Expert Systems With Applications 271 (2025) 126663

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A framework for adapting conversational intelligent tutoring systems to enable collaborative learning

Pablo Arnau-González *, Sergi Solera-Monforte, Yuyan Wu, Miguel Arevalillo-Herráez

Departament d'Informàtica, Universitat de València, Avda. de la Universidad s/n, 46100, Spain

ARTICLE INFO

Dataset link: https://zenodo.org/records/14163440

Keywords:
Collaborative learning
Human–computer interaction
Intelligent tutoring systems
Web development
Intelligent systems and AI in engineering

ABSTRACT

Despite the general consensus on the advantages of collaborative learning, most of the research in Intelligent Tutoring Systems (ITSs) only considers the case of individual learners. This is mainly due to the technical challenges in adapting individual learning systems to support collaborative learning. In this paper, we present a framework aimed at adapting individual-learner tutoring systems to enable collaboration among students, leveraging the advantages of both intelligent tutoring systems and peer collaboration, without significant changes to the original Human–Computer Interaction model. The proposed framework is designed under the assumption that the adapted ITS is built as a web application, ensures horizontal scalability of the ITS, and relies exclusively on Open Source software and tools. Evaluation results of the framework demonstrate that while the system's complexity increases, there are no perceivable rises in response times or cpu usage.

1. Introduction

In the landscape of education, technology has played a transforming role in redefining how we teach and learn. Intelligent Tutoring Systems (ITSs) emerge as a powerful tool in the digital educational evolution, offering students tailored, adaptive learning experiences (Mousavinasab et al., 2021). These systems leverage artificial intelligence and machine learning to provide personalized feedback and guidance, addressing each learner's unique needs and pacing, providing a positive impact on student's learning (Kulik & Fletcher, 2015). However, most ITS designs currently focus primarily on one-on-one teaching scenarios, where the system interacts with a single student. While effective in addressing individual learning needs, such designs lack the ability to facilitate collaborative learning. Collaborative learning, as an educational method that fosters knowledge construction through group interactions, cooperative tasks, or discussions, has been shown to significantly enhance students' critical thinking, communication skills, and teamwork abilities. Therefore, integrating the concept of collaborative learning into ITS design can further improve educational outcomes and provide students with a more comprehensive and diverse learning experience (Walker, Rummel, McLaren, & Koedinger, 2007).

Fusing ITS with collaborative learning allows students to assume both roles as teachers and learners, enhancing the consolidation of acquired skills and introducing the invaluable benefit of collaborative knowledge exchange (Kulik & Fletcher, 2015). By adopting this model, students benefit from the guidance of an intelligent tutor, who, in turn, plays a pivotal role in steering study sessions. This symbiotic relationship encourages a more comprehensive understanding of the subject matter, as students actively contribute to each other's learning experiences (Olsen, Belenky, Aleven, & Rummel, 2014a). However, the implementation of such a dual-capability system is a complex task that goes beyond mere technical intricacies. It necessitates a profound understanding of how students will engage with the system and collaborate with their peers. Factors such as user interface design, communication protocols, and the establishment of a conducive online environment must be carefully considered to ensure the seamless integration of both intelligent tutoring and collaborative learning components. Successful implementation will not only facilitate knowledge transfer but also foster a collaborative and interactive learning ecosystem that reflects the evolving landscape of modern education.

Hypergraph-based INtelligent Tutoring System (HINTS) (Arevalillo-Herráez, Arnau, & Marco-Giménez, 2013; Arnau, Arevalillo-Herráez, Puig, & González-Calero, 2013) is an ITS designed to assist students through the solution process of Math Word Problems (MWP), which present the statement of the mathematical problem in natural language instead of mathematical symbols. Unlike other conventional tutoring systems that can only provide tutoring on final answer (Conejo et al., 2004), HINTS leverages hypergraph theory to provide tailored guidance to learners. HINTS operates by constructing hypergraphs that represent

* Corresponding author.
E-mail addresses: pablo.arnau@uv.es (P. Arnau-González), sergi.solera@uv.es (S. Solera-Monforte), yuyan.wu@uv.es (Y. Wu), miguel.arevalillo@uv.es (M. Arevalillo-Herráez).

https://doi.org/10.1016/j.eswa.2025.126663
Received 19 June 2024; Received in revised form 21 January 2025; Accepted 22 January 2025
Available online 2 February 2025

the underlying structure of MWPs, effectively capturing the intricate relationships between variables, operations, and problem-solving steps. This approach allows HINTS to offer students not only fine-grain scaffolding but also a deeper understanding of the problem-solving process itself.

Collaborative systems remain scarce in the educational field, but they are anticipated to gain interest with recent advancements in language models. These developments could soon enable the use of agents acting as students (Arevalillo-Herráez, Ayesh, Rezakhanlou, & Arnau, 2024; Solera-Monforte, Arnau-González, & Arevalillo-Herráez, 2024), potentially addressing significant challenges, such as the need to group real students or combine multiple student models to ensure appropriate content sequencing. In this paper, we propose a framework for adapting Intelligent Tutoring Systems (ITS) to enable collaborative learning, using HINTS as a case study. We intentionally focus on educational systems and their specific needs, proposing a generic and adaptable solution tailored to this type of platform. The proposed framework leverages established open-source tools to offer a comprehensive and lightweight approach to session sharing. The method described minimizes the added processing overhead compared to individual interactions, as well as the changes needed to the ITS architecture, user interfaces, and underlying logic to incorporate collaborative features.

The rest of this paper is organized into 6 sections. Section 2 describes the related work in Intelligent Tutoring Systems. Section 3 details the proposed adaptation framework. Section 4 covers the adaptation of the HINTS system using the proposed framework. Section 5 evaluates the adapted system. Finally, Section 6 concludes the limitations and final considerations when using the proposed framework.

2. Related work

2.1. Intelligent tutoring systems

Research on ITSs is an ongoing and interdisciplinary topic, involving pedagogy, psychology, and computer science (Guo et al., 2021). The evolution of ITSs has spanned over 50 years, starting with early relevant works such as the SCHOLAR system in the 1970s (Carbonell, 1970), shortly followed by similar systems (Brown, Burton, & Bell, 1975; Clancey, 1979; Stevens & Collins, 1977).

In general, ITSs can be classified into two categories according to the allowed interaction with the student: constraint-based (Mitrovic, Martin, & Suraweera, 2007) or model tracing (Blessing, Gilbert, Ourada, & Ritter, 2009). Constraint-based systems operate within a framework of strict rules that must be adhered to without any deviation. This allows students to employ their own problem-solving procedures within these predefined constraints. Model tracing systems have a representation of how the mental process for a given problem should be, and guide the student through the scheme.

ITSs have a presence in a wide variety of domains. According to a recent literature review study (Mousavinasab et al., 2021), most research is currently focused towards the implementation of ITSs for Computer Science, Math, and Health Sciences/Medicine. The study also found that more than 70% of the systems examined specialized in teaching a specific subject and, therefore, confirming that most of the Intelligent Tutoring Systems are predominantly aimed at instructing in a single topic.

In math education, ITSs have been employed to help students develop their skills in different areas such as math word problem-solving (Arevalillo-Herráez et al., 2013; Nguyen, Tran, Do, & Pham, 2020), algebra (Jaques, Seffrin, Rubi, de Morais, Ghilardi, Bittencourt, & Isotani, 2013; VanLehn, Banerjee, Milner, & Wetzel, 2020) and logic (Lanzilotti & Roselli, 2007). These systems match the user input with the expected answer and provide adaptive feedback based on student responses. Nguyen et al. (2020) designed a conversational intelligent tutoring system based on question answering to address high schools in algebra problem-solving. The ITS consisted of a chatbot that the students used to solve a series of math problems. Similarly, Arnau-González, Arevalillo-Herráez, Albornoz-De Luise, and Arnau (2023) adapted a button-based conversational for solving algebra and arithmetic problems. Other systems, such as RadarMath (Lu, Pian, Chen, Meng, & Cao, 2021) proposed to automatically grade students and based on that grade, provide tailored recommendations on subsequent topics to study.

Computer science has also been a target area for research on ITSs. The recent literature review (Francisco & de Oliveira Silva, 2022) found that most of the systems were related to software development (which includes algorithmic design and other programming concepts), although other topics such as network architecture were also present (Al-Hanjori, Shaath, & Naser, 2017; Alshawwa, Al-Shawwa, & Abu-Naser, 2019). Relevant works in this area include Hooshyar et al. (2018) which proposes a solution-based approach for training the problem-solving skills by offering the student possible solution in the flowchart design process. Other systems opted to use tips during the activities (Price, Dong, & Lipovac, 2017), or a socratic approach (Alshaikh, Tamang, & Rus, 2021).

Regarding interaction, traditional systems often interact with the student through a button-based approach. However, there has always been an interest in making students interact naturally with the system (Albornoz-De Luise, Arnau-González, & Arevalillo-Herráez, 2022). Conversational Intelligent Tutoring systems facilitate learners interaction with intelligent tutors through natural language improving the learning experience. Its main capabilities include comprehension of the natural language, generating adaptive responses, and employing pedagogical strategies to support individual learning style. Conversational ITS has been applied in various subjects. AutoTutor (Nye, Graesser, & Hu, 2014), MyScienceTutor (Ward et al., 2013), are tutors designed to support students while learning basic math and science. Siette (Conejo et al., 2004; Conejo, Guzmán, & Trella, 2016) is focused around evaluation on broad topics, such as humanities. STUART assisted students problem in an e-learning environment, demonstrated using intelligent agents can effective reduce workload of human tutors (Damasceno et al., 2020).

Recently, researchers have chosen to create highly personalized systems following a multi-agent approach. This trend has shown capabilities for providing students with diverse and more personalized guidance and assistance (Lippert, Shubeck, Morgan, Hampton, & Graesser, 2020). For example, Mahmoud (2018) presents AGTutor, an ITS for teaching Arabic grammar in Egyptian schools. The system is composed by 5 agents that manage different aspects of the interaction with the student, such as the interface, or the learning strategy. By breaking the traditional components of an ITS into smaller pieces, the authors create expert modules that adapt better to the specific needs of the student.

2.2. Collaborative intelligent tutoring systems

In the context of Intelligent Tutoring systems, typically the students interact only with the tutor without reciprocating with peers. Over the years, researchers have developed tutoring systems that allow collaboration. Collaborative Intelligent Tutoring Systems (CITS) have evolved from ITS. A CITS provides a collaborative learning environment, allowing students to interact not only with the computerized system but also with other students for mutual understanding and explanation.

Stahl (2013) states that the process of collaborative knowledge-building relies on two key components: (a) personal individual understanding and (b) social knowledge-building. Initially, individuals develop tacit pre-understandings of the world. When these pre-understandings lead to misconceptions or problematic interpretations, they must be revised or corrected. This revision can occur internally, through the reinterpretation of existing knowledge structures, or externally, through interaction with artifacts and symbolic representations. If the newly reinterpreted tacit understanding successfully resolves the issue, it is integrated as acquired knowledge. However, if the problem

remains unresolved, social knowledge-building takes place. In this social process, the individual articulates their idea to peers, who, in turn, contribute their own perspectives. These perspectives may or may not align with the original idea, leading to discussion and debate. Through rational argumentation—assuming transparency and a genuine intent to collaborate—the group reaches a shared understanding, which becomes accepted knowledge. Once consensus is achieved, the new knowledge may be externalized through artifacts, such as mathematical symbols or conceptual models, forming the foundation for further knowledge construction. Regarding knowledge acquisition, collaborative learning fosters communication, critical thinking, problem-solving skills, and a deeper understanding of the subject matter (Ghavifekr, 2020). From a social perspective, it can reduce anxiety, enhance motivation, improve social skills, and teach students how to work effectively in teams (Laal & Ghodsi, 2012; Olsen, Belenky, Aleven, & Rummel, 2014b).

The review presented in Ubani and Nielsen (2022) classified CITS in whether they provided feedback from their problem-solving skills or their verbal/non-verbal interaction. Relevant works on CITS include (Olsen et al., 2014a, 2014). Which design an extension to the Cognitive Tutor Authoring Tools to support collaborative learning environment through multiple synchronized tutor engines. Olsen et al. (2014a) developed a CITS for teaching equivalent fractions. The collaborative tutor supported synchronized web collaboration that enabled two students to work together in the same problem through a chat system. However, the interaction was limited as the student's decision would not alter the response of other students. In order to see other peers' proposed solution, students needed to submit their own answers. This means that the proposed system did not allow for two or more users to solve the same problem instance simultaneously.

In the field of medicine, relevant works include COMET, a collaborative approach to an intelligent tutoring system, which was developed from the year 2004 onwards (Suebnukarn & Haddawy, 2004). The researchers analyzed the techniques that tutors employed for teaching medical reasoning to medicine students during collaboration. The work was later expanded in Suebnukarn and Haddawy (2006a, 2006b), where they implemented a Bayesian network-based approach to stimulate the learning of medicine students. COMET's collaborative learning approach consists of having all the students work on a single problem and providing them with an area for creating hypotheses and open discussion. Whenever one of the students created a hypothesis, this hypothesis appeared in all the students' hypotheses. This approach proved similar reasoning paths to students interacting directly with a human tutor.

Furthermore, not only the collaboration between students needs to be taken into account, but also how the students are grouped when solving the problem. As such, there are different strategies regarding the grouping of collaborative systems that must be taken into account when implementing a CITS. The main strategies in this regard are the student-selected groups, on which students freely choose their collaborators; the randomly-assigned groups, in which members are put together randomly; homogeneously-assigned groups, on which students are grouped based on their academic similarities, and heterogeneously assigned groups, which, in contrast, are formed with students to create a group with a variety in skills, ranges, gender, etc. Nhan and Anh Nhan (2019). The pros and cons of each strategy, as well as the complexity of their implementation, must be taken into account when designing a collaborative system.

While most of the collaborative systems that have been studied provide some degree of collaborative participation, direct interactions among students within the same problem instance remain limited. Hence, in terms of collaborative learning as '[...] engaging in active dialogue, sharing ideas, and working together towards common goals' (Major, 2020), is only partially achieved in most CITS implementations, while the number of CITS that successfully facilitate active dialogue and simultaneous problem-solving on the same problem-instance on a unified interface remains reduced.

However, Collaborative Intelligent Tutoring Systems still face technical issues in their implementation. According to Haq, Anwar, Basharat, and Sultan (2020) most collaborative systems do not implement an optimal approach for merging collaborative capabilities and fail to propose a truly collaborative system, many collaborations are created in the classroom (outside of the system) or the collaboration happens asynchronously. Examples are systems like Olsen et al. (2014a) in which students interact through an audio chat but the response is individual, or Walker, Rummel, and Koedinger (2014) which allows for chat communication, but assigns the role of teacher to one of the members of the collaboration. More importantly, we found that out of all the studied works, none provided technical details that allowed for a replication of the system architecture. Unlike traditional ITS, CITS must balance individual and collaborative learning, support diverse collaboration structures, and account for cultural differences (Olsen, Belenky, Aleven, & Rummel, 2013; Olsen, Belenky, et al., 2013). Moreover, there have been recent attempts to create a collaborative framework for ITS. Haq et al. (2020) propose a collaborative intelligent tutoring system framework. The proposed methodology highlights the need for collaborative systems to have an online collaboration and an intelligent tutor that provides feedback on the user's actions. However, the study does not delve into the technical intricacies of implementing collaboration in an ITS beyond arguing that the general architecture must follow a Model-View-Controller design pattern. Similarly Lahmadi, Ouadoud, El Khattabi, Rahhali, and Oughdir (2024) recalls the same weaknesses on current CITS and tries to establish a common framework for the implementation of CITS. This work provides a more in-depth analysis of the requirements to implement collaboration in practice, however, still fails to discuss technological options for implementing collaboration in ITS.

3. Adaptation framework

The proposed framework targets intelligent tutoring systems that currently provide live feedback to students while they engage in learning activities and plans to support collaborative learning by allowing several students to concurrently interact in the activity.

3.1. Issues of the adaptation to collaborative environment

There are three obstacles to overcome in order to enable collaborative learning on an ITS and the first two are related to the *Current Solution Status Object* (CSSO). The CSSO is the representation of the status of the resolution of the problem/exercise being solved by one or more students. It usually stores information regarding the resolution of the problem, e.g, on an algebra tutoring system, the CSSO contains information regarding the identified letters, equations, and others.

The first problem is related to how sessions work. Generally, every action the user performs on the problem is annotated on the CSSO, which is stored as a session variable. Session variables are typically stored by making use of Cookies on the user's browser or on in-server databases, tying the CSSO to the user session, makes it inaccessible by other clients, thus, preventing modifications on the resolution status of the problem. Therefore, in order to allow multiple participants to collaborate on solving the same problem collaboratively it is necessary that different users can act on the same CSSO.

There are several ways of allowing multiple clients to change one object on the server, however, they all come down to having that object accessible for all the users and allowing write access to the object to the clients belonging to the same resolution group.

The second problem is related to how the rest of the clients are notified when there is a change in the CSSO. Typically REST applications, rely on requests to the web server with HTTP verbs (GET, POST, PUT, etc.) to get, or change the status of objects in the server. However, HTTP(S) responses are only sent to the requester. Having multiple

clients perform GET operations just to get CSSO's real-time updates would lead to an unnecessary increase in traffic, especially considering that users will, generally, take a certain amount of time between each action on the CSSO and respond accordingly to the feedback proposed by the system.

Since actively polling the server is not a viable solution, a third problem arises: how to notify the participants of a room about relevant information regarding the current session, which may or may not be related to the CSSO. This includes changes, such as a user entering/leaving the collaboration, the exercise being reset or passed by another participant or a teacher, among others.

In summary, the main obstacles to overcome on a collaborative ITS are those regarding the update of the CSSO by multiple users while also keeping track of the relevant information of the group (e.g., usernames, notes, messages, etc.).

3.2. Proposed architectural adaptation

The process of adapting an ITS into a collaborative environment requires that the identified issues are overcome. To do so, we propose dividing the ITS into three separate layers, namely the persistence layer, application layer, and communication layer.

First, the persistence layer is responsible for storing all the objects related to the system's operations such as users' data, problem statements and, especially, the CSSO. Storing the CSSO on a separate layer allows us to address the issue of having several users accessing the solution status by providing a centralized point onto which the CSSO may be modified. On the persistence layer, the CSSO may be stored either using an external service for the storage of ephemeral objects or in a database.

While the CSSO object can be stored in both SQL and NoSQL databases, storing it in a relational database ensures data integrity but may be challenging due to its complexity. In this context, in-memory databases are often used to externalize session variables for multi-tenant applications, particularly those that support serialized object storage. Storing serialized objects simplifies storage and migration of the ITS into a collaborative environment while also reducing complexity.

Second, the communication layer will handle communications between the server and the group of students solving a particular problem, in a way that users may be notified whenever an action on the CSSO has been acted upon. This is done by implementing a Publication/Subscription (Pub/Sub) mechanism on which users subscribe to the CSSO's updates, and get notified whenever there is a change in the CSSO. In this case, there are a variety of tools suitable for the task, and the one to be used will depend on the software stack the ITS is built on.

Third, the application layer is responsible for interacting with the other two layers as well as with the students via a web interface. It acts as a bridge between the CSSO and users, so whenever the user sends an action to the system, the application layer modifies the CSSO and communicates the system's feedback to the students through the communication layer. Furthermore, the application layer must also contain a module in charge of handling the information on each group, so that, as previously mentioned, the system can also notify the user, not only of the changes on the CSSO, but other information as well (e.g., usernames, messages between users, etc.).

In our specific scenario, our proposal introduces a component for automatic group generation. This component is responsible for creating groups of users that want to solve a specific problem. The main objective of the GroupMaker is to gather the students who are waiting to solve an exercise and create the groups following a grouping strategy, which depends on the specifics of each ITS (Nhan & Anh Nhan, 2019). However, the groupmaker is not extensible to other systems that might prefer a manual assignment of groups.

Finally, Fig. 1 illustrates the proposed framework schema for a collaborative session. In this case, the adaptation to enable collaborative learning is composed of the following steps:

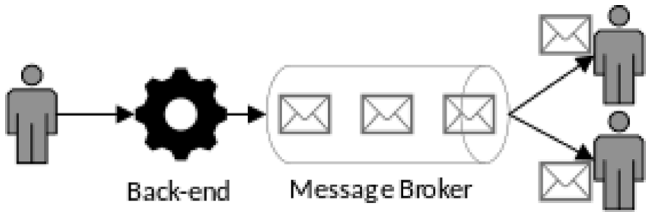

Fig. 1. In a collaborative session, the input from each user in the room is sent to the back-end, where the ITS will provide the updated CSSO based on the student's input. Such CSSO will be passed on to the message broker, and sent to the students.

1. A student is placed on a room.[1]
2. The system assigns each student on the room an identifier (i.e. an UUID) for a specific problem and CSSO: The identifier shall be the same for each user on the room. This information is stored on the persistent layer.
3. The system subscribes to the updates of the solution identifier.
4. The student sends input to the system.
5. The system processes the student's input and notifies that the current solution status has changed.
6. Students' interfaces are updated accordingly.
7. Steps 4 to 6 are repeated until the problem is finished.

To sum up, the proposed adaptation for the ITS framework consists of a) centralizing the CSSO in order to allow modifications by different users and b) processing modifications on CSSO and notifying users accordingly through the division of the system into specialized layers for such tasks.

3.3. CSSO synchronization

In the proposed ITS framework, maintaining the synchronization of the Collaborative Solution Status Object (CSSO) is essential for enabling smooth real-time collaboration among students. This synchronization ensures that each student is working with the most up-to-date version of the solution, reflecting the changes made by others in real-time. The synchronization of the CSSO is managed through the interaction of the communication layer and the persistence layer. The CSSO is stored centrally within the persistence layer, ensuring that all modifications to the CSSO are made in one location. This centralized storage model prevents discrepancies between different users' versions of the solution, allowing the system to efficiently manage and propagate updates to all participants.

The proposed architecture is tailored for collaborative scenarios where changes to a central shared object are typically infrequent, structured, and not highly sensitive to the order of actions. This design choice allows for reduced overhead by not requiring fine-grained data handling mechanisms like Operational Transformation (OT) (Ellis & Gibbs, 1989) unless necessary. However, in contexts requiring continuous, fine-grained user interactions (e.g., collaborative document editing), OT mechanisms or Conflict-free Replicated Data Types (CRDTs) may need to be implemented to maintain data consistency across distributed clients in real-time.

To support multiple types of interactions, the architecture could adopt a hybrid approach where coarse-grained updates are managed through traditional lock-based or lock-free synchronization mechanisms, while finer-grained changes leverage OT or CRDT frameworks. This dual-layered approach would ensure that resources are used efficiently in scenarios where coarse and fine-grained changes coexist.

[1] As previously mentioned, the grouping strategies may be specific to the system which is being implemented.

3.4. Proposed interface adaptation

The User Interface (UI) of an ITS heavily depends on the system itself, as well as the interactions allowed within the system. However, certain remarks must be kept in mind when adapting the interface to support collaborative learning and allow students to interact naturally between them and the ITS.

First of all, the importance of the UI does not only rely on giving the user a mechanism to communicate with the system. It also allows for feedback regarding relevant information.

In general, the interface should be designed with an interaction-first approach (Lowyck & Pöysä, 2001). It should be clear to the students whether they interact with the system, with their peers, or with both, if that is a valid communication case in the specific ITS. Therefore, the interface must provide the required information for students to know who they are interacting with, and change the type of interaction if desired.

In the scenario of a conversational system, the students must be able to choose whether the message is sent to the system or to the rest of the group. Some options to include this behavior could be having a separate button for communicating with the system/the peers or creating two separate communication containers. Advanced systems could implement a module that predicts the destination of the user input while adhering to the established guidelines for building Human-AI systems (Amershi et al., 2019). Failure to comply with these guidelines might have a negative impact on the user experience.

Another aspect to consider when adapting the interface of a collaborative system is that most systems allow the student to take notes. It is a system design choice whether all the notes are shared with the rest of the group, or each of the students has a private, and a shared note section. Whichever the case is, students should have, at least, a shared section for note-taking. Moreover, this section should be updated in real-time with the notes from all the other members of the group.

In summary, the key aspect of adapting ITS interfaces to allow online collaborative interaction between the students is the capability of the user interface to refresh in real-time with the information provided by the rest of the group.

4. Case-study: Adaptation of the HINTS system

4.1. Description of the original system

Hypergraph-Based INtelligent Tutoring System (HINTS) is an Intelligent Tutoring System focused on supporting individual students in acquiring skills for arithmetical/algebraic problem solving by tutoring the student through problems posed as Math Word Problems (MWP). HINTS is capable of supervising the solution process for a given MWP. It achieves this by employing a hypergraph-based representation of the problem at hand. The system is capable of navigating the graph in order to establish the relationships proposed by the student or correct them whenever the student has misidentified part of the relation, e.g. when the student confuses an operator between *quantities*[2] or directly no relation exists between the mentioned quantities.

To this end, users are provided with a chat-based interface for solving mathematical problems, while chatting with the system. This interaction allows users to send messages like *"let x represent the money spent on sweets"* or *"$x = 20 + y$"*, with the system duly annotating these actions within the graph representation of the problem and providing relevant feedback to the user as messages and personalized notes.

The system operates as a stateful web application. The original system has the following components: a database, and a web application divided into both front-end and back-end. Logically, HINTS is divided in two layers, the application and the persistence layer, as shown in Fig. 2. In the original implementation, the web server retains the CSSO as a session variable. Subsequently, the server handles this information, updating the CSSO based on the resolution status that corresponds to the user's input (i.e., defining a variable or identifying an equation), and notifying the user.

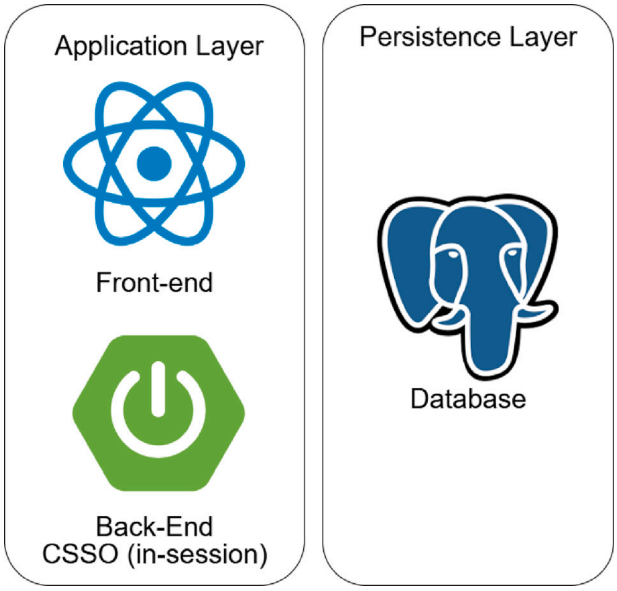

Fig. 2. Architecture diagram of the original HINTS System.

4.2. Architectural adaptation of HINTS

Following the proposed scheme for adaptation, the HINTS system has been divided into the proposed layers. Fig. 3 represents an architecture diagram depicting the different layers and how they are connected for the HINTS case study. Each layer has to be implemented in a way to minimize the complexity, while also focusing on modifying the ITS to a minimum.

Firstly, the application layer is formed by the back-end, which holds the business logic related to handling the system (e.g. waiting rooms, users, etc.), the ITS itself, and the front-end, to allow the user to interact with the system and receive feedback. As mentioned before, the technologies to be used on the application layer are to be chosen regarding the ITS. In our implementation, the back-end of the application layer has been implemented in Java using Spring due to the legacy application, including the ITS, being implemented in such technology. In our implementation, the Spring back-end is also responsible for handling the groupmaker process. Similarly, React.js has been used as front-end technology, following the original implementation of the HINTS system

Secondly, the persistence layer is composed of two components, a PostgreSQL database and a Redis in-memory database.

The PostgreSQL database manages the long-term storage of data, including users grouped in a room, the exercises, translations, exercise settings, and others.

On the other hand, RedisDB is used to manage a) the CSSO associated to each room on a key–value pair (e.g. a key *room-id* whose value is the CSSO itself, with information such as the exercise, notebook, users associated to such CSSO, and so on), and b) the users that are queued waiting for a room to be assigned (e.g. a key *waitroom-id* whose value is the users who are currently waiting to solve the exercise collaboratively on such room). As such, to implement waiting rooms on the proposed system, in the case of collaborative examples, the *id* is

[2] A *quantity* being an abstract representation of a part of the problem (e.g. "*Susan's age*" on a MWP).

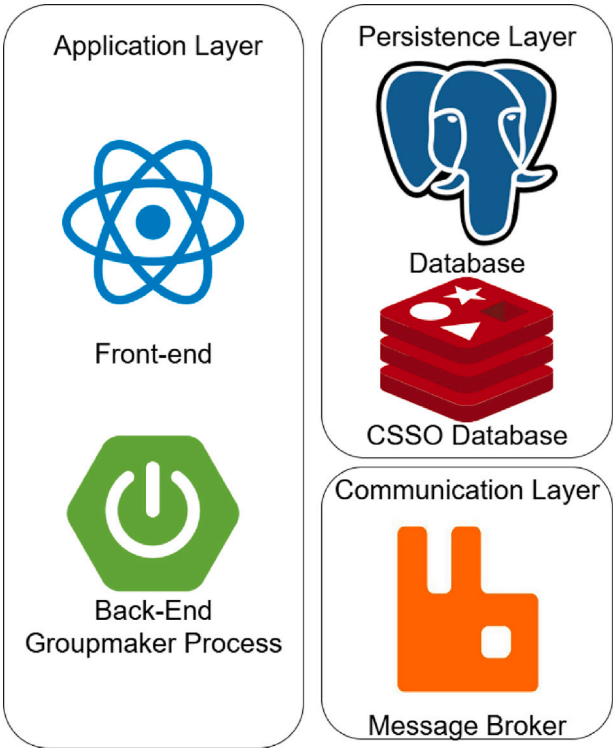

Fig. 3. Architecture diagram of the proposed adaptation for collaborative learning.

shared by all the users in the same chat room. In order to assign the *id*, a scheduled process is created on the application layer that consults for queued users waiting to solve a specific problem on the Redis database, and creates groups of N participants following a predefined grouping strategy (in the specific scenario of HINTS, $N = 2$ and the grouping strategy is random).

Finally, the communication layer, which is responsible for the asynchronous communication between the users in each room, is composed of a RabbitMQ message broker, which manages the rooms as message queues. The motivation behind using RabbitMQ on the communication layer is to implement a broker messages that implements the STOMP protocol.[3] The STOMP protocol allows clients to send and receive messages across different platforms. It allows them to send messages to topics, which will in turn be processed by the consumer. Specifically, the topics for which RabbitMQ and the STOMP protocol will be used are the topics *topic/username* and *topic/room-{uuid}*, where the *username* is the username of the user waiting for the room to be created and the *uuid* represents the identifier of the room once it has been created and stored on RedisDB. The messages sent to that room will be resent to that topic so that users of the room can receive them. When the user wants to solve a problem, they will be placed in a waiting room, and will subscribe to their own topic (topic/{username}), once the room is created (the number of users for the solving to start has reached N), the room uuid is then sent to the user's topic, which means the room has been created. From that point on, all the participants will be able to communicate and edit the CSSO by sending messages to *topic/room-{uuid}* all messages can be sent to that room. Since HINTS provides users with a limited set of actions, the likelihood of concurrent modifications compromising the CSSO's consistency is low. CSSO write operations only occur during equation definitions and when assigning letters to quantities. To prevent inconsistencies due to data dependencies (e.g., a message reading the CSSO state before another message completes processing), a semaphore can be implemented to lock the CSSO during each message processing. If multiple peers attempt to submit equations simultaneously, the system can process them in any order to determine validity. The primary scenario that could lead to inconsistencies is if two or more students simultaneously propose different letters to name the same quantity. In this case, processing messages on a first-come, first-served basis would allow the system to accept the first proposal and reject subsequent ones.

The steps on the system whenever a user selects a problem to solve are managed as follows:

1. The user selects a problem to solve.
2. Upon selection, the client's front-end initiates a request, sending an HTTP request to the system to create a chat room dedicated to solving the selected exercise and also subscribes to their topic.
3. This HTTP request is received by the back-end, which then assigns the user to a waiting room, denoted as waiting-room-*id*, where *id* represents the unique identifier of the exercise, within RedisDB.
4. Concurrently, the back-end establishes a topic specific to the user, formatted as a topic.username. This topic is designated for communicating essential information about the room, including the room's identifier, which the user will utilize once the room is allocated.
5. A scheduled job within the system periodically scans the waiting rooms at set intervals. Utilizing a strategy that groups users into pairs (N = 2) based on a random selection method, the job facilitates the creation of a room for users waiting in the same room once a pair is identified.
6. As soon as two users are paired in the waiting room, the scheduled job transitions them to a designated chat room, identified by a unique identifier (e.g., a UUID).
7. Following the creation of the chat room, the back-end notifies each user by publishing the room's identifier on their respective topics.
8. Upon receiving the notification, the client gets the CSSO information of such room and establishes a bidirectional connection with the system, enabling real-time communication within the room.
9. Through the connection, the client facilitates the exchange of information between the participants and the Intelligent Tutoring System, or among the peers, regarding the room's activities.
10. When a message is sent, the back-end acquires a semaphore for the specific CSSO in order to update it. Once the message is processed, and the CSSO is update, it releases the semaphore.
11. This process of information exchange continues until the exercise is completed.

In order to further illustrate the changes required to allow for Collaboration, we have created a GitHub repository that contains an illustrative Merge Request[4] displaying most of the changes required to adapt our system to allow collaboration. Although the shown code is not functional as it needs additional classes, it serves to show that most of the changes required to implement collaborative functionality are related to how the CSSO is loaded from the `HttpSession` in the case

[3] A different option that would reduce the deployment complexity would have been to use Redis for the communication layer as well, as it offers Pub/Sub capabilities. However, Redis is not compatible with the STOMP protocol, and using a different protocol for the websocket communication would significantly increase the logic in the server, as Spring provides built-in support for STOMP.

[4] https://github.com/SPAM-research/CollaborativeFrameworkDemo/pull/2/files

Table 1
Mean response time and CPU usage.

Users	Original		Framework		Collaborative	
	Response time (ms)	CPU (%)	Response time (ms)	CPU (%)	Response time (ms)	CPU (%)
100	16.713 ± 1.71	1.928 ± 0.62	20.313 ± 1.51	1.913 ± 0.54	26.693 ± 35.86	2.490 ± 10.09
200	31.283 ± 39.13	7.875 ± 2.01	19.577 ± 1.46	4.860 ± 1.40	34.849 ± 20.69	8.239 ± 2.35
300	53.886 ± 35.13	11.477 ± 1.36	19.756 ± 2.09	8.935 ± 0.89	70.537 ± 54.33	13.582 ± 3.03
400	97.679 ± 66.25	13.905 ± 1.68	22.923 ± 165.17	12.161 ± 2.47	120.834 ± 155.86	16.607 ± 2.26
500	102.217 ± 70.12	14.372 ± 1.44	25.821 ± 64.54	15.016 ± 2.06	129.809 ± 135.33	17.654 ± 2.14
600	133.260 ± 84.47	15.762 ± 1.31	71.592 ± 354.81	18.621 ± 3.79	156.005 ± 163.28	19.311 ± 2.27
700	152.621 ± 77.20	16.949 ± 1.35	126.329 ± 80.92	21.379 ± 2.01	243.237 ± 133.68	21.295 ± 1.83
800	183.506 ± 111.48	17.635 ± 1.65	209.995 ± 686.02	22.649 ± 2.81	252.507 ± 152.14	21.768 ± 1.94
900	229.602 ± 346.49	19.458 ± 3.11	254.795 ± 405.91	24.048 ± 3.04	294.145 ± 266.73	22.904 ± 2.02
1000	339.913 ± 211.17	19.466 ± 2.04	343.672 ± 609.73	26.099 ± 4.38	265.470 ± 144.35	23.198 ± 2.37

of the legacy system, or from the Redis. Additionally, we have included the `GroupMaker` process of HINTS to illustrate a simple algorithm for creating the user groups.

In summary, the HINTS system was adapted for collaborative learning by a) reorganizing its architecture into application, persistence, and communication layers, b) utilizing different technologies for the application layer and data management, and c) introducing RabbitMQ for asynchronous messaging, enabling real-time collaboration among users. It is worth noting that, as presented, the technologies can be agnostic from the implementation, as a wide variety of technologies allow the implementation of the proposed architecture.

5. Validation of the proposed adaptation framework

The proposed framework is based on the initial hypothesis that adequate logical separation of the components will not have a negative impact on the system performance, although there might be a slight increase in the response time due to added communication with extra services.

The proposed framework's performance will be validated against the original system in terms of response time and CPU usage. This analysis is done by testing and evaluating the same scenario the original system and the system adapted to the proposed framework. The scenario simulates the behavior of N users accessing the system and solving a given problem. This evaluation has been carried for $N = 100$, 200, …, 1000 concurrent users, each evaluation has been repeated a total of 50 times.

The implementation of the original system and the proposed framework for the testing have been deployed in a Kubernetes cluster, in a constrained environment, with a maximum assignation of 3 vCPU cores and 5 GB of RAM. Furthermore, the Java Virtual Machine, has been configured to allocate 4 GB at maximum of heap memory.

The evaluation of the response time of the system will be done using the Apache JMeter tool,[5] which allows for load-testing web applications. The CPU usage analysis of the web-server has been obtained using the PodInsights tool,[6] which reports CPU usage metrics on milliseconds intervals.

During the system evaluation, each emulated user solves a given mathematical problem, this is done by sending three requests. First, the user gets the problem specification, then defines a variable, and finally solves the problem. The performance of the ITS is heavily influenced by the number of users and the corresponding volume of messages sent, which will then be subject to analysis. To ensure a fair comparison between all the experiments, all users are solving the same problem and sending exactly the same messages. Since the internal system is deterministic, this ensures that the internal computation is identical for all user messages.[7,8]

Table 1 shows the mean processing time and CPU usage for a) the original implementation; b) the proposed adaptation in a single setting; and c) the proposed adaptation in a collaborative setting. As can be observed, the system provides higher response times for the proposed framework version, this is due to overhead produced by the communication with the extra services. The difference between the original and the adapted system is in all cases below 100 ms which is still below the threshold of 1 s for freely interacting with information (Miller, 1968). Both tables show that the average difference is below that threshold.

These results are displayed in Figs. 4 and 5, which show the average response time and CPU usage, together with their 80% confidence interval.

Although the proposed framework introduces extra processing, results show it will not be perceived by users and that it will not impact their quality of experience, as the average response time remains below the 300 ms threshold for less than 800 concurrent users. Despite the minor drawback of a barely noticeable longer response time per message, the benefits of enabling collaborative learning make this trade-off worthwhile. Furthermore, in cases where an increase in response time may not be deemed acceptable for certain Interactive Tutoring Systems (ITS) where timing is crucial, or where delays could be significantly larger due to the ITS's construction, the proposed system's distributed nature facilitates the implementation of self-scalable services to overcome the delays that may appear on these tutoring systems. The extra serialization of data increases the computational complexity, therefore increasing the CPU usage. The reported results show that CPU usage increases consistently with the number of users for the original and proposed framework.

6. Conclusions

We have presented a framework for adapting Intelligent Tutoring Systems (ITS) to support collaborative learning environments. The presented work, builds upon recent works by Lahmadi et al. (2024) and Haq et al. (2020) presenting a methodology for adapting ITS to support collaboration. By leveraging distributed systems and open-source technologies such as Redis and RabbitMQ, our approach enables real-time collaboration among students while maintaining the integrity and responsiveness of the ITS. The framework is particularly well-suited for ITS that provide feedback on short actions, such as mathematics-oriented tutoring systems (e.g., RadarMath) and short-question tutoring systems (e.g., Siette (Conejo et al., 2004), AutoTutor (Nye et al., 2014), and MyScienceTutor (Ward et al., 2013)). However, ITS requiring precise synchronization for shared editing tasks may need additional consistency mechanisms, such as operational transformation (OT) methods (Ellis & Gibbs, 1989).

[5] https://jmeter.apache.org/
[6] https://github.com/cloudmedialab-uv/PodInsights
[7] Data available at: https://zenodo.org/records/14163440
[8] It is important to note that our Hypergraph-based ITS processes each message internally. The performance impact from thousands of users will be significant, but the specific impact of message volume or concurrent users will vary based on the ITS implementation.

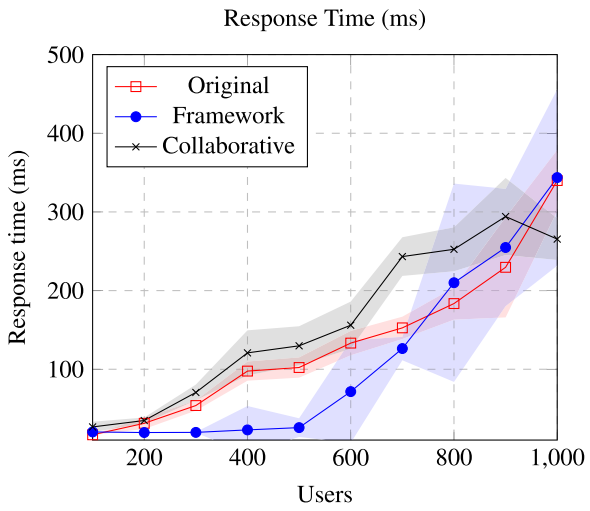

Fig. 4. Response time (ms) for different amounts of user load (Original vs Proposed Framework).

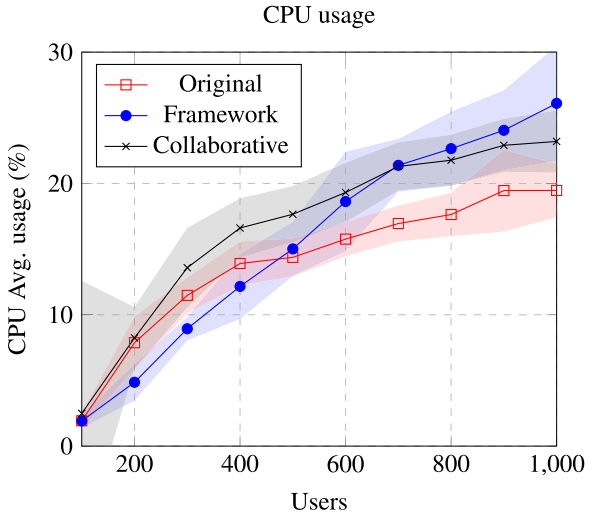

Fig. 5. % CPU Usage for different user load (Original vs Proposed Framework).

The proposed framework aligns with the recommendations by Collazos et al. (2007), who emphasize the importance of fostering collaboration and providing software tools that facilitate real-time interaction. By enabling the integration of real-time chat interfaces, our framework directly supports these objectives, allowing students to communicate, share ideas, and collaborate actively in real time. This promotes a more engaging and dynamic learning environment, fostering deeper collaboration between learners and enhancing the overall educational experience. Complementing this, the shared CSSO provides a common and synchronized view of the problem's progress, directly supporting the principles of shared understanding and mutual awareness, and ensuring all students are working with the latest information and can effectively coordinate their efforts. Moreover, the framework imposes no restrictions on the type or periodicity of the feedback provided by the tutor. It allows for flexible feedback mechanisms tailored to the specific needs of the tutoring system. In addition, it supports collaborative learning by analyzing student interactions, which are recorded, at least in the CSSO. This allows the adopting system to offer collaborative support based on real-time data, enhancing the learning experience and promoting more effective group dynamics.

Although our framework does not specifically collect data that could assist the tutor in evaluating the quality of the collaborative interaction, it is practically possible to analyze interactions in the CSSO to infer insights that can guide the collaborative feedback process. Second, the framework focuses on chat-based communication, which, while effective, may not fully capture the range of collaboration needs in some educational contexts. More comprehensive multimodal support, such as shared whiteboards or video feeds, could enhance collaboration, especially in tasks that require visual interaction or complex problem-solving, which in turn will require the implementation of discussed granular control over the CSSO. Finally, the real-time data exchange and communication involved in the framework raise potential security and privacy concerns. Adopting systems should implement their own additional measures, such as encryption and user authentication, to ensure the privacy of student interactions and protect sensitive data.

To assess the proposed adapted system performance, we compared the proposed framework to our original implementation using a non-inferiority approach. While the new layers introduce additional response time and latency, the delay remains within acceptable limits for real-time interaction—less than a second—leading to improved user productivity and satisfaction (Shneiderman, 1984). Furthermore, our evaluation indicates minimal differences in CPU usage between the original and proposed frameworks, even in collaborative scenarios, reinforcing the efficiency of our approach. Moreover, a demo version of the envisioned communication setup is released in a public repository.[9] In the provided source-code, it is shown how the handling of the CSSO is done before passing the information to the ITS for processing and the dump of the CSSO after the processing.

Future work will focus on testing the adapted system in a real classroom environment to evaluate its effectiveness in actual educational settings and gather feedback for further improvements. Additionally, we plan to develop new metrics for analyzing collaboration, leveraging advanced language models to assess the quality and depth of student interactions. These future developments will help refine the framework's ability to offer meaningful collaborative support and adapt to diverse learning contexts.

The proposed framework offers a scalable and efficient solution for integrating collaborative learning into ITS. It requires minimal modifications to the general architecture of web-based ITS, primarily involving changes to CSSO read/write operations, the addition of a grouping mechanism, and the implementation of message transmission logic. Integration with widely used web frameworks is straightforward, as many popular technologies already offer native support. For example, Java frameworks like Spring Boot provide ready-made solutions, while Python frameworks such as FastAPI and Django, as well as Ruby on Rails, are also compatible, enabling seamless deployment of ITS solutions with collaborative features. The adaptation of HINTS serves as a proof of concept, demonstrating the framework's potential to enhance educational technologies. Future work will focus on optimizing system performance and expanding collaborative learning features, such as adaptive grouping mechanisms and real-time feedback on group dynamics.

CRediT authorship contribution statement

Pablo Arnau-González: Conceptualization, Methodology, Software, Supervision, Validation, Writing – original draft. **Sergi Solera-Monforte:** Data curation, Investigation, Software, Writing – original draft. **Yuyan Wu:** Investigation, Validation, Writing – original draft. **Miguel Arevalillo-Herráez:** Funding acquisition, Supervision, Writing – review.

[9] https://github.com/SPAM-research/its-adaptation-collaborative-demo.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Pablo Arnau-Gonzalez reports financial support was provided by Government of Valencia. Yuyan Wu reports financial support was provided by Government of Valencia. Miguel Arevalillo reports financial support was provided by Spain Ministry of Science and Innovation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been supported by Generalitat Valenciana, Spain through project CIGE/2023/063, grant CIAPOS/2022/163, grant ACIF/2021/439, CIACIF/2023/344, and project CIAICO/2023/065; project TED2021-129485B-C42, funded by MCIN/AEI/10.13039/501100011033 and the European Union "NextGenerationEU"/PRTR; and project PID2023-150960NB-I00, funded by the Spanish Ministry of Science, Innovation and Universities and the European Union.

Data availability

Experimental data available at: https://zenodo.org/records/14163440..

References

Al-Hanjori, M. M., Shaath, M. Z., & Naser, S. S. A. (2017). Learning computer networks using intelligent tutoring system. *International Journal of Advanced Research and Development*, *2*(1).

Albornoz-De Luise, R. S., Arnau-González, P., & Arevalillo-Herráez, M. (2022). On providing natural language support for intelligent tutoring systems. In *International conference on artificial intelligence in education* (pp. 564–568). Springer.

Alshaikh, Z., Tamang, L. J., & Rus, V. (2021). Experiments with auto-generated socratic dialogue for source code understanding. In *CSEDU (2)* (pp. 35–44).

Alshawwa, I. A., Al-Shawwa, M., & Abu-Naser, S. S. (2019). An intelligent tutoring system for learning computer network ccna. *International Journal of Engineering and Information Systems (IJEAIS)*, *3*(2), 28–36.

Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., et al. (2019). Guidelines for human-AI interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems* (pp. 1–13).

Arevalillo-Herráez, M., Arnau, D., & Marco-Giménez, L. (2013). Domain-specific knowledge representation and inference engine for an intelligent tutoring system. *Knowledge-Based Systems*, *49*, 97–105.

Arevalillo-Herráez, M., Ayesh, A., Rezakhanlou, H., & Arnau, D. (2024). Using large language models to integrate virtual students in computerized learning platforms. In *Proceedings of the 2024 IEEE international conference on systems, man, and cybernetics* (pp. 1–2).

Arnau, D., Arevalillo-Herráez, M., Puig, L., & González-Calero, J. A. (2013). Fundamentals of the design and the operation of an intelligent tutoring system for the learning of the arithmetical and algebraic way of solving word problems. *Computers & Education*, *63*, 119–130.

Arnau-González, P., Arevalillo-Herráez, M., Albornoz-De Luise, R., & Arnau, D. (2023). A methodological approach to enable natural language interaction in an intelligent tutoring system. *Computer Speech and Language*, *81*, Article 101516.

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

Brown, J. S., Burton, R. R., & Bell, A. G. (1975). SOPHIE: A step toward creating a reactive learning environment. *International Journal of Man-Machine Studies*, *7*(5), 675–696.

Carbonell, J. R. (1970). *Mixed-initiative man-computer instructional dialogues* (Ph.D. thesis), Massachusets, USA: Massachusetts Institute of Technology. Dept. of Electrical Engineering.

Clancey, W. J. (1979). Tutoring rules for guiding a case method dialogue. *International Journal of Man-Machine Studies*, *11*(1), 25–49.

Collazos, C. A., Guerrero, L. A., Pino, J. A., Renzi, S., Klobas, J., Ortega, M., et al. (2007). Evaluating collaborative learning processes using system-based measurement. *Journal of Educational Technology & Society*, *10*(3), 257–274.

Conejo, R., Guzmán, E., Millán, E., Trella, M., Pérez-De-La-Cruz, J. L., & Ríos, A. (2004). SIETTE: A web-based tool for adaptive testing. *International Journal of Artificial Intelligence in Education*, *14*(1), 29–61.

Conejo, R., Guzmán, E., & Trella, M. (2016). The SIETTE automatic assessment environment. *International Journal of Artificial Intelligence in Education*, *26*, 270–292.

Damasceno, A. R. P., Martins, A. R., Chagas, M. L., Barros, E. M., Maia, P. H. M., & Oliveira, F. C. M. B. (2020). STUART: An intelligent tutoring system for increasing scalability of distance education courses. In *Proceedings of the 19th Brazilian symposium on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery.

Ellis, C. A., & Gibbs, S. J. (1989). Concurrency control in groupware systems. In *Proceedings of the 1989 ACM SIGMOD international conference on management of data* (pp. 399–407). New York, NY, USA: Association for Computing Machinery.

Francisco, R. E., & de Oliveira Silva, F. (2022). Intelligent tutoring system for computer science education and the use of artificial intelligence: A literature review. In *CSEDU (1)* (pp. 338–345). SciTePress.

Ghavifekr, S. (2020). Collaborative learning: A key to enhance students'social interaction skills. *MOJES: Malaysian Online Journal of Educational Sciences*, *8*(4), 9–21.

Guo, L., Wang, D., Gu, F., Li, Y., Wang, Y., & Zhou, R. (2021). Evolution and trends in intelligent tutoring systems research: a multidisciplinary and scientometric view. *Asia Pacific Education Review*, *22*(3), 441–461.

Haq, I. U., Anwar, A., Basharat, I., & Sultan, K. (2020). Intelligent tutoring supported collaborative learning (itscl): a hybrid framework. *International Journal of Advanced Computer Science and Applications*, *11*(8).

Hooshyar, D., Ahmad, R. B., Yousefi, M., Fathi, M., Horng, S.-J., & Lim, H. (2018). SITS: A solution-based intelligent tutoring system for students' acquisition of problem-solving skills in computer programming. *Innovations in Education and Teaching International*, *55*(3), 325–335.

Jaques, P. A., Seffrin, H., Rubi, G., de Morais, F., Ghilardi, C., Bittencourt, I. I., et al. (2013). Rule-based expert systems to support step-by-step guidance in algebraic problem solving: The case of the tutor PAT2Math. *Expert Systems with Applications*, *40*(14), 5456–5465.

Kulik, J., & Fletcher, J. D. (2015). Effectiveness of intelligent tutoring systems: A meta-analytic review. *Review of Educational Research*, *86*.

Laal, M., & Ghodsi, S. M. (2012). Benefits of collaborative learning. *Procedia - Social and Behavioral Sciences*, *31*, 486–490, World Conference on Learning, Teaching & Administration - 2011.

Lahmadi, Y., Ouadoud, O., El Khattabi, Z. M., Rahhali, M., & Oughdir, L. (2024). Developing a collaborative learning environment for ITS: A new model based on IMS-learning design. *Statistics, Optimization & Information Computing*.

Lanzilotti, R., & Roselli, T. (2007). An experimental evaluation of logiocando, an intelligent tutoring hypermedia system. *International Journal of Artificial Intelligence in Education*, *17*(1), 41–56.

Lippert, A., Shubeck, K., Morgan, B., Hampton, A., & Graesser, A. (2020). Multiple agent designs in conversational intelligent tutoring systems. *Technology, Knowledge and Learning*, *25*, 443–463.

Lowyck, J., & Pöysä, J. (2001). Design of collaborative learning environments. *Computers in Human Behavior*, *17*(5–6), 507–516.

Lu, Y., Pian, Y., Chen, P., Meng, Q., & Cao, Y. (2021). Radarmath: An intelligent tutoring system for math education. In *Proceedings of the AAAI conference on artificial intelligence, vol. 35* (pp. 16087–16090).

Mahmoud, M. H. (2018). A multiagents based intelligent tutoring system for teaching arabic grammar. *International Journal of Education and Learning Systems*, *3*.

Major, C. (2020). Collaborative learning: A tried and true active learning method for the college classroom. *New Directions for Teaching and Learning*, *2020*(164), 19–28.

Miller, R. B. (1968). Response time in man-computer conversational transactions. In *AFIPS '68 (fall, part i), Proceedings of the December 9-11, 1968, fall joint computer conference, part i* (pp. 267–277). New York, NY, USA: Association for Computing Machinery.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: The constraint-based approach. *IEEE Intelligent Systems*, *22*, 38–45.

Mousavinasab, E., Zarifsanaiey, N., R. Niakan Kalhori, S., Rakhshan, M., Keikha, L., & Ghazi Saeedi, M. (2021). Intelligent tutoring systems: a systematic review of characteristics, applications, and evaluation methods. *Interactive Learning Environments*, *29*(1), 142–163.

Nguyen, H. D., Tran, D. A., Do, H. P., & Pham, V. T. (2020). Design an intelligent system to automatically tutor the method for solving problems. *International Journal of Integrated Engineering*, *12*(7), 211–223.

Nhan, H., & Anh Nhan, T. (2019). Different grouping strategies for cooperative learning in english majored seniors and juniors at Can Tho University, Vietnam. *Education Science*, *9*, 59–75.

Nye, B., Graesser, A., & Hu, X. (2014). AutoTutor and family: A review of 17 years of natural language tutoring. *International Journal of Artificial Intelligence in Education*, *24*(4), 427–469.

Olsen, J. K., Belenky, D. M., Aleven, V., & Rummel, N. (2013). Intelligent tutoring systems for collaborative learning: Enhancements to authoring tools. In *Artificial intelligence in education: 16th international conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. proceedings 16* (pp. 900–903). Springer.

Olsen, J. K., Belenky, D. M., Aleven, V., & Rummel, N. (2014a). Using an intelligent tutoring system to support collaborative as well as individual learning. In *Intelligent tutoring systems: 12th international conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. proceedings 12* (pp. 134–143). Springer.

Olsen, J. K., Belenky, D. M., Aleven, V., & Rummel, N. (2014b). Using an intelligent tutoring system to support collaborative as well as individual learning. In *Intelligent tutoring systems: 12th international conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. proceedings 12* (pp. 134–143). Springer.

Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J., & Ringenberg, M. (2013). Authoring collaborative intelligent tutoring systems. *Grantee Submission*.

Olsen, J. K., Belenky, D. M., Aleven, V., Rummel, N., Sewall, J., & Ringenberg, M. (2014). Authoring tools for collaborative intelligent tutoring system environments. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Intelligent tutoring systems* (pp. 523–528). Cham: Springer International Publishing.

Price, T. W., Dong, Y., & Lipovac, D. (2017). Isnap: Towards intelligent tutoring in novice programming environments. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 483–488). New York, NY, USA: Association for Computing Machinery.

Shneiderman, B. (1984). Response time and display rate in human performance with computers. *ACM Computing Surveys*, *16*(3), 265–285.

Solera-Monforte, S., Arnau-González, P., & Arevalillo-Herráez, M. (2024). Be my mate: Simulating virtual students for collaboration using large language models. In S. Mahamood, N. L. Minh, & D. Ippolito (Eds.), *Proceedings of the 17th international natural language generation conference: system demonstrations* (pp. 1–3). Tokyo, Japan: Association for Computational Linguistics.

Stahl, G. (2013). A model of collaborative knowledge-building. In *International conference of the learning sciences* (pp. 70–77). Psychology Press.

Stevens, A. L., & Collins, A. (1977). The goal structure of a socratic tutor. In *Proceedings of the 1977 annual conference* (pp. 256–263). New York, NY, USA: Association for Computing Machinery.

Suebnukarn, S., & Haddawy, P. (2004). A collaborative intelligent tutoring system for medical problem-based learning. In *Proceedings of the 9th international conference on intelligent user interfaces* (pp. 14–21). New York, NY, USA: Association for Computing Machinery.

Suebnukarn, S., & Haddawy, P. (2006a). A Bayesian approach to generating tutorial hints in a collaborative medical problem-based learning system. *Artificial Intelligence in Medicine*, *38*(1), 5–24.

Suebnukarn, S., & Haddawy, P. (2006b). Modeling individual and collaborative problem solving in medical problem-based learning. *User Modeling and User-Adapted Interaction*, *16*, 211–248.

Ubani, S., & Nielsen, R. (2022). Review of collaborative intelligent tutoring systems (CITS) 2009–2021. In *2022 11th international conference on educational and information technology* (pp. 67–75). IEEE.

VanLehn, K., Banerjee, C., Milner, F., & Wetzel, J. (2020). Teaching algebraic model construction: A tutoring system, lessons learned and an evaluation. *International Journal of Artificial Intelligence in Education*, *30*(3), 459–480.

Walker, E., Rummel, N., & Koedinger, K. R. (2014). Adaptive intelligent support to improve peer tutoring in algebra. *International Journal of Artificial Intelligence in Education*, *24*, 33–61.

Walker, E., Rummel, N., McLaren, B. M., & Koedinger, K. R. (2007). The student becomes the master: integrating peer tutoring with cognitive tutoring. In *Proceedings of the 8th iternational conference on computer supported collaborative learning* (pp. 751–753). International Society of the Learning Sciences.

Ward, W., Cole, R., Bolaños, D., Buchenroth-Martin, C., Svirsky, E., & Weston, T. (2013). My science tutor: A conversational multimedia virtual tutor. *Journal of Educational Psychology*, *105*(4), 1115–1125.