open source software, and this team architecture will be available to the general public after appropriate testing.

While the development of ITSs for teams is not entirely new, there remains much uncharted territory in the field. With technological advancements to the GIFT team tutoring software architecture, researchers can begin to explore different training tasks and variable team sizes. While GIFT was created as a tool for creating and implementing individual ITSs, it has been adapted to accommodate team tasks. Future authoring tools might look to the lessons learned documented in this article and provide ways to evaluate individuals and different combinations of team members without having to create separate tutors for each one. More specifically, of the 10 steps documented above, *Step 9 Encode conditions in the tutor* is the only one currently in which we work directly with the GIFT software. In the future, we envision working with GIFT in all 10 steps. GIFT would help the tutor creator analyze the team task, suggest individual skills and team skills and behavioral markers, recommend performance metrics based on those, list possible feedback messages based on different bodies of research theory, and derive conditions based on all of the above.

One of the most difficult components of GIFT to generalize across team tasks is a generic system for visualizing team interactions, e.g., an instructor dashboard for group after-action review, or real-monitoring tools that show how the team is doing right now vs. 30 min ago. Kopecky (2014) describes a tool called the Mixed-Reality Toolkit (MRT) that serves a related goal: monitoring multi-person training scenarios in which learners are interacting with multiple technology systems that require tight integration. However, the MRT was originally designed to integrate and monitor a federation of technologies rather than patterns of team behaviors within the scenario. Designing a generic team visualization system might also be inspired by the sports player tracking technologies developed to track athletes' movements across multiple sports, e.g., SportsVU or Second Spectrum.

A difficulty arises when considering the development of a team task for larger teams. The Surveillance Scenario assigned both team members the same roles with the same responsibilities, but it is unrealistic to expect larger teams to behave the same way. A second team tutor being developed by the authors includes three team members with role variation. Different role assignment also serves the purpose of constraining interactions between team members. If each person specializes in a particular aspect of a task, there is a set number of ways that the task can be accomplished. However, there is still much to learn about how teams function and how certain interdependencies between team members impact their interactions with each other. In a hierarchical team, where one member dictates what the other members do, there is a relatively straightforward relationship between the commander and his or her subordinates. A more egalitarian team structure, however, can add complexity to relationships and introduce constructs that require more research to understand. One consideration that must be taken into account is team orientation: how does the individual member perceive him or herself relative to the team? Another consideration is what Salas, Shuffler, Thayer, Bedwell, and Lazzara (2014) refer to as team cognition: is everyone on the same page with the same understanding of their roles and the task that needs to be accomplished?

This challenge of designing feedback for a team tutor to gain the attention of the learner appropriately without undue distraction is also a challenge in the design of human-robot team interactions. In future work, the feedback approach used with team ITSs could benefit from models developed in this growing research domain, e.g., Mortimer and Elliott's multimodal attention management approach (Mortimer and Elliott 2017) or the human-agent social systems framework by Lohani et al. (2017).

Goals for the immediate future include further operationalizing of team constructs, the creation of a multiple-role task for three team members, and the development of a flexible tutor in GIFT that can accommodate interactions between more than two people. Incremental progress is being made in designing ITSs for teams and with GIFT in the hopes of expanding the boundaries of what is currently possible with team tutoring.

Acknowledgments This work was supported by the U.S. Army Research Laboratory. The authors thank the anonymous reviewers for their careful analysis and suggestions, particularly the idea of a TKF.

References

Bangert-Drowns, R. L., Kulik, C. L. C., Kulik, J. A., & Morgan, M. (1991). The instructional effect of feedback on test-like events. *Review of Educational Research, 61*(2), 213–238.

Blessing, S. B., Gilbert, S. B., Ourada, S., & Ritter, S. (2009). Authoring model-tracing cognitive tutors. *International Journal for Artificial Intelligence in Education, 19*(2), 189–210.

Bonner, D., Gilbert, S., Dorneich, M.C., Burke, S., Walton, J., Ray, C., & Winer, E. (2015a). Taxonomy of teams, team tasks, and tutors. In *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)* (p. 189).

Bonner, D., Walton, J., Dorneich, M. C., Gilbert, S. B., Winer, E., & Sottilare, R. A. (2015b). The Development of a Testbed to Assess an Intelligent Tutoring System for Teams. In *Proceedings of the Workshops at AIED 2015*. Madrid.

Bonner, D., Gilbert, S., Dorneich, M. C., Winer, E., Sinatra, A. M., Slavina, A., MacAllister, A., & Holub, J. (2016a). The Challenges of Building Intelligent Tutoring Systems for Teams. Paper presented at the Human Factors & Ergonomics Society (HFES) Annual Meeting, Washington, D.C.

Bonner, D., Slavina, A., MacAllister, A., Holub, J., Gilbert, S., Sinatra, A. M., Dorneich, M., & Winer, E. (2016b). The hidden challenges of team tutor development. In R. Sottilare & S. Ososky (Eds.), *Proceedings of 4th Annual GIFT Users Symposium (GIFTSym4)* (pp. 49–60). U.S. Army Research Laboratory.

Buche, C., Querrec, R., De Loor, P., & Chevaillier, P. (2004). Mascaret: Pedagogical multi-agents system for virtual environment for training. *Journal of Distance Education Technologies, 2*(4), 41–61.

Burnham, B. R. (2010). Cognitive load modulates attentional capture by color singletons during effortful visual search. *Acta Psychologica, 135*(1), 50–58.

Cannon-Bowers, J. A., & Salas, E. (1997). A framework for developing team performance measures in training. In M. T. Brannick, E. Salas, & C. Prince (Eds.), *Team performance assessment and measurement: Theory, methods, and applications* (pp. 45–62). Hillsdale: Lawrence Erlbaum Associates.

Capuano, N., Marsella, M., Salerno, S. (2000). ABITS: An agent based intelligent tutoring system for distance learning. In *Proceedings of the International Workshop on Adaptive and Intelligent Web-Based Education Systems*, ITS.

Cooke, N. J., Salas, E., Cannon-Bowers, J. A., & Stout, R. J. (2000). Measuring team knowledge. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 42*(1), 151–173.

Corbett, A. T., & Anderson, J. R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*(4), 253–278.

Corbett, A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent tutoring systems. *Handbook of human-computer interaction, 5*, 849–874.

Crandall, B. W., & Hoffman, R. R. (2013). Cognitive task analysis. In J. D. Lee & A. Kirlik (Eds.), *The Oxford handbook of cognitive engineering* (pp. 229–239). New York: Oxford University Press.

Ellis, A. P., Bell, B. S., Ployhart, R. E., Hollenbeck, J. R., & Ilgen, D. R. (2005). An evaluation of generic teamwork skills training with action teams effects on cognitive and skill based outcomes. *Personnel Psychology, 58*(3), 641–672.

Fowlkes, J., Dwyer, D. J., Oser, R. L., & Salas, E. (1998). Event-based approach to training (EBAT). *The International Journal of Aviation Psychology, 8*(3), 209–221.

Franconeri, S. L., Hollingworth, A., & Simons, D. J. (2005). Do new object capture attention? *Psychological Science, 16*(4), 275–281.

Gilbert, S., Winer, E., Holub, J., Richardson, T., Dorneich, M., & Hoffman, M. (2015a). Characteristics of a multi-user tutoring architecture. In *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym3)* (p. 39).

Gilbert, S. B., Blessing, S. B., & Guo, E. (2015b). Authoring effective embedded tutors: An overview of the extensible program specific tutor (xPST) system. *International Journal of Artificial Intelligence in Education, 25*(3), 428–454.

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81–112.

Horstmann, G. (2002) Evidence for attentional capture by a surprising color singleton in visual search. *Psychological Science 13*(6), 499–505.

Hoffman, M., & Ragusa, C. (2015). Unwrapping GIFT: A primer on authoring tools for the Generalized Intelligent Framework for Tutoring. In *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)* (p. 11).

Hughes, A. M., Gregory, M. E., Joseph, D. L., Sonesh, S. C., Marlow, S. L., Lacerenza, C. N., Benishek, L. E., King, H. B., & Salas, E. (2016). Saving lives: A meta-analysis of team training in healthcare. *Journal of Applied Psychology, 101*(9), 1266–1304.

Jones, M. B. (1974). Regressing group on individual effectiveness. *Organizational Behavior and Human Performance, 11*(3), 426–451.

Jonker, C. M., van Riemsdijk, M. B., & Vermeulen, B. (2011). Shared mental models. In M. De Vos, N. Fornara, J. V. Pitt, & G. Vouros (Eds.), *Coordination, Organizations, Institutions, and Norms in Agent Systems V!. Lecture Notes in Computer Science* (Vol. 6541). Berlin: Springer.

Koedinger, K. R., Anderson, J.R., Hadley, W.H., & Mark, M.A. (1997). Intelligent tutoring goes to school in the big city. *International Journal for Artificial Intelligence in Education, 8*, 30–43.

Kopecky, K. E. (2014). A software framework for initializing, running, and maintaining mixed reality environments. Ph.D., Iowa State University, Ames, Iowa.

Kulik, J. A., & Kulik, C. L. C. (1988). Timing of feedback and verbal learning. *Review of Educational Research, 58*(1), 79–97.

Kumar, R., Ai, H., Beuth, J. L., & Rosé, C. P. (2010). Socially-capable conversational tutors can be effective in collaborative-learning situations. In *International Conference on Intelligent Tutoring Systems* (pp. 156–164). Springer Berlin Heidelberg.

Lam, C. F., DeRue, D. S., Karam, E. P., & Hollenbeck, J. R. (2011). The impact of feedback frequency on learning and task performance: Challenging the "more is better" assumption. *Organizational Behavior and Human Decision Processes, 116*(2), 217–228.

Lohani, M., Stokes, C., Dashan, N., McCoy, M., Bailey, C. A., & Rivers, S. E. (2017). A Framework for Human-Agent Social Systems: The Role of Non-Technical Factors in Operation Success. In *Advances in Human Factors in Robots and Unmanned Systems* (pp. 137-148). Springer.

Mortimer, B. J., & Elliott, L. R. (2017). *Information transfer within human robot teams: Multimodal attention management in human-robot interaction*. Paper presented at IEEE Cognitive and Computational Aspects of Situation Management (CogSIMA).

Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education, 10*, 98–129.

Murray, T., Blessing, S., & Ainsworth, S. (2003). Authoring tools for advanced technology learning environments: Toward cost-effective adaptive, interactive and intelligent educational software. Springer Science & Business Media.

Nadler, D. A. (1979). The effects of feedback on task group behavior: A review of the experimental research. *Organizational Behavior and Human Performance, 23*(3), 309–338.

Piramuthu, S. (2005). Knowledge-based web-enabled agents and intelligent tutoring systems. *IEEE Transactions on Education, 48*(4), 750–756.

Porter, C. O., Hollenbeck, J. R., Ilgen, D. R., Ellis, A. P., West, B. J., & Moon, H. (2003). Backing up behaviors in teams: The role of personality and legitimacy of need. *Journal of Applied Psychology, 88*(3), 391.

Prichard, J. S., Stratford, R. J., & Bizo, L. A. (2006). Team-skills training enhances collaborative learning. *Learning and Instruction, 16*(3), 256–265.

Rickel, J., & Johnson, W. L. (1998). STEVE (video session): A pedagogical agent for virtual reality. In *Proceedings of the second international conference on Autonomous agents* (pp. 332–333). ACM.

Rosen, M. A., Bedwell, W. L., Wildman, J. L., Fritzsche, B. A., Salas, E., & Burke, C. S. (2011). Managing adaptive performance in teams: Guiding principles and behavioral markers for measurement. *Human Resource Management Review, 21*(2), 107–122.

Salas, E., Shuffler, M. L., Thayer, A. L., Bedwell, W. L., & Lazzara, E. H. (2014). Understanding and improving teamwork in organizations: A scientifically based practical guide. *Human Resource Management, 54*(4), 599–622.

Salomon, G., & Globerson, T. (1987). Skill may not be enough: The role of mindfulness in learning and transfer. *International Journal of Educational Research, 11*(6), 623–637.

Schaafstal, A. M., Johnston, J. H., & Oser, R. L. (2001). Training teams for emergency management. *Computers in Human Behavior, 17*(5–6), 615–626.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research, 78*(1), 153–189.

Shute, V. J., & Psotka, J. (1994). Intelligent tutoring systems: Past, present and future. In D. Jonassen (Ed.), *Handbook of Research on Educational Communications and Technology*. Scholastic Publications.

Singley, M. K., Fairweather, P. G., & Swerling, S. (1999). Team tutoring systems: Reifying roles in problem solving. In *Proceedings of the Computer Support for Collaborative Learning* (p. 66). International Society of the Learning Sciences.

Sottilare, R. A. (2014). Examining opportunities to reduce the time and skill for authoring adaptive intelligent tutoring systems. In *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)* (pp. 3-10).

Sottilare, R. A., Brawner, K. W., Goldberg, B. S., & Holden, H. K. (2012). The generalized intelligent framework for tutoring (GIFT). *Orlando: US Army Research Laboratory-Human Research & Engineering Directorate (ARL-HRED)*.

Sottilare, R., Holden, H. K., Brawner, K. W., & Goldberg, B. S. (2011). Challenges and emerging concepts in the development of adaptive, computer-based tutoring systems for team training. Paper presented at the The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC), Orlando, FL.

Sottilare, R. A., Ragusa, C., Hoffman, M., & Goldberg, B. (2013). Characterizing an adaptive tutoring learning effect chain for individual and team tutoring. In *Proceedings of the Interservice/Industry Training Simulation & Education Conference*. Orlando.

Sottilare, R. A., Burke, C. S., Salas, E., Sinatra, A. M., Johnston, J., & Gilbert, S. B. (2017). Designing Adaptive Instruction for Teams: A Meta-Analysis. *International Journal for Artificial Intelligence in Education*. https://doi.org/10.1007/s40593-017-0146-z.

Tellinghuisen, D. J., & Nowak, E. J. (2003). The inability to ignore auditory distractors as a function of visual task perceptual load. *Perception & Psychophysics, 65*(5), 817–828.

Traum, D., Rickel, J., Gratch, J., & Marsella, S. (2003). Negotiation over tasks in hybrid human-agent teams for simulation-based training. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems (pp. 441-448)*. ACM.

Von Mühlenen, A., Rempel, M. I., & Enns, J. T. (2005). Unique temporal change is the key to attentional capture. *Psychological Science, 16*(12), 979–986.

Walton, J., Dorneich, M. C., Gilbert, S., Bonner, D., Winer, E., & Ray, C. (2014). Modality and timing of team feedback: Implications for GIFT. In *Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2)* (pp. 199–207).

Walton, J., Bonner, D., Walker, K., Mater, S., Dorneich, M., Gilbert, S., West, R. (2015a). *The Team Multiple Errands Test: A Platform to Evaluate Distributed Teams*. Paper presented at the Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing.

Walton, J., Gilbert, S., Winer, E., Dorneich, M., & Bonner, D. (2015b). *Evaluating Distributed Teams with the Team Multiple Errands Test*. Paper presented at the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC). Orlando.

Whatley, J. (2004). An agent system to support student teams working online. *Journal of Information Technology Education, 3*, 53–63.

Yin, J., Miller, M. S., Loerger, T. R., Yen, J., & Volz, R. A. (2000). A knowledge-based approach for designing intelligent team training systems. In *Proceedings of the fourth international conference on Autonomous agents* (pp. 427–434). ACM.

Zachary, W., Cannon-Bowers, J., Bilazarian, P., Krecker, D., Lardieri, P., & Burns, J. (1999). The advanced embedded training system (AETS): An intelligent embedded tutoring system for tactical team training. *International Journal of Artificial Intelligence in Education, 10*, 257–277.