2018

Lessons Learned Creating a Team Tutoring Architecture: Design Reconsiderations

Keith Brawner
United States Army Research Laboratory

Anne M. Sinatra
United States Army Research Laboratory

Stephen B. Gilbert
Iowa State University, gilbert@iastate.edu

Lessons Learned Creating a Team Tutoring Architecture: Design Reconsiderations

Abstract

Intelligent Tutoring Systems have been constructed for many reasons by numerous groups of researchers and practitioners. The Generalized Intelligent Framework for Tutoring (GIFT) was created in part to unify disparate research into a commonly shared research output. Unlike individual tutoring, however, few groups have constructed an Intelligent Team Tutoring System (ITTS) (Robert A Sottilare, 2018; R.A. Sottilare, Holden, Brawner, & Goldberg, 2011). While GIFT was created in order to address the problems of individual tutoring systems, small group (i.e., squad level) team tutoring is one of its goals, and it was modified to support the instruction of teams of individuals. These modifications were consistent with the original GIFT design goals for individual tutoring systems (individual assessment/feedback files), but this ultimately proved cumbersome for the construction of a reusable set of modules and processes for teams. This paper reports on the experience of constructing an ITTS software architecture using GIFT, but expands on the challenges encountered during the design process, and what could be done differently during redesign. Perhaps more important than discussion of the original team tutoring design or an improved team tutoring design is a discussion of the reasoning behind using certain principles and implementations. As among the first to try to construct a reusable team tutoring structure, the lessons learned should be considered as a basis for future implementations.

Disciplines

Ergonomics | Human Factors Psychology | Operational Research

Comments

This chapter is published as Brawner, Keith, Anne M. Sinatra, and Stephen M. Gilbert. "Lessons Learned Creating a Team Tutoring Architecture: Design Reconsiderations." In *Design Recommendations for Intelligent Tutoring Systems: Volume 6 - Team Learning and Taskwork* (A.C. Graesser, X. Hu, A.M. Sinatra, and R.A. Sottilare, eds.). Orlando, FL: U.S. Army Research Laboratory, 2018, pages 201-215.

CHAPTER 20 – LESSONS LEARNED CREATING A TEAM TUTORING ARCHITECTURE: DESIGN RECONSIDERATIONS

Keith Brawner[1], Anne M. Sinatra[1], Stephen Gilbert[2]
[1]Army Research Laboratory, [2]Iowa State University

Introduction

Intelligent Tutoring Systems have been constructed for many reasons by numerous groups of researchers and practitioners. The Generalized Intelligent Framework for Tutoring (GIFT) was created in part to unify disparate research into a commonly shared research output. Unlike individual tutoring, however, few groups have constructed an Intelligent Team Tutoring System (ITTS) (Robert A Sottilare, 2018; R.A. Sottilare, Holden, Brawner, & Goldberg, 2011). While GIFT was created in order to address the problems of individual tutoring systems, small group (i.e., squad level) team tutoring is one of its goals, and it was modified to support the instruction of teams of individuals. These modifications were consistent with the original GIFT design goals for individual tutoring systems (individual assessment/feedback files), but this ultimately proved cumbersome for the construction of a reusable set of modules and processes for teams. This paper reports on the experience of constructing an ITTS software architecture using GIFT, but expands on the challenges encountered during the design process, and what could be done differently during redesign. Perhaps more important than discussion of the original team tutoring design or an improved team tutoring design is a discussion of the reasoning behind using certain principles and implementations. As among the first to try to construct a reusable team tutoring structure, the lessons learned should be considered as a basis for future implementations.

Types of Team Tutoring

Team tutoring is a complicated process, and requires updating existing ITS software architecture to support a team (Gilbert, Dorneich, Walton & Winer, 2018). However, the updating of the architecture is more than just duplicating what has been done for an individual multiple times. Careful consideration needs to be given to how the team will be tutored, how the team's performance will be assessed, and the type of task that the team will be tutored on. In the case of GIFT, since it is a domain-independent ITS architecture, updating it to support teams is particularly difficult. The framework needs to be able to support multiple types of domains/content areas, multiple configurations of teams, as well as multiple types of team performance assessments. In addition to this, GIFT must be able to support tutors that are focused on teamwork, taskwork, and collaborative learning.

Further, team tutoring must support different team task configurations, and the different team pedagogies that might be used. Teamwork and taskwork tutoring differ from each other, as teamwork focuses on the general overarching principles of working together with a team, and taskwork is specific to the team task that is being engaged (Salas, 2015). In order to engage in teamwork tutoring, there should be a representation of generalized positive team behaviors that need to occur to reach a goal, as well as methods of measuring those desired team behaviors. To engage in taskwork training, however, the focus is more on the actual performance of the task itself and the role that each of those team members play in it; performance on the task is usually more easily measured. The structure and roles within the team are going to vary greatly based on what the overall task is that is being tutored. Additionally, some team tutoring tasks including collaborative problem solving or collaborative learning may have a different learner configurations. Collaborative learning theory (Roschelle, Suthers, & Grover, 2014) characterizes teams as a community in which some team members might be core members, while other members are more peripheral. Thus, a key element of monitoring team dynamics is measuring the degree to which each team member is involved in

the community, e.g., team members' motivation for collaborating, the level of joint attention to the task, and the degree to which the team members reflect on the teamwork process. From the point of view of ITTS architecture, this theory creates a software requirement that the ITTS architecture support monitoring of each team member individually, as well as each member's interactions with teammates.

GIFT

GIFT is made up of a series of interchangeable modules which are linked together by a message bus (Robert A Sottilare, Brawner, Sinatra, & Johnston, 2017). The core Modules of GIFT are the Domain, Learner, Pedagogical, Gateway, and Sensor. However, GIFT is an architecture, and on top of this module foundation is built a number of less standardized components – items such as authoring tools (i.e., the GIFT Authoring Tool), dashboards (i.e., the learner panel), and post-processing tools (i.e., the Event Reporting Tool). This ecosystem also includes less commonly used, or more commodity-manufactured, components such as a user database (i.e., User Management System), content repository (i.e., Nuxeo), and software libraries for external applications (i.e., XML Remote Procedure Calls).

GIFT is relatively simple, containing a few key modules, which now seek standardization through the IEEE Adaptive Instructional Systems group. Each of the core modules relates to the learner, e.g., a model of the learner's state, content to teach the learner, or expert information needed to evaluate the learner. It is tempting to design a "Team GIFT" to be the same thing – a model of the team's state, content to teach the team, and expert information needed to evaluate the team. As GIFT was built for an individual learner and then built out as a collection of differing tools in order to support the learner – the hope is that many of the tools can be re-used and not reinvented for team tutoring purposes. However, as has been reported in the literature, there are challenges involved with scaling up intelligent tutoring systems for teams (Bonner, Gilbert, et al., 2016; Bonner, Slavina, et al., 2016; S. Gilbert et al., 2015; S. B. Gilbert et al., 2017). Since a team's performance depends strongly on the team skills of the members', not only on the individuals' task skills, the complexity of the ITTS's model of the both the learners and the state of the training scenario grows exponentially. Additionally, there is the challenge of keeping GIFT domain-independent, therefore, any tools that are developed for authoring in team tutoring in GIFT need to support multiple types of team configurations.

Team GIFT

An initial team surveillance task tutor was developed using GIFT, and has been detailed in the literature (S. B. Gilbert et al., 2017). One of the main goals in the development of the tutor was demonstrating that GIFT could be adapted for use by multiple players in a shared environment. The first version of the tutor consisted of two players working together to monitor their own portions of an environment in a scenario in the computer-based software Virtual Battlespace 2. Challenges that needed to be overcome were how to assess the team members as individuals and as a team, and how to have GIFT assess their performance as well as provide feedback simultaneously. The goals of this initial tutor creation included creating a shared terminology; creating a militarily-relevant team task; and establishing an approach that can facilitate the development of team tutors. The challenges were overcome primarily by adapting the existing feedback and assessment structure (Domain Knowledge Files) to be able to track performance by both individuals and by a team. Primarily this was done by having a separate DKF for each individual player, and an additional DKF that assessed the team tasks and could provide feedback to both players if needed. The synchronization challenge was solved in this particular instance by utilizing a "lobby" in which both players checked in prior to the beginning of the tutoring session, as is used in many multiplayer commercial games. This lobby method is generalizable, as shown by the multitudes of commercial games, but the approach may have to differ based on the training applications in use with GIFT.

The initial team surveillance task was used to create two tutors. The first of these was the Surveillance Tutor, which had two team members with the same role. The second tutor was called "Surveillance Tutor with Sniper," which was a three player version which involved creating an additional DKF file for the new team role and their interactions with other team members. These two tutors represent a first attempt to engage in team tutoring in the GIFT framework. They can be thought of "successful" as they were operational enough to accomplish the task of team tutoring while being designed flexible enough to capture new tasks, domains, and team structures. However, many of the approaches that were used in the creation of the two tutors were tightly coupled with the task that existed, and the addition of tasks lacked in both generalizability and scalability. The simple addition of a new team member involved the addition of a role, its assessment, its communication, and the assessment of the communication; further additions would face exponential complexity. While the initial two ITTS met their goals, but it is now time to consider how the approach could be improved; are there other approaches that could achieve the same goals but have additional scalability?

First Pass Design and Walkthrough

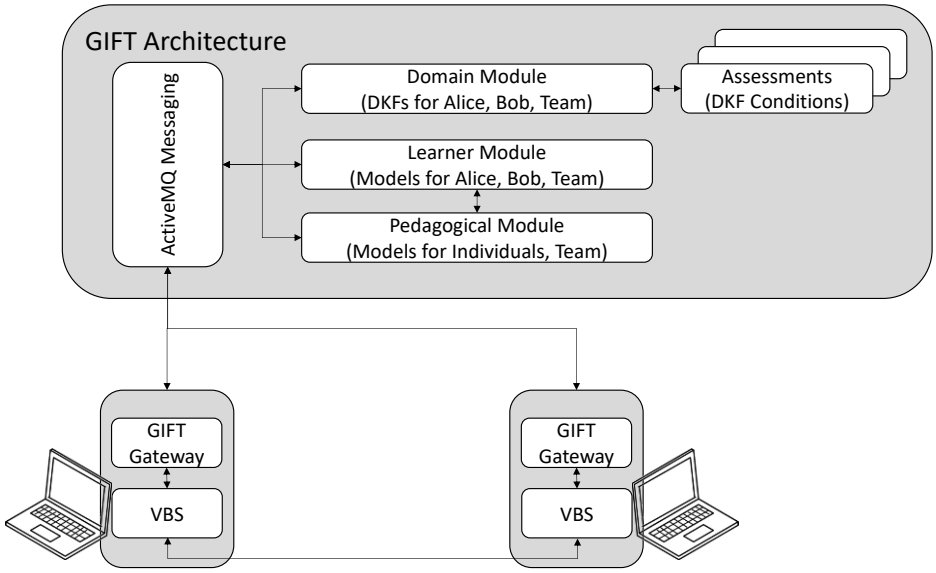

Figure 3. Initial Team Tutoring System Design (S. B. Gilbert et al., 2017)

As part of the initial GIFT ITTS approach, the team decided that the team tutoring system and architecture would be built on top of the existing GIFT structure. The tutoring for a team consisted of the tutoring for each team member in the same manner as an individual tutor (based on an individual DKF assessment), and that the team members would additionally receive tutoring as part of team tutoring components (based on a team DKF assessment). See Figure 1 for a visual representation of the approach used in the initial GIFT team tutor. This decision to treat "the team" in a similar way that GIFT would treat an individual had ramifications for the core GIFT modules, but also for the less standardized components and for the commonly-used components. The changes to individuals' modules in order to support this decision are discussed next, on a module basis. For an individual application, a single Gateway Module was needed to link the external simulation to the rest of GIFT. For an individual learner, the Gateway Module translated simulation messages into GIFT messages and sent them to the Domain Module for assessment. For a team of learners, the Gateway Module remains unchanged, except that messages for each player were appropriately routed to the individual Domain Modules for the individual learners, and messages regarding each player should also be copied to a Team Domain Module (or, more accurately, the existing Domain Module

with a team configuration). As mentioned above, the need to have multiple players within the same simulation also imposed the requirement to have a shared start time, a "game lobby" functionality and other general synchronization logic.

As noted above, for an individual learner, the Domain Module has a configuration called a Domain Knowledge File (DKF). The DKF contains the information about how the learner will be assessed in the external application, and the domain specific feedback that they are to receive; it has links to content, assessments, and feedback which are controlled by pedagogical direction. For a team of learners, the GIFT architecture was tweaked to allow the existence of multiple simultaneous DKFs. Thus, each learner had an individual DFK, and a team DKF was added containing similar information. The team DKF processed *team* performance, *team* assessment, and *team* feedback (for both teamwork and taskwork). For a team with three team members, one can imagine six DKFs: one for each team member, one for the team of three, and one for each pair of team members so that pair communications can be monitored. This architecture served the purpose of monitoring both individual and team performance, but it leads to the challenge that each DKF functions like an independent tutor that does not know the actions of its tutoring colleagues. See below for more detail on the implications.

For an individual learner, the Learner Module gathered, stored, and reported information about the learner. In individuals, these attributes are items such as personality type, learning preference, as well as information about performance, learning, and affect. Following the logic described above, to extend this for teams, the Team Learner Module (Learning Module with a "team" configuration coming from the Domain Module), would gather, store, and report information about the team. Relevant team specific learner module components would include team performance and team-level assessments of state, such as measure of communication or trust. However, in the two surveillance ITTSs, this team Learner Module did not add any specific value, as no domain-general team items were programmed to be observed. This design decision required results in the Domain Module to drive team instructional decisions. The result of this is that it is not easy to reuse this approach between domains, as many assessments rely on the content of the messages that team members communicate to each other. In theory, however, the configuration of team assessments, such as frequency of communication acts, may be transferrable between domains.

For an individual learner, the Pedagogical Module controls the instruction, including actions such as ordering content, requesting assessment, changing scenario difficulty, providing hints and feedback, and other instructional actions. Following the logic above, for a team, a team pedagogical module would need to make the important decision of when to give feedback to specific team members and when to give feedback to the entire team. In the surveillance ITTSs, the normal pedagogical module was fed information tagged with "team" simply passed the feedback onwards. Future implementations should allow for team-specific models, modeling of differing team members, and other pedagogically-specific items.

Also, a key requirement for tutoring teams is monitoring the overall amount of feedback received by team members. With the multiple-DKF architecture mentioned above, it is easy for learners to be inundated with too much feedback with messages arriving from their individual DKF, the team DKF, and any sub-team DKFs that might exist. As a specific example, a learner who does not report an observation of an enemy ship in a timely manner may be graded poorly on their individual observation performance, graded poorly as a team on a communication metric, and graded poorly as a team for reporting observations – resulting in three pieces of feedback individually and two pieces of feedback for the team. A better team tutoring system would handle individual and team feedback differently, such as providing individual task feedback during the event and saving team feedback for an After Action Review (AAR). Ideally the team pedagogical module would filter feedback messages from the multiple DKFs, correctly model a chain of command, or other more complex items.

In the surveillance tutors, an additional Feedback Filter component was created. Normally, GIFT offers feedback to the learner after every action that changes the learner's overall assessment state (Above, At, or Below Expectation). In tasks in which every learner action is noteworthy pedagogically, e.g., where learner actions consist of answering questions or completing tasks, this approach makes sense. But if the granularity of the learners' tasks is smaller, such that the overall pattern of learner actions is of pedagogical interest, but not the individual actions, feedback need not be given for each action. The surveillance tutors fit the latter category, with learners performing many repetitive small actions. Thus, the Feedback Filter allowed only every *nth* feedback message to reach the learners, where *n* depended on the subtask.

Design Failures

One of the design goals of GIFT is to provide a pedagogically sound approach to authoring a tutor that does not require the subject matter expert to understand the elements that make up a tutoring system. The tutor author brings his or her content to the system, and the system is primarily responsible for configuring the tutor. A part of this design trade-off is that the *individual* components of the tutoring system can be *more* complex, while reducing complexity overall. As a specific example, consider the creation of a domain-dependent model of content applicable to multiple new and unknown application tasks (e.g., modeling how to dress wounds in multiple application contexts such as in field, in the home, and in the workplace); this task is more complex than modeling the domain specifically for one application (e.g., how to dress wounds just at home), but did not require the additional creation of learner modeling techniques, instructional techniques, feedback delivery mechanisms, and other tutoring components. Although the individual model of the domain was more complex, the overall complexity required to make a tutoring system decreased. This approach, while adding work for an individual component (the wound dressing model), gives an overall reduction to system authoring cost as applications are expanded.

The architecture of the Feedback Filter, while a good first attempt, presented several problems. First, it was based on the assumption that feedback about individual performance and team performance were of equal weight. In reality, the nature of the task and coaching pedagogy would determine this weighting. For a small team with tightly interdependent tasks, the individual feedback may be critical for the overall team. For a team of people who act more independently, but share a common overall purpose, individual feedback may play a stronger role. Also, the approach of simply "passing along feedback" belies the careful strategic approach to giving feedback taken by a professional team coach. For the individual, every feedback statement is a criticism, even if taken constructively. While sometimes the bitter pill of receiving feedback can be mollified by a framing of "It's for the good of the team," that approach can be used only a limited number of times. Thus, team feedback statements and individual feedback statements are not equivalent in impact on the learner, and should not be counted as such. Finally, different learners have different tolerances for receiving feedback. Ideally, this filter would be personalized for the team member based on a profile of whether the learner benefits from more or less feedback (ex. frequent feedback for novices, infrequent feedback for experts)..

This complexity is compounded through the interactions of the various components. The original GIFT architecture called for a single sequential round trip from the learner to the training environment to the Gateway Module to the Domain Module to the Learner Module to the Pedagogical Module and then back to the Domain Module, to the Gateway Module, to the training environment and to the learner. This represents a single loop of human-system interactions. The above design calls for a *parallel* linear interaction (adding “Team” to each Module), which is tied back into the training environment for *multiple* learners. The human-system interaction is then a product of *two* loops that interact: one for individual tutoring and one for team tutoring. This makes testing, debugging, and tracking activities more complex, even without considering the challenges of time-delayed feedback, the construction of after-action review reports, or real-time feedback.

One approach to handling this is to create a broader, more generalized version of the Feedback Filter that one might call the Learning Experience (LX) Module (Cushard, 2012). Just as companies seek to optimize the customer experience they offer, the LX Module would maintain awareness of each team member's experience in the simulation: their initial learning profiles, how successful they are, their cognitive load, the amount of feedback they've received, how their performance is changing, how much impact they have on other team members, etc. – all the factors that a quality professional coach takes into consideration. This "executive" module of GIFT would be based on LX models of different team tasks and different team structures, so that it could be re-usable from team to team and task to task of the same genre.

GIFT Design Principles

GIFT was initially designed to solve complex but definitive problems. The first of these is the problem of lack of standardization, causing increased initial construction cost of the ITS. The second is the difficulty of modifying an existing system to suit new purposes. These problems were addressed through the separation of logic and the creation of a common infrastructure. The goals for team tutoring *include* the goals for individual tutoring with the added goal of being able to provide tutoring to multiple individual team members and to the team as a whole.

Separation of Logic

Meeting the two goals discussed above starts with the separation of logic. Separation of logic, or dividing the system architecture into maximally separate independent modules, so that each could be updated separately for new training situations, is the primary design goal. This approach accommodates shared authoring tools, interchangeable instructional domains, keeping experimental control for instructional modifications, and other items. Separating the logic into modules additionally allows for the topic area (domain) of a tutor to be changed to another topic (domain) area with relatively minor edits. Similarly, it allows for the improvement of pedagogy across *all* topic areas through the replacement of a pedagogical module with an improved pedagogical module without further change.

For a team, this issue matters – team assessment is not likely to be held constant across multiple team structures. A good basketball team and a good aircraft maintenance team have both a) different jobs (score points, fix planes), b) different measures of what good taskwork is (pass ball and create openings, delegate out work), and c) different models of what good teamwork is (communicate frequently, leave people alone to work). It seems possible that a model of team instruction may not be held constant over multiple domains, though it could be that, if well designed, a relatively small set of team models could satisfy many domains (e.g., the Pareto 80/20 principal).

Another approach to consider is a parent-child inheritance hierarchy within modules. For example, some pedagogical principles, like "If the learner is getting frustrated, try to offer him/her some small successes," might generalize across domains. Other pedagogical principles might be specific to the domain. For example, if learning to write complex Excel formulas with multiple nested IF conditions, a principle might be, "If the learner can't debug the complex Excel formula, suggest breaking it up across multiple spreadsheet cells and building it up piece by piece." This more specific pedagogical principle could be considered to be a child class instance of the parent class principle of offering small successes. Thus, a new ITS or ITTS could be authored with a generic template of parent class principles, saving significant time, and then add more specific child instances as necessary. This same hierarchical inheritance approach could apply to domain models and feedback instances as well.

Figure 4 illustrates an example of an ITTS architecture that would use this inheritance approach to guide members of a medical team: a doctor and two nurses. The tutor contains domain models for general medical

task work, along with more specific models of doctor knowledge and nurse knowledge. Also, it contains a domain model for teamwork generally, and a more specific instance of that model focused on teamwork on medical teams. The domain models would like generate many possible feedback messages for any given configuration of world states and learner actions, and pass those messages to Pedagogical Prioritization for winnowing. The Pedagogical Prioritization module contains a model of the most general pedagogy, as well as more specific models for team pedagogy, and even more specifically, medical team pedagogy. Also, it contains pedagogical techniques designed specifically for doctors and nurses. This module would prioritize feedback messages based on pedagogical guidelines and pass the surviving options to the Feedback Filter or LX Module, which contains models of each of the learners, as well as a team model and sub-team models. This overall module is responsible for making sure the learners are not overwhelmed by feedback, and would maintain a history of their learning experiences so far so that it can optimize each person's learning. This module finally chooses among the multiple feedback messages passed to it and sends them along to the learners.

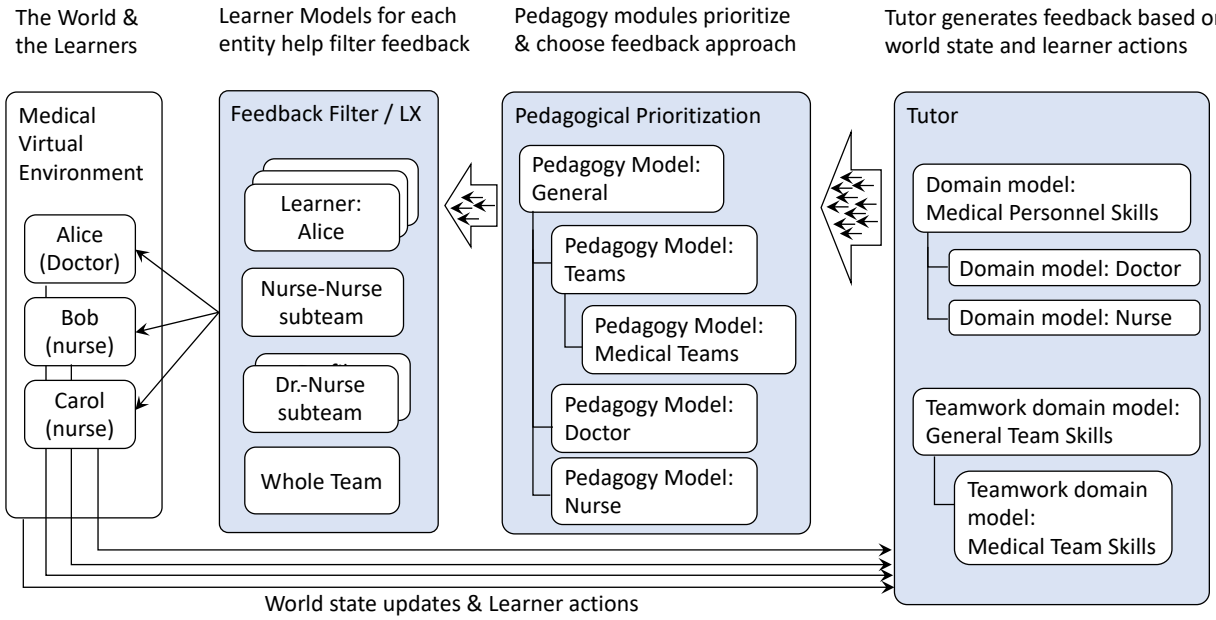

Figure 4: Example of an ITTS architecture featuring hierarchical inheritance models. A doctor and two nurses are the team.

Common infrastructure and Pipeline

In the same manner as GIFT for individual tutoring is built upon interchangeable modules for the purpose of both production and experimentation, a team tutoring architecture should mirror this functionality. Achieving the first goal of logic and process separation should allow for the achieving of a second goal of a common infrastructure. Advantages of construction in this manner is the ability to share a common set of authoring tools, a data processing pipeline, and standardization of requirements for construction. While these are not explicit architectural principles, they are a desired product of the other principles.

Individual *and* Team Tutoring

For individuals, tutoring is synonymous with individualized feedback and path planning through content. For teams, however, teaming items are divided among both taskwork and teamwork. Taskwork is the tasks

that have to be accomplished (it is domain dependent). Teamwork is how the team is performing with respect to itself (it is mostly domain independent). These items are somewhat separate, as a team can be internally consistent and work well together but still fail to accomplish the task. As a concrete example, consider a losing World Cup team; the team is clearly very skilled and works well together, but in this specific instance did not accomplish their task. Similarly, it is possible to have poor team performance but accomplish mission or task objectives. As a concrete example, consider a middle school soccer team with a single exceptional athlete. The team may perform poorly as a team, but still win because of the individual athlete's performance. Tutoring a team to perform well (taskwork) and to be a functional unit (teamwork) are two *separate* items, but a flexible domain-independent team tutoring architecture should strive to accommodate both goals.

Team Design Examples – Training Different Types of Teams

The structure of teams can vary significantly. Teams might have members with different roles, the same role, or a mix. Teams may have a central hierarchical leader, shared leadership, or no explicit leader. Team members may be co-located or distributed. These are just a few of the different characteristics of team structure described in (Bonner et al., 2015). Also, team tasks themselves can vary significantly in interdependence across team members, independent subtasks performance, and other criteria. In the following section we describe two different team tasks with different configurations of roles.

These two examples highlight the differences that can exist in the roles, responsibilities and tasks of team members in different domains. A generalized ITTS framework needs to be flexible enough to allow for the authoring of tutors in both of these areas as well as other needed configurations.

Anti-Submarine Warfare (ASW) Helicopter Team

Anti-submarine warfare (ASW) helicopter team members have distinct roles with separate tasks. Physically, using active sonar reveals the position of the sonar system to those within range. An alternative to the use of surface ship mounted active sonar, which effectively announces ship position to enemies, is to use helicopter-mounted "dipping" active sonar. As it is technically difficult to hit a helicopter with an underwater-launched weapon without revealing the submarine's position, this approach to active sonar is preferred in many contexts. The goal of an ASW Helicopter team is to find hidden submarines from the (relative) safety of an airborne platform. The ASW Helicopter team has three key members – a pilot, a copilot, and a sonarman, with the primary tasks of flying the helicopter, coordinating intelligence with the host ship, and operating the sonar to find submarines, respectively. Theoretically there is no overlap between the tasks in each of these roles, while cross-training is common in practice.

Combat Outpost Guard Team

A combat outpost guard team has a very different configuration than an ASW helicopter team. The primary goal of a combat outpost is to extend the depth of the security area, or provide safety to forward observing positions who may be encircled by enemy forces. There can be many members of a stationed combat outpost guard, but a minimum of two posted guards is common. The mission of the posted guards is to observe and report on relevant activities they observe outside the secure area until ordered to withdraw, advance, or some other order change. Functionally, each member of the team has the same mission during this time period.

Team Architecture Design Goals

Design Goals for Reuse

GIFT, as originally designed for individuals, can be reused to provide tutoring for teams, provided that the tutoring system adequately represents each team member's task role. As a design approach, this means that the individual tutoring for the combat outpost guard team can be identical without any significant change – the system can apply the assessments written for the individual for each team member. Reuse of individual tutoring is wholesale. Similarly, changing the individual tutoring from tutoring two individuals to tutoring ten individuals playing similar roles involves little, if any, changes. Similarly, the performance of the team can be a more simplistic roll-up of individual performance metrics; the security of the outpost is mostly a summation of the security at each point. The assessment configuration of teamwork performance (e.g., communication, coordination, etc. (R. Sottilare et al., 2017)), however, may be able to be reused from another domain which is heavily communication-oriented.

The ASW team, however, has different roles. Each role has very different responsibilities – piloting aircraft; tactics and procedure; operation of underwater acoustic sensors. The individual tutoring that each of these learners receives is significantly different; there is little opportunity for reuse of domain content. GIFT, however, may reuse all of the other components of tutoring – Learner Module, Pedagogical Module, Gateway, etc. The team performance metrics cannot be a simplistic roll-up of individual performances; there is not an additive relationship between 20 seconds of smooth hovering and spotting an enemy submarine. Alternatively, the performance of the team, or teamwork measures, such as communication and coordination (Sottilare, 2017), may be able to be reused from a training environment similar to the above example guard team task.

The intention behind a team tutoring framework is that, at a minimum, the teamwork tutoring component can be reused across instructional domains. Further, that the generated team tutor can be expanded to accommodate additional team members and roles easily. Additionally, similar to how GIFT does not change the Learner/Pedagogical/etc. Modules in order to instruct a new domain, the team tutoring system should be able to perform these actions on the team level – keep a history of the team, have a consistent instructional model, etc. This design goal for a team tutoring architecture to support the training of many teams has two derived requirements: for the team component to be agnostic to both the domain of interest and the structure of the team.

Agnostic to the Domain

Firstly, the structure to support team modeling and instruction must be agnostic to the domain of instruction. If correctly designed, this is a direct product of modularity. The ability to shift a team tutor from one type of team and task to another is critical to the creation of a team tutor; what good is a team tutoring architecture if it can only tutor ASW Helo teams? One way to think about team taskwork is to think of it as domain-dependent teamwork.

Agnostic to Team Structure

It is possible that different structures of teams should be instructed differently. The authors, however, hypothesize that there are basic models of instruction or principles of teamwork that are applicable regardless of team structure. The hope for such models is that they can be applied as baseline models for many types of teams, assuring that all teams get at least some instruction. An example of a simplistic model of

team instruction, similar to the initial simplistic models for individual tutoring GIFT, is “feedback on underperformance given immediately upon observation.” Another simplistic model of team pedagogy is a “speak the same language” model which monitors for vocabulary overlap and gives corrective feedback when vocabulary overlap is not observed. Both of these models are applicable regardless of team size and structure, albeit simplistic. More complex models can be constructed with similar data, as enabled by architectural design.

Design Tradeoffs Considered for Taskwork and Teamwork

Design Tradeoff Evaluation

The design tradeoffs will be considered with their ability to provide for the following list of desired features, discussed above:

1. Separation of logic among modules
2. Ability to provide common infrastructure, such as for authoring tools and database structure
3. Ability to tutor individually
4. Ability to tutor team taskwork
5. Ability to tutor teamwork
6. Ability to accommodate differing team structures
7. Ability to accommodate differing team assignments
8. Ability to be authored

Option 1 – Do Nothing: The Crowd of Tutors.

GIFT is capable of team tutoring by default through the use of multiple simultaneous domain models: one for each player, one for the team, and others for subteams, if necessary. This represents the failures of the first design, discussed earlier. This approach provides good scores on the separation of logic and common infrastructure. Upon use of this design, the team is instructed as though it is an individual learner – with content mapped to its joint traits, scenarios assigned in an escalating difficulty mastery-learning paradigm, and feedback given to all team members. A direct result is that no individualized feedback is available from the system (except on individual tasks created in the individual tutors), but that team taskwork performance is the primary means of adaptation. No team-specific modeling pedagogy is assigned, although authoring considerations could provide teamwork items as authored domain-dependently within the Domain Module. The structure does accommodate for differing team structures and roles. One of the primary disadvantages of this approach is that each of the learner components acts essentially as a separate, independent tutor. Thus, each team member on a three-person team might get feedback from his or her individual tutor and the team tutor, without those two tutors knowing what each other's feedback was. If the three-person task involved important pair-wise relationships, then it might require a learner module for each individual (A, B, and C), a learner module for each pair of individuals (A-B, B-C, A-C) and a learner module for the team (A-B-C). This "crowd" of six independent tutors could lead to significant feedback overload and even mixed messages that conflict in their direction. However, much like Selfridge's classic Pandemonium model of perception (Selfridge, 1958), in which we recognize objects by having a crowd of neural "demons" each simultaneously shouting about the features they see, and listening to the demons that shout loudest, a Crowd of Tutors approach could have a filter that quiets all but the "loudest" tutor feedback.

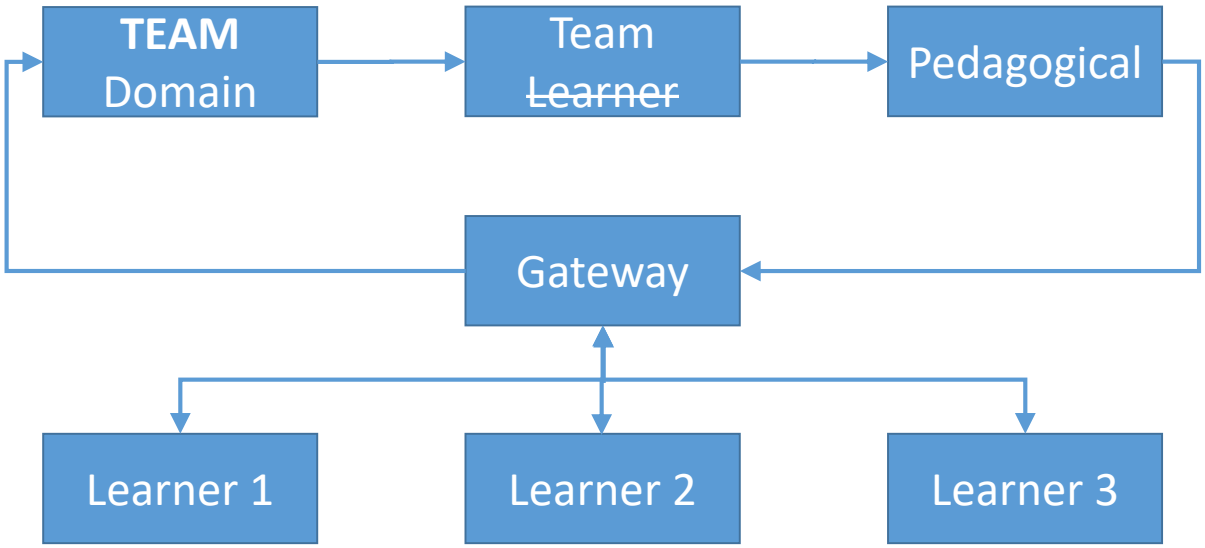

Figure 5. Do Nothing option for ITTS design

Option 2 – Individual Tutors for Individuals, Information Reported to Single Team Tutor Information (Initial Design).

The drawbacks of this type of design have been more thoroughly discussed in the earlier sections. Suffice to say that the design allows for individual interchangeability and significant flexibility, but that it does so at the cost of the maintenance of many modules. While the system is *capable* of tutoring the individual team members on taskwork and teamwork, the creation of the logic to assess, give feedback, and post-process imposes significant burden on the author. The author must answer the questions of what each team member should be performing with respect to his or her individual task, with respect to the overall team task (taskwork), and with respect to being a member of the team (teamwork). This is more appropriate for teams such as the ASW team where roles are sufficiently different with little taskwork overlap.

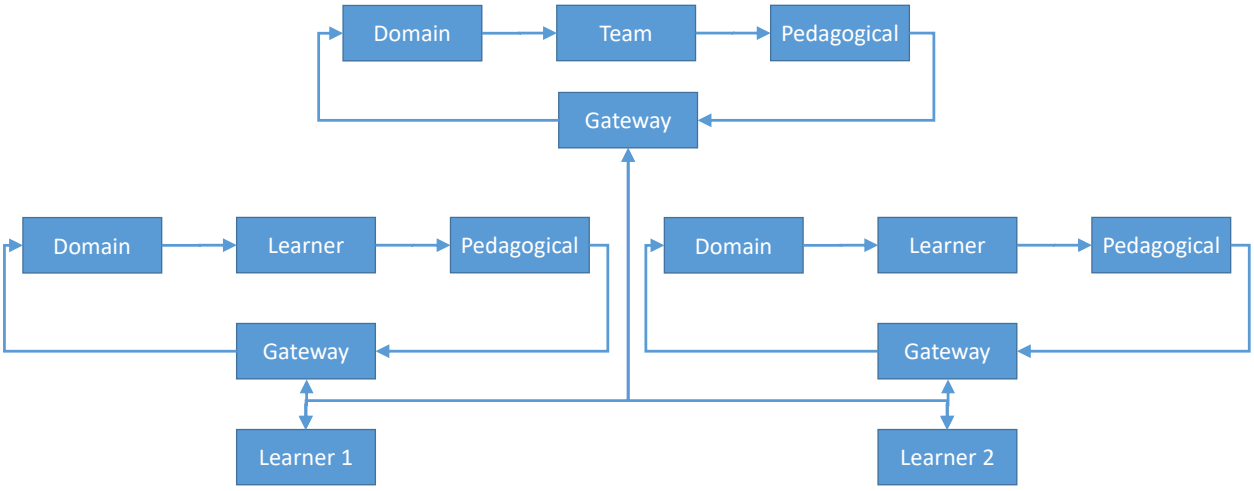

Figure 6. Initial Design for ITTS

Option 3 – "Team" Modules for Team Tasks

In an effort to moderate the complications of authoring team and individual tasks, and to standardize team taskwork and teamwork items, a structure of "team" Domain items, team modeling and team pedagogy can be considered. This mitigates many of the problems faced in the initial design. Further, it allows the common tools for team items without being particularly concerned about the individual domains. Authoring tools may be able to be templated for authoring team items, asking streamlined questions for individual domains such as "how often should team members communicate?" and "what should elements of a shared language or mental model be?".

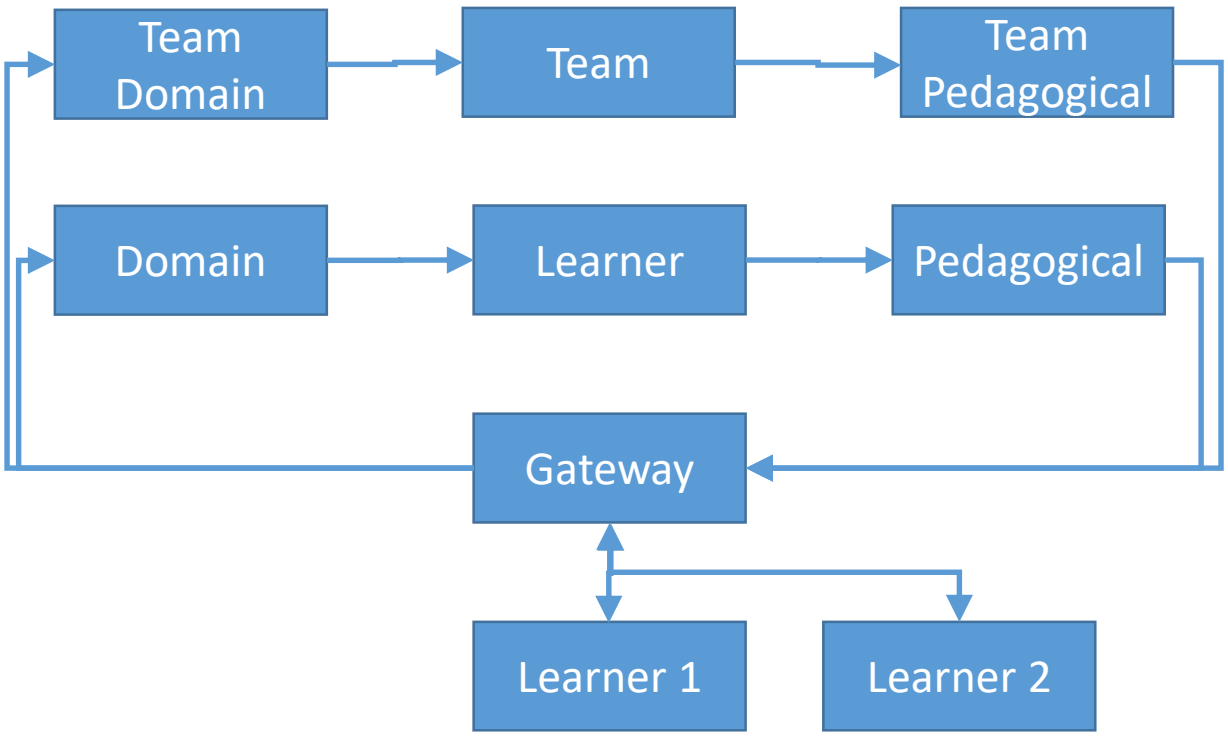

Figure 7. Model of Team Tasking overlaid to individual tutoring

Option 4 – "Role" Modules for Tasks, Team Roll-up

In an effort to moderate the complications of authoring team and individual tasks, and to standardize team taskwork and teamwork items, a structure of "team" Domain items, team modeling and team pedagogy can be considered. This mitigates many of the problems faced in the initial design. Further, it allows the common tools for team items without being particularly concerned about the individual domains. Authoring tools may be able to be templated for authoring team items, asking streamlined questions for individual domains such as "how often should team members communicate?" and "what should elements of a shared language or mental model be?". Further, it allows for team member to switch roles and not be penalized for accomplishing team goals; as a concrete example if the ASW Pilot and Co-Pilot switch seats but the total team mission is still accomplished. This type of design allows for an ease of authoring, as individual tasks can be mapped to one/more roles and evaluated at the "world" level for their accomplishment. Further, it allows for the portability of team pedagogy and models, abstracted from the domain.

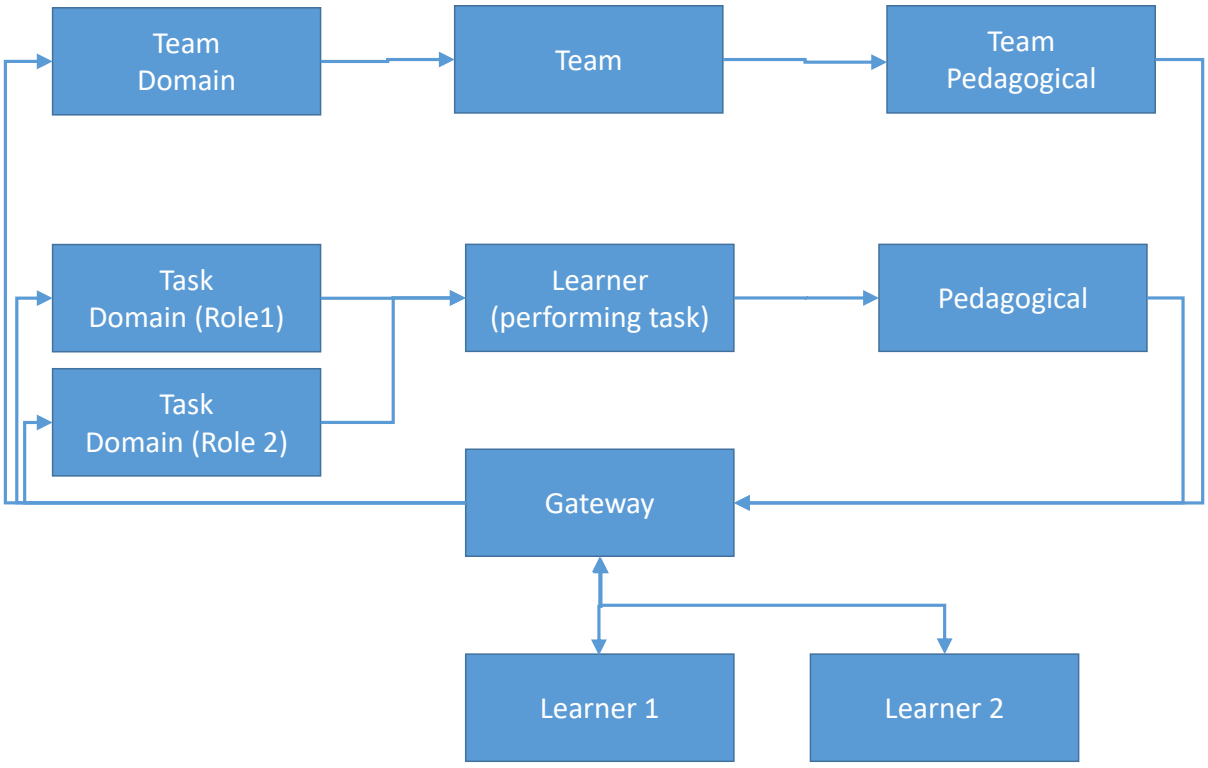

Figure 8. "Role" Based Model of ITTS

Design Recommendations for a Team Tutoring System

Table 1. Table of Design Tradeoffs

Individual Tutoring	Poor	Good	Good	Good
Team Taskwork	Good*	Good	Good	Good
Team Teamwork	Poor*	Good	Good	Good
Differing Team Structure	Fair	Good	Good	Good
Differing Team Roles	Fair	Good	Good	Good
Ability to be Authored	Poor	Fair	Fair	Good
	*Dependent upon authoring			
	**Many modules, difficulty in commodizing			

As we have described in this chapter creating a team tutoring framework is a difficult challenge. Creating a team tutor is difficult, but trying to account for all types of configurations of teams and retain flexibility makes it harder when designing an ITTS framework; achieving generalizability and scalability while not

compromising to individual items is fundamentally difficult. We have discussed different possible approaches of creating an architecture that can support multiple types of team tutoring and different domains. At the time of writing, it seems as though the creation of individual roles, logic for mapping individual team members to roles, assigning traditional tutoring over roles, and transferring team instructional models between domains as domain-general team training provides the greatest level of scalability and generalizability. This is what this paper recommends as a path moving forward, although implementation details such as "team member to role disambiguation" may prove more difficult than initially assessed.

References

Bonner, D., Gilbert, S., Dorneich, M. C., Winer, E., Sinatra, A. M., Slavina, 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., … 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.

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

Bonner, D., Slavina, A., MacAllister, A., Holub, J., Gilbert, S. B., Sinatra, A. M., . . . Winer, E. H. (2016). The hidden challenges of team tutor development.

Bonner, D., Walton, J., Dorneich, M. C., Gilbert, S. B., Winer, E., & Sottilare, R. A. (2015). *The Development of a Testbed to Assess an Intelligent Tutoring System for Teams.* Paper presented at the Workshop on Developing a Generalized Intelligent Framework for Tutoring (GIFT): Informing Design through a Community of Practice.

Cushard, B. (2012). Learning Design: When You Just Don't Know Where to Start. from http://www.bill-cushard.com/2012/10/learning-design-when-you-just-dont-know.html

Gilbert, S., Winer, E., Holub, J., Richardson, T., Dorneich, M., & Hoffman, M. (2015). *Characteristics of a Multi-User Tutoring Architecture.* Paper presented at the Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym3), Orlando, FL.

Gilbert, S. B., Slavina, A., Dorneich, M. C., Sinatra, A. M., Bonner, D., Johnston, J., . . . Winer, E. (2017). Creating a team tutor using GIFT. *International Journal of Artificial Intelligence in Education*, 1-28.

Gilbert, S., Dorneich, M., Walton, J., & Winer, E. (in press). Five Lenses on Team Tutor Challenges: A Multidisciplinary Approach. In J. Johnston, R. Sottilare, A. M. Sinatra & C. S. Burke (Eds.), Building Intelligent Tutoring Systems for Teams: What Matters. Bingley, UK: Emerald Publishing.

Roschelle, J., Suthers, D., & Grover, S. (2014). CIRCL Primer: Collaborative Learning. from http://circlcenter.org/collaborative-learning/

Salas, E. (2015). *Team training essentials: A research-based guide*: Routledge.

Selfridge, O. G. (1958). Pandemonium: a paradigm for learning in mechanisation of thought processes.

Sottilare, R., Burke, C., Salas, E., Sinatra, A., Johnston, J., & Gilbert, S. (2017). Towards a design process for adaptive instruction of Teams: A Meta-Analysis. *International Journal of Artificial Intelligence in Education*.

Sottilare, R. A. (2018). A Comprehensive Review of Design Goals and Emerging Solutions for Adaptive Instructional Systems. *Technology, Instruction, Cognition & Learning*, *11*(1).

Sottilare, R. A., Brawner, K. W., Sinatra, A. M., & Johnston, J. H. (2017). An Updated Concept for a Generalized Intelligent Framework for Tutoring (GIFT).

Sottilare, R. A., 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*.