International Journal of Science Academic Research
Vol. 07, Issue 10, pp.1568-1602, October 2026
Available online at http://www.scienceijsar.com
ISSN: 2582-6425

Research Article

AN INTELLIGENT TUTORING SYSTEM AUTHORING TOOL, CHECKER, AND PLATFORM DEVELOPMENT THAT REPRODUCES THE CLASSROOM LECTURE STYLES OF HUMAN TEACHER

[1]Gwang-Hyok Kim, [2]Yu-Il Kim [3,*]Song-Hwan Kwon, [4]Gyong-Jin Kim, [5]Gang-Bom So, [3]Sin-Hye Jang, [3]Ha-Gyong Kim, [3]Jong-Nam Rim, [3]Chung-Song Ko, [3]Song-Il Pak, [3]Chol-Jun Hong and [6]Myong-Son Choe

[1]Institute of Satellite Information Technology, State Academy of Sciences, Pyongyang, Democratic People's Republic of Korea
[2]Information Technology Department, Sinuiju College of Industrial Technology, Sinuiju, Democratic People's Republic of Korea
[3]Artificial Intelligence Faculty, University of Science, Pyongyang, Democratic People's Republic of Korea
[4]Unjong Science Region Postgraduate School, Pyongyang, Democratic People's Republic of Korea
[5]Information Department, Sinuiju Honoured Ex-servicemen College of Economics, Sinuiju, Democratic People's Republic of Korea
[6]Institute of Information Technology,University of Science, Pyongyang, Democratic People's Republic of Korea

Received 12thAugust 2026; **Accepted** 15thSeptember 2026; **Published online** 30thOctober 2026

Abstract

This paper describes a method for developing intelligent tutoring systems (ITSs) that reproduce the classroom teaching format as taught by human instructors. We have already published monographs on the intelligent tutoring system framework and platform development. In those monographs, we propose a hierarchical graph partitioning method in a subject to facilitate graph construction and prevent inconsistencies in graph construction, which is represented as an extension of the knowledge graph. However, there were areas that were not developed in depth in the implementation using Hidden Markov Models (HMMs) and Markov Decision Process (MDP). This paper (book) will delve deeply into the problems raised in developing intelligent tutoring systems that replicate the classroom teaching format as taught by human instructors. What it means to reproduce the classroom teaching format as taught by human instructors is not about gathering multiple students in a classroom for uniform instruction, but rather replicating the teaching format exactly. This includes reproducing elements such as the display of picture data on TV, the associated video playback, the interpretation using human teacher's voice about the picture data displayed on TV, the question and response. Our intelligent tutoring system consists of an intelligent tutoring content authoring tool, a checker (or player) for the authored intelligent tutoring content, and a platform. Therefore, we propose the development of an authoring tools, a checker (or player) development, and a platform development method. We also propose a pedagogical module development approach for three tutoring models, such as the one-to-one tutoring model, the tutoring model using the knowledge graph, and the tutoring model using Partially Observable Markov Decision Process (POMDP). In this paper, we propose an acyclic violation checking algorithm that can occur if we construct a hierarchical partition-type knowledge graph using a graph drawing tool and a way to hierarchically partition the knowledge graph. Automatic construction of knowledge graphs requires a vast amount of educational digital resources. We also propose a method to manually construct hierarchically partitioned knowledge graphs using visualized graph drawing tools included in knowledge graph authoring tools, if there is no or lack of extensive pedagogical digital data.

Keywords: Intelligent Tutoring Systems, Step-by-Step Tutoring Model, Knowledge Graphs, POMDPs, Hierarchical Partitioning of Knowledge Graphs.

1. INTRODUCTION

Today, intelligent tutoring systems (ITSs) are increasingly effective due to the development of artificial intelligence, the development of Large Language Models, and the construction and use of knowledge graphs. Many good methods, principles and models for the intelligent tutoring systems have been proposed, known, shared, and countless papers have been published.Intelligent tutoring systems can be considered in terms of development and usage.Development is a matter of consideration to universities, organizations, and companies. These developments include framework and platform development. Frameworks include authoring tools and checkers (or players). Authoring tools can be developed for the web or for personal computer (or desktop) applications that can be downloaded and used on site. When developing or supplying the authoring tools needed by a specialist in the field, a player or checker that reproduces or checkers the results of the production must be developed and supplied simultaneously. Next, the platform must be developed.

Corresponding Author:** ***Song-Hwan Kwon,
Artificial Intelligence Faculty, University of Science, Pyongyang, Democratic People's Republic of Korea

The platform should load the results created by experts or teachers in the field into its system (web servers or application of personal computers) and then interact with the student to assist the learner in his learning activities.The use of intelligent tutoring systems is relevant to two groups of people. One group of people, as typical learners, perform their learning activities with appropriate intelligent tutoring systems. This kind of people only targets the platform. Another group of people is experts and teachers in the subject or field, belonging to the middle-level developer in the field of intelligent tutoring systems. This group of people uses the framework, authoring tools and checkers (players). Teachers or professionals in the field use authoring tools to create domain models, basic data of student model, and basic data of pedagogical model for the topic. Also, we use a player for delivery and a checker for improving the performance of the product.In this regard, we can classify the intelligent tutoring system as follows. One is an intelligent tutoring system (called embedded in the future) where the platform embeds the product of the authoring tool, and the other is an intelligent tutoring system (semi-embedded) that simultaneously provides the authoring tool and platform. Another category is the so-called open source, script (or SDK), and plug in types. The Open Source is a system that provides a

fully open source of the framework and platform of the intelligent tutoring system. Script (or SDK) type is a system that can be used to modify or add some of the functions of the system provided in its own way. The plug-in is a system that can use some of the functions associated with the intelligent tutoring provided in conjunction with their intelligent tutoring system.

When developing or using intelligent tutoring systems in a country, region or universities, the following problems can be encountered:

a. Will you still use the embedded system that is already being supplied worldwide?
b. Will you use a semi-embedded version?
c. Will you use an open source version?
d. Will you use a script type (or SDK type)?
e. Will you use the plug-in type?
f. Using the methods and principles already proposed in relation to the intelligent tutoring system, will we develop ourselves entirely?

Both options have advantages and disadvantages. Our paper will describe the challenges of developing an intelligent tutoring system fully on its own. In particular, we will describe the challenges in developing a tutoring framework and platform that reproduces the classroom teaching style of human teachers.

The development of intelligent tutoring systems, especially the development of authoring tools and checkers as frameworks, and the development of platforms, are interconnected from the design stage to the development. Thus, the design and development of authoring tools should consider the requirements of the platform, and the design and development of the platform should consider the design of authoring tools.The development of the intelligent tutoring system authoring tool should consider experts and teachers in the field who use the authoring tool and define the skills they need to do. The checker or player must import the generated authoring results to check whether they work properly or not, reproducing the interaction with the student. Of course, the authoring results can contain different kinds of files and a database that defines the relationships between those files. Or the result of authoring may be encoded. The functionality and role of the platform are almost similar to the checker. In the case of authoring tools, domain models (i.e., domain expert knowledge) should be created for the domain. In addition, if the type of intelligent tutoring system to be built is knowledge-tracking and adaptive learning-based, it should be possible to create some of the student models to track the learner's knowledge state and also part of the pedagogical model for learning path generation. The development of checkers and platforms requires the development of student models that can predict and estimate learning situations, and UI design and development for teaching models, interactions, and so on.

The paper will describe the specifications, functional diagrams, system design diagrams, and system flow diagrams of the developed intelligent tutoring system.The proposed intelligent tutoring system works based on knowledge graphs with a single relation type or directed acyclic graphs. Knowledge graphs we consider is a hierarchical partitioned knowledge graph constructed in a hierarchical fashion. For experts and teachers in the field to use such graphs, and to enable the creation of such graphs in authoring tools, principles and approaches of partitioning the knowledge graph into hierarchical forms must be proposed. Also, when aiming at visualization of graph creation, problems in the design and development of graph drawing tools must be solved, especially when acyclic violation checking algorithms occurring during graph creation must be proposed.

If we want to use POMDP for learner learning state tracking and learning path generation, the problem of converting the constructed knowledge graph into POMDP format either in an automatic or semi-automatic manner has to be solved. Also, a machine learning approach to POMDP parameters is established and its function should be implemented. In addition, approaches to linearly reduce the exponential amount of computation in POMDP and a method of invalid state extraction should also be proposed.The learning activities of the intelligent tutoring system considered in this paper are driven by step-by-step intelligent tutoring contents, tutoring contents using only the knowledge graph, and tutoring contents using POMDP. We also propose system design and implementation approaches for these tutoring contents.The intelligent tutoring system we consider is a system that reproduces the classroom instruction style of human teacher. Therefore, we will develop a paper to suit this characteristic.

2. INTELLIGENT TUTORING SYSTEMS (ITSs)

The paper will address, review and summarize the following research questions about intelligent tutoring systems.

RQ 1: What are the Intelligent Tutoring Systems?

Intelligent tutoring systems are computer-based learning environments that provide adaptive, personalized tutoring and immediate feedback, and also mimic one-to-one human tutors. Intelligent tutoring systems have a research history of about 50 years, starting from the study of the SCHOLAR system (Carbonell[18]) in the 1970s, and similar systems have been continuously studied [19-22].In general, intelligent tutoring systems consist of four components: a domain model (expert model), a student model (user profile) that tracks learners' learning and cognition, a pedagogical model (tutoring model) that relates to pedagogical strategies and path guidance and a user interface [62-76]. In the following Figure, we use the term module instead of model in the sense that we implement the model.

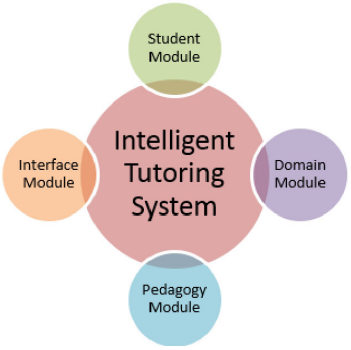


Figure 1. The Components of ITSs [63]

The domain model is a model that includes knowledge, concepts, and rules about the current learning topic, and the student model is a model that tracks the learner's cognitive state, proficiency, and incorrect views by analyzing the interaction and progress between the student and the system.

The pedagogy model determines the instructional strategy when to give a hint, give an explanation, inform a new one, and present challenging problems, and guides the learning path. The user interface provides a friendly environment for the student and teacher and allows them to interact [49].

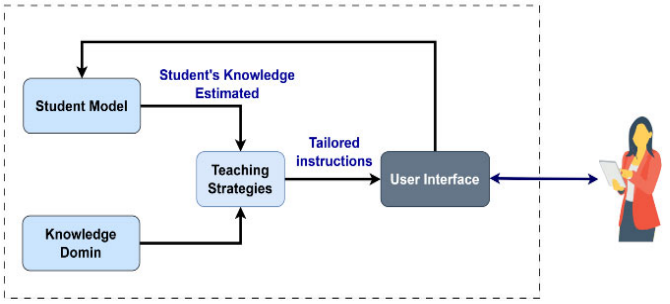


Figure 2.The General Flow Diagram of ITSs [50]

RQ 2: What are the types of intelligent tutoring systems and what are the typical examples of such types?

The first intelligent tutoring system in the 20th century was the Intelligent Tutoring Tools (ITTs) developed by the Byzantium research project, which included six universities. This intelligent tutoring system has received positive appraisal from many research institutions that use it as a general tutoring system builder [6]. Using this constructor, you can create an Intelligent Tutoring Applet (ITA). Using this builder, many teachers created ITAs on different topics [6, 23].

- Conversational Chatbots/Dialogue Tutors

These systems use natural language processing to assist students in their learning through pedagogical conversation, as human tutors and guide students' learning through suggestion and motivational language, rather than providing immediate answers to students. The Sghaier [17] reports that the large language model is promoting new developments in higher education, especially in personalized and customized education.

AutorTutor: Through a Socratic style of conversation, it helps learners learn about a topic.
Khanmigo (by Khan Academy): It helps students to learn mathematics and science.
Mizou: It enables teachers to create tailored chatbots that support a specific curriculum.

- Cognitive Tutors (Model-Tracing Tutors)

These systems model the cognitive processes involved in problem solving. These systems model the learner's step-by-step learning approach to a problem and implement immediate feedback in case of errors.
MATHia (by Carnegie Learning): Proceeding structured and personalized mathematics.
Andes Physics Tutor: used in education to help solve physical problems.
ESC101-ITS: This system was developed by IIT Kanpur to assist in introductory programming.

- Adaptive Learning Platforms (Knowledge-Tracing Tutors)

These systems model the student's knowledge state. This allows the learner to track the knowledge concept that he has learned perfectly successfully, dynamically fine-tune the complexity of learning, and adjust the learning path to suit the student.

DreamBox Learning: Assists in student mathematics learning in real time.

There are also ASSISTments, which provide immediate feedback in solving mathematical problems, and ActiveMath, which is an adaptive mathematics tutoring based on the Web.

- Intelligent Game-based and Simulation Tutors

These systems use a simulated or game environment for interactive training. It is mainly used in training or program development with technical difficulties.
ST Math: Use visual quiz problems for mathematics instruction.

Flight Simulators: It is used to support training in flight simulation environments.

- Multimodal/Specialized Tutors

These systems use audio, handwriting, and visual signals rather than keyboards for student performance [50].

Amira Learning: It can hear students' reading and provides real-time guidance.

ELSA Speak: We use the system with AI technology to improve the pronunciation of students.
Squirrel AI: This system uses student-centered dialogue data when performing a speech dialogue.

- Collaborative Intelligent Tutoring Systems (CITS)

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.

RQ 3: What are the major intelligent tutoring systems and platforms?

The main intelligent tutoring systems and platforms are as follows.

The system Carnegie Learning is known to use cognitive science to conduct high-level mathematics instructing.

McGraw Hill's Assesstment and LEarning in Knowledge Spaces (ALEKS) is an online tutoring system and evaluation program that includes mathematics subject, chemistry, statistics, and commercial subject-related course resources [3]. ALEKS is a web-based platform that uses adaptive queries to determine the knowledge status of a student in particular.

Technological Pedagogical Content Knowledge (TPACK) is an AI-driven intelligent tutoring system that supports personalized mathematics learning based on the content knowledge framework. TPACK uses reinforcement learning and natural language processing (NLP) to provide context-sensitive feedback, interactive simulation environment [4].

DreamBox Learning is an adaptive math tutoring platform that fine-tunes the query in real time.

ST Math uses spatiotemporal and visual quizzes problems without resorting to text to teach mathematical concepts.

ASSISTments is a platform for helping teachers in a way that provides appropriate feedback to students' mathematics learning progress.

ASSISTments Dataset 2009-2010 includes student interaction logs of students who were exposed to the ASSISTments online teaching platform during 2009-2010 [5]. This dataset is the most widely used benchmarks dataset in Educational Data Mining and the Intelligent Tutoring Community, which provides a very rich foundation for student learning history modeling and performance prediction. Each row of this dataset contains detailed interactive attribute information such as student identifiers, problem numbers (IDs), specific skills (knowledge components) names, success rates (hits/misses), and response sequences over a period of time. The features of this dataset are structured to support complex modeling tasks such as Bayesian Knowledge Tracing (BKT), Deep Knowledge Tracing (DKT), and Dynamic Bayesian Networks (DBN). Researchesuses these data to develop and verify adaptive learning algorithms, analyze the impact of multiple technique matching, and explore the knowledge perceived over time.

Squirrel AI is known as a system that focuses on advanced algorithms for adaptive learning, especially in large-scale environments.

Duolingo uses AI-driven language learning exercises that provide personalized instruction and instant feedback. The Duolingo Intelligent Tutoring System uses artificial intelligence and machine learning algorithms to model human tutors and adapt to students' speed and language levels. Duolingo as an intelligent tutoring system has the following functions: adaptive learning, modeling, and instant feedback. Adaptive learning functions analyze learners' learning performance in real time, such as fine-tuning the complexity of the lesson and discovering the vulnerability. In the modeling function, we create an individual model for each student based on each individual's history to identify the exercises needed for each learner. Duolingo supports student learning through AI-driven dialogue and interaction, and provides immediate feedback when learning errors occur.

Cognitive Tutor Authoring Tools (CTAT) is known as a basic framework that uses example-tracing and cognitive modeling for tutoring. This system works based on a model of how students acquire and apply knowledge. These systems use a step-by-step instructional principle and provide feedback control to the learner. Cognitive Tutor (CTAT) allows pedagogical researchers with limited programming experience to create (cognitive) tutors [16].

SmartTutor is a system developed by the University of Hong Kong University (HKU), which integrates Internet technology, pedagogical research and artificial intelligence to support student learning.

AutoTutor simulates the teaching style and teaching strategy of human teachers to assist college students in learning computer hardware, operating systems, and Internet subjects. AutoTutor understands the learner's input through the keyboard and then performs interactive actions such as feedback, guidance, feed forward, and hints.

In particular, the intelligent authoring tools include ASPIRE, the Cognitive Tutor Authoring Tools (CTAT), GIFT, ASSISTments Builder, and AutoTutor tools [14].

There are also intelligent tutoring systems such as Algebra Tutor PAT, SQL-Tutor, EER-Tutor, COLLECT-UML, StoichTutor, Mathematics Tutor, eTeacher, ZOSMAT, REALP, CIRCSIM-Tutor, Why2-Atals, ActiveMath, ESC101-ITS, and ErAdaptrEx[15].

ACTIVEMATH is a generic web-based learning system that dynamically generates interactive (mathematical) courses adapted to the student's goals, preferences, capabilities, and knowledge [26].

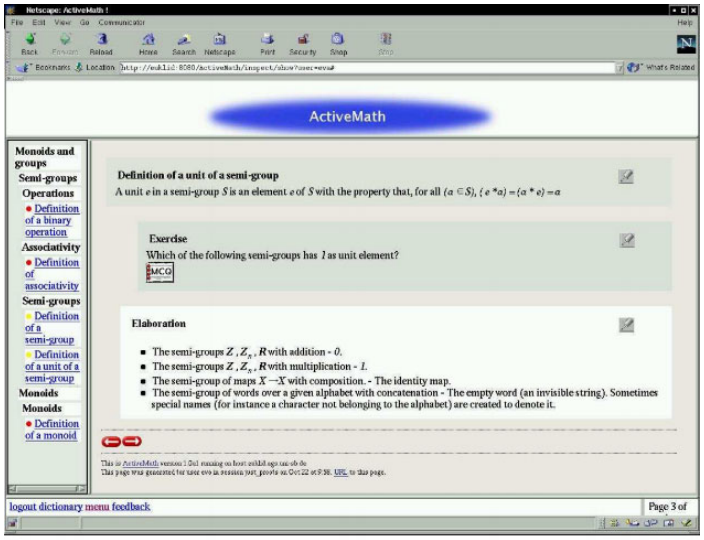


Figure 3.A Screenshot of an ActiveMathSession [27]

RQ 4: What are the main characteristics of typical intelligent tutoring systems?

The main characteristics of typical intelligent tutoring systems are as follows.

As personalized learning, each student applies different teaching strategies, examples, and practical exercises. There is student modeling, which has detailed history of student expertise, gaps in knowledge, and misunderstandings. There is an instant feedback function, which provides real-time feedback, an error description, scaffolding for problem-solving and resource presentation. Active learning is not motivated by information obtained passively but by motivating students to solve problems in a more active and active stance. Intelligent tutoring systems have an expert model of the subject to guide and assist students' learning efficiently, which is called domain knowledge.

RQ 5: What do MOOCs look particularly like?

MOOCs (Massive Open Online Courses) are free, low-cost, web-based, and educational-related curriculum datasets, with a very large number of participants, sometimes involving many universities, companies, and organizations, to create curricula. The characteristics of MOOCs include lecture videos, interactive quiz games, and conference results.

RQ 6: What is the general architecture of the intelligent tutoring system?

The general architecture of intelligent tutoring system is as follows. By a consensus of many researchers (Nwana, Freedman, Nkambou et al.) the basic components of the intelligent tutoring system were identified as follows [8-10]: Domain models, student models, pedagogical models, and user interface.

The domain model consists of expert knowledge, facts, rules, and problem-solving strategies for specific topics. The domain model is divided into two domains. one is the content domain and the other is the academic performance evaluation domain. The content domain includes the curriculum, subject knowledge and structured knowledge. The academic performance assessment domain should have models that are ready to solve the problems that the system faces, to detect and diagnose errors in the student's learning process in time, and to evaluate the learner's performance. The domain model plays the role of "what" in the intelligent tutoring system. A domain model, also known as cognitive model or expert knowledge model, can be constructed using either of the two theory of learning. One learning theory is ACT-R, which considers all possible steps required to solve the problem. The model built on the basis of this theory includes concepts, rules, and problem-solving strategies of the domain that we want to learn [10]. This model is a source of expert knowledge, acts as a standard for evaluating student achievement and performs error detection[10]. Another theory is StellanOhlsson's theory, known as constraint-based modeling (CBM), in which the domain model is represented as a set of constraints for exact solutions [11-12].The student model is used to assess how much improved the cognitive status and academic achievement by the student's learning process. The student model includes diagnostic information and learning progress tracking. Diagnostic information includes time spent on task execution, types of errors, patterns associated with assistance, and knowledge that judges that a concept is completely mastered. In learning progress tracking, the system can be used to update in real time information about student achievement, inferring the advantages and disadvantages of student learning. The student model plays the role of "who" in the intelligent tutoring system. In intelligent tutoring systems, the process of student modeling is called model tracking. In constraint-based intelligent tutoring systems, negative feedback (error-based feedback) or positive feedback is performed based on the results of the student model.

The main components of the tutoring activity include the introductory, the representation of the tutoring content, the presentation of practical problems, and the application. The pedagogical model receives the results from the domain model and the student model and decides which instructional strategy or action to take. The pedagogical model plays a "how" role in the intelligent tutoring system. The pedagogical model includes instructional strategies, learning path selection functions, and adaptability. The instructional strategies include suggestions, feedback, explanations, and worked examples. Learning path selection determines a sequence of topics and learning contents that meet the student's needs and the learning objectives. Adaptive ability selects the optimal instructional strategy from a library that allows the student to succeed. In the problem-solving process-based intelligent tutoring systems, learners hope to guide what to learn next in their current learning situation. Also, intelligent tutoring systems provide feedback to enable the learner to reach the learning goal at a reasonable time if he/she makes mistakes [13]. Using knowledge tracking techniques, intelligent tutoring systems track the student's learning process and preserve its history. For example, the cognitive tutoring system developed by Carnegie Mellon University represents knowledge-tracking information called skillometer, which represents the degree of learner's cognitive success for each expertise involved in solving algebraic problems by visual graphs. When a student requests assistance such as a hint or a mistake occurs, the knowledge tracking data and skillometer are updated automatically and in real time [5-7].

User interface models enable interaction and communication with students and intelligent tutoring systems. An interactive and a visual environment are necessary. In interactive environments, there are presentation of training data, query presentation, and receipt of learner input.

Tommaso [24] proposed an easy and fast way to develop an interface for intelligent tutoring authoring tools using Large Language Models and prompt engines. This tool allows you to develop an authoring tool without programming.

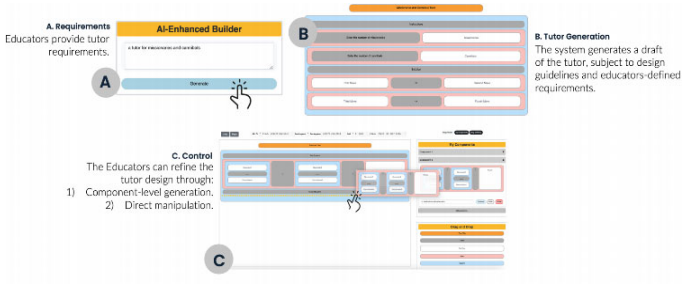


Figure 4.Illustration of Our AI-Enhanced Tutor Builder System: Educators Provide Input Requirements (A) which Inform the Automated Generation of a Draft Tutor Interface (B), Followed by the Educator's Hands-on Refinement through Component Generation and Direct Manipulation (C), Visualizing the Integration of Generative Design and Educator-Driven Customization[24]

In the paper from Priynka[25], a method for designing intelligent tutoring systems that helps students in online programming is proposed. In the following, the proposed methods are presented through diagrams and algorithms.

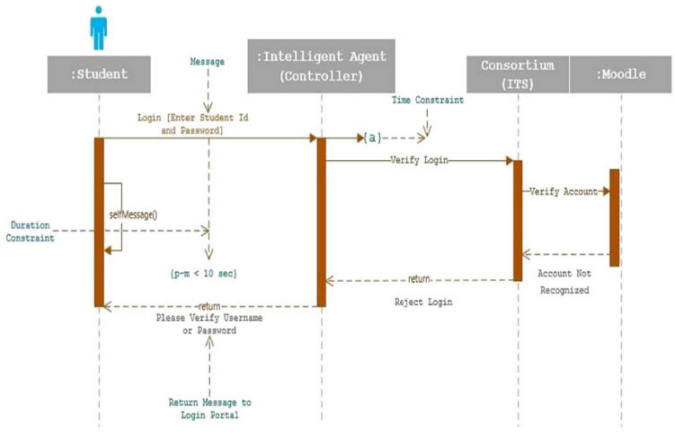


Figure 5. Shows the Sequence Diagram that Describes a Login Interaction among a Set of Objects Participated in for Example (Student, Intelligent Agent, Consortium, and Moodle) that has a Collaboration (or Scenario on Intelligent Tutoring), Arranged in Chronological Order; It Shows the Objects Participating in the Interaction[25]

[Algorithm 1. Case-based Reasoning for Intelligent Tutoring System (ITS)] (from [25])

for each request until max-problem do

Step 1: Determine the weights of variables from Problem Request
i. wt = {wt1, wt2, wt3, wt(m-1), wtm}
if (PR is true) and (C is Same) then
Step 2: Calculation of local similarities for each Old Problem
ii. S (wtN, wtp) = 1
else
Step 3: Calculation of global similarities for each Old Problem
iii. S (CN, CPs) = ∑ wt (si (CN, CPs))
Step 4: Best solution resulting global similarities
Step 5: Retrieve the solution
Step 6: Present solution
end

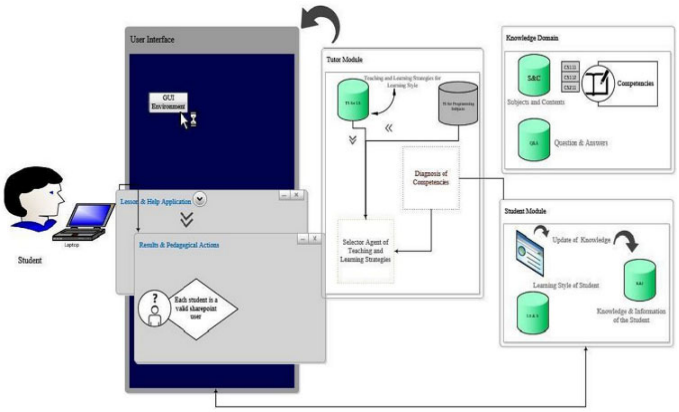


Figure 6. Shows the Proposed Architecture for ITS. Since the Programming Subjects have been Chosen and Customized, they will be Made Available to the Student through the Moodle Platform. The Assessment will be Applied after the Student has Revised a Subject, and the Database that Stores the Student's Output will be Updated with the Results of the Evaluation[26]

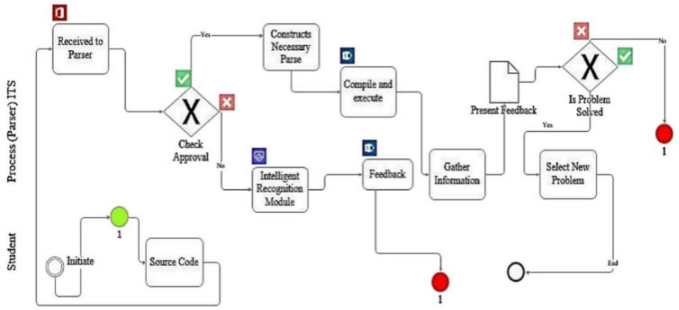


Figure 7. The Flow Chart Process Model for the Proposed ITS. Due to the Complexities of Semantic Decoding, ITS can only Mentor a Limited Subset of Programming Languages Concurrently. Therefore, at some Point, It Necessitates Making Restrictions on a Small Subset[25]

In addition to the four models mentioned above, two additional functions exist. One is the authoring tools and the other is the knowledge management units. The authoring tools allow the expert to add and update the instructional content and rules. Another complementary function is the expert system, which uses rules to make the necessary decisions during the instructional activities.

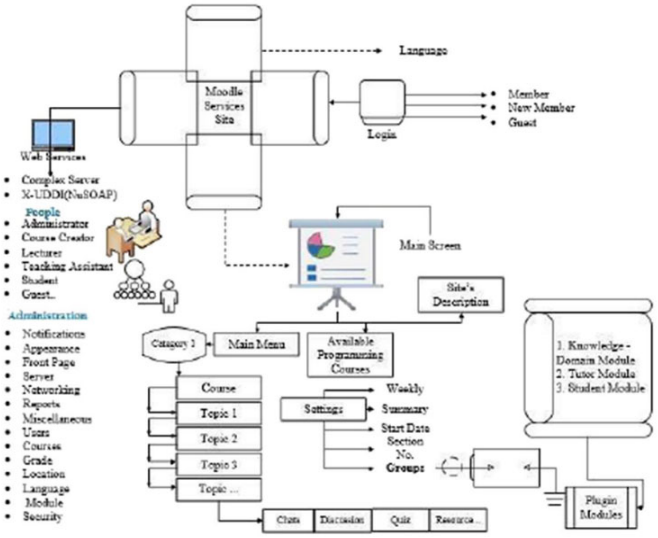


Figure 8. The Broad-Spectrum Architecture of Moodle and its Functions and how the ITS System will be Embedded into this LMS[25]

In paper from MUHAMMAD [27], a variety of ITSs and ATSs using different methods are reviewed.

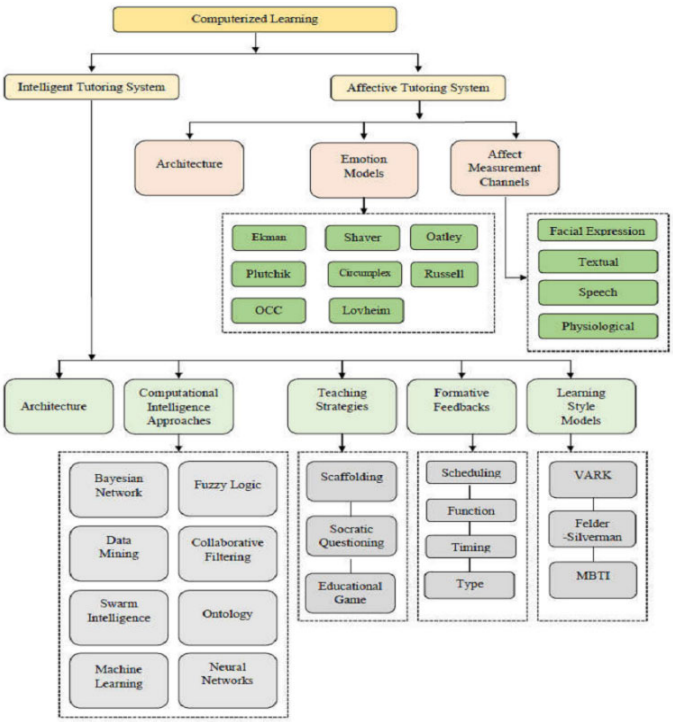


Figure 9. State-of-the-Art Approaches for Devising ITS and ATS[27]

RQ 7: What are the methods of designing and developing intelligent tutoring systems?

Corbett [7] summarized that the design and development of an intelligent tutoring system is done by repeatedly applying the following four steps: Needs assessment, cognitive task analysis, development of the initial version of tutoring systems, and evaluation.

The first step, the needs assessment, is the instructional design process related to software development. This step includes learner analysis and consultation with experts in the field. This step involves designing the student domain and the knowledge domain. This design step should consider the probability that a

student will solve a problem, the time to reach the desired benchmark, and the ability to address the probability that a student will apply that expertise in practice in the future. Next, the interface should be well designed to enhance the effect of interaction [13].

The next step, the cognitive task analysis, is to conduct consultation with domain experts and to analyze and design how to observe the tutoring and learning activities.
Next, there are development step for a first version of system and evaluation step for the constructed system.

RQ 8: What are the types of UI design of intelligent tutoring systems?

The UI design styles of intelligent tutoring systems are quite diverse. These interfaces are designed to suit the instructional strategy used, such as from constraint-based rules to dialogue interaction strategies. The main design styles are as follows. There is dialogue interface, adaptive and dynamic interface, gamified interface, model-tracking (problem-solving) interface, multi-modal interface, and data-driven interface.

- Conversational/dialogue-based interface

This approach uses natural language processing based on a powerful Large Language Model to perform human teacher-like activities in a dialogic manner. Important features include Chatbot interface, tailored response, and context understanding.

- Adaptive/dynamic interface

In this form, content representation and their placement vary according to the student's behavior, learning speed and cognitive status. The main features are personalized navigation, adaptation of content to the level of expertise, and real-time customization of the interface environment. This type of design aims to minimize the burden and unnecessary consumption of learning by displaying the necessary information regarding the tendency of the individual learner's learning history.

- Gamified interface

This format integrates game-style design elements to enable more active learning activities with a higher interest in learning. The main features are points, leaderboards, achievements, and visual effects (e.g., graphics or maps representation of learning progress) such as games.

- Model-tracing/problem-solving interface

These interfaces allow for design, considering all the critical instructional strategies, the learner's mental learning activities against the expert model. The main features are step-by-step guidance, instant feedback, and structured problem- solving domain. An example is Cognitive Tutor. It is now considered appropriate to apply such a STEM (mathematics, physics, computer science) education.

- Multimodal/immersive interface

This format is known as a very advanced technology that goes beyond the effects of traditional graphical user interfaces (GUIs) by including such interactions with 3D environments or audio. The main features are virtual reality (VR), augmented reality (AR), and voice user interfaces (VUI). It is suitable for medical practice, simulation-based combat training, hand-held technical upgrading, etc. [50].

- Minimalist and data-driven interfaces

This format uses data visualization to show the student's performance. The main features are grid-based structure, hierarchical visual data, etc.

RQ 9: What relationship are the knowledge graphs and the intelligent tutoring system?

The knowledge graphs and intelligent tutoring systems are integrated to develop intelligent, more individualized and context-oriented educational platform. While the intelligent tutoring system performs individualized learning guidance, diagnoses learning situations and errors, and makes appropriate teaching, the knowledge graph plays the role of a structured knowledge base that provides the necessary context, semantic relationships, and educational structure to the AI. A knowledge graph acts as a domain model in intelligent tutoring system [29]. Researchers with the same view as Alan Turing have the view that knowledge graphs play a bridge between human reasoning and machine processing. The relationship between these two in tutoring is discussed in detail as follows.

a. Structuring the learning domain

The knowledge graphs organize educational-related data (concepts, expertise, topics) into a network of nodes (entities) and edges (relationships). Such relationships can be either "prerequisite" or "part of" or so on.

b. Providing personalized learning paths

The knowledge graphs allow an ITS to match the learning progress of individual learners to the knowledge graph and perform tailored learning. As such an example, there is the identification of learning errors (or weakness identification). That is, when an ITS identifies that a student has a very difficult learning of a concept (e.g., informed search), it can perform a backward search to find the essential prior concept (e.g., uninformed search) of that concept through a knowledge graph. Another example of tailored learning is adaptive recommendation. Such an ITS can recommend exercises that include a set of knowledge and high-density information designed to maximize the knowledge gap of a particular student [28].

c. Increasing the interpretability and precision of AI-driven systems

In modern intelligent tutoring systems, the knowledge graph is integrated with Large Language Models, which can help overcome errors or illogical explanations of LLMs. There are a precise retrieval and Explainable tutoring [31].

Precise retrieval: When a student asked a question, the intelligent tutoring system uses the knowledge graph to retrieve the precise and domain-specific information. This approach is called knowledge graph-enhanced retrieval-augmented generation (KG-RAG) [30].

Explainable tutoring: An intelligent tutoring system using a knowledge graph can explain why this particular topic was recommended, not acting like a black box.

Intelligent tutoring systems guide the learning path that is suitable for the student based on the knowledge graph.

3. KNOWLEDGE GRAPHS

We will ask and analyze the following questions

RQ 1: What is a knowledge graph?

A knowledge graph, also known as a semantic network, describes the relationship between entities as a network representation of real-world entities such as objects, events, situations, or concepts. Information about knowledge graphs is generally stored in a graph database and visualized by a graph structure.

In an attempt to make full use of the search experience of users and speed up the search, a knowledge graph was first formulated and proposed by Google in 2012. After Google's knowledge graph in 2012, the terms knowledge graph, knowledge base and ontology were used without significant difference. The knowledge graph is essentially a structured semantic network, a semantic knowledge base with a directed graph structure which is used to describe concepts and interrelationships in the physical world expressed in symbolic form [1, 2]. Typical knowledge graphs are DBPedia, Wikidata, and Google Knowledge Graph. DBPedia, Wikidata are two different knowledge graphs for data on Wikipedia.org, both published in RDF format. The Google Knowledge Graph is represented through Google Search Engine Results Pages (SERPs), which provides information about what humans want to search for. The Google Knowledge Graph consists of over 500 million objects and source data obtained from Freebase, Wikipedia, CIA World Factbook, and others.

The components of an intelligent tutoring system are three, the nodes (or vertices), the edges, and the labels. A node can be any object, place, or person. By an edge, the relation between nodes (or vertices) is defined. For example, a node could be an algorithm, like "greedy-best-first-search", and an algorithm, like "informed search". An edge would be to categorize a relationship as a part-of relationship between "greedy best first search" and "informed search".

RQ 2: Why build knowledge graphs? (What are the strengths of knowledge graphs?)

The ability of knowledge graphs is to model the world in an intuitive way for humans and in a very useful way for machines. Here are their key strengths. It enables automatic reasoning. This is the most powerful capability of the knowledge graph. This is a form of logical reasoning that is impossible in traditional databases or vector retrieval systems. Next, knowledge graphs capture the clear and real-world relationships between entities and create a multidimensional view of the data. Finally, the knowledge graphs enable explanatory artificial intelligence, significantly reducing inaccuracies and AI "hallucinations".

RQ 3: What are the applications of knowledge graphs?

The knowledge graph has many applications, especially in retail, recommendation systems, finance, healthcare, education, etc. The application of the knowledge graph is described in detail in Xiaohan[1] (Figure 10). The Xiaohan[1] emphasize that knowledge graphs have many applications, especially in the field of education.

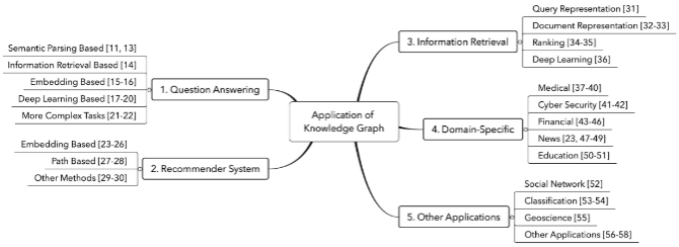


Figure 10. Application Fields of KGs [1]

RQ 4: Do knowledge graphs construct manually or automatically?

There are manual, automatic, and hybrid construction approaches for building knowledge graphs. The manual construction method is a traditional method, in which the domain experts manually defines entities, creates ontology and defines the meaning of relations. The manual method has high accuracy, but it is very time and cost consuming. Automatic construction uses algorithms such as machine learning, Large Language Models (LLMs) to extract triples (topic-topic description predicate-topic) directly from unstructured or semi-structured data without human intervention. Automatic construction approaches use natural language processing techniques, such as REBEL, to identify entities and relations. Automatic construction approaches are currently challenging for domain-specific terminology identification and graph elaboration. In the educational field, KnowEDU is a tool to auto-construct knowledge graphs [2]. This system applied recurrent neural network (RNN) models on pedagogical data to extract instructional concepts, and the educational relationships that interconnect the pedagogical concepts were identified through a probabilistic association rule mining algorithm using the performance data of students.The hybrid approach is the most practical one, first automatically obtained by AI and then calibrated more carefully manually by human experts. AI constructs the initial triplet of knowledge graphs from billions of documents and human experts make fine-tuning of it.

The manual construction method also used Neo4j.

RQ 5: How is a knowledge graph constructed?

Knowledge graphs are accomplished through the construction of knowledge graphs, which is a term used to refer to the process of building a structured network between entities and relationships. The automatic construction of knowledge graph includes traditional pipelining method and agentic knowledge graph construction pipelining method. Then, traditional knowledge graph construction is done in the following steps [2].

a. Data acquisition stage:
This stage collects data from different data sources: structured ones such as databases, semi-structured sources such as Wikipedia, and unstructured sources such as text.

b. Information extraction stage:
Several methods and tools can be used for information extraction. For example, we can use methods such as named entity recognition. You can use "Chao Deng" to recognize and extract entities, and "Occupation: Actor" to extract relationships between them.

c. Knowledge learning and reasoning stage:
This stage is used to complement missing information and refine the information about previously extracted relationships.

d. Sorting (or alignment) of entities and relations:
This stage is to sort the terms present in the same real-world into one and combine the data from different data sources.

e. Evaluation stage:
This stage measures the quality of the knowledge graph constructed by checking granularity, integrity and consistency.
Traditionally, this multi-stage process was implemented using deterministic, batch-oriented pipelines. Most foundational knowledge graphs have been built this way, taking data from various sources, transforming it into entities and relationships, and loading it into a graph database. While conceptually straightforward, these pipelines are rigid and difficult to adapt [28].

A traditional KG pipeline typically looks like this [29].

a. Data indraft: Connect to a source system (databases, APIs, documents).
b. Parsing and transformation: For data conversion, applying their own script or ETL tools.
c. Entity and relationship extraction: Use rule-based NLP or statistical models to detect entities and their connections.
d. Entity resolution: Merge duplicate entities using string similarity or handcrafted heuristics.
e. Validation: Apply fixed rules to check consistency with a predefined ontology.
f. Load into graph: Insert explicit triples to the graph database.
The components of the knowledge graph schema (ontology) obtained using the traditional KG pipelining process are follows [29].

a. Classes (node labels): The categorizes and types of nodes are included.

They define what kind of entities exist in a knowledge graph.
Examples: person, university, student, teacher, subject.

b. Properties (edge labels): This is the category and types of edges present in the graph and defines the types of relationships that exist between nodes.
Examples: part-of, Premise-of, is-a

c. Domains and ranges (the rules): a set of constraints related to node connections that ensure that the corresponding graph is logical and consistent.

Domain: represents the starting node type of an edge.
Range: represents the end node type of an edge.

RQ 6: What is knowledge graph partitioning?

In general, knowledge graph partitioning is the process of partitioning a large knowledge graph into smaller, more tractable subgraphs or fragments. Knowledge graph partitioning is intended to adapt graph processing power to memory or to perform parallel processing through multiple computing node. It also generally works to reduce the complexity of graph construction, to solve scientific and logical inconsistencies, and to facilitate graph construction.In our paper, we describe how hierarchically building a knowledge graph for a subject or topic according to pedagogical and cognitive principles, and how to use a partitioned knowledge graph in an intelligent tutoring system.

RQ 7: In general, how many nodes are extracted when constructing a knowledge graph in a subject?

In general, the number of nodes generated when constructing a knowledge graph may range from hundreds to millions, or even billions.

RQ 8: What is the need to hierarchically partition the knowledge graph?

The need to hierarchically partition the knowledge graph is described in [4]. Generally, for large task scheduling, they use knowledge graph partitioning to partition large tasks into multiple small subtasks.

However, unlike this general graph-splitting principle, we propose a hierarchical graph-partitioning approach in one subject to facilitate graph construction and avoid inconsistencies in graph-building.

We propose a hierarchical graph partitioning method in a knowledge graph with only a single relationship, namely the essential prior knowledge. Such a graph can be viewed as a special case of a knowledge graph with only the “former-latter” relationship. This knowledge graph is a kind of directed acyclic graph. Directed acyclic graphs serve as a powerful tool for intelligent tutorial systems to manage complexity, ensure logical learning paths, and provide personalized learning methods that fit each learner’s needs based on their learning progress and understanding.

First, the nodes of such a graph represent knowledge concepts rather than raw data points. This is consistent with the definition of the knowledge graph.

Second, the single relation of such a knowledge graph reflects the semantic relationship. That is, the "former-latter" relation provides a structured meaning such as order or priority, rather than a simple connectivity relation.

Third, this graph represents a relationship structured by a graph.

Fourth, the knowledge graph of a single relation enables us to use the inferential knowledge. This enables the intelligent tutoring systems to provide a learning path suitable for the learner.

RQ 9: How does an intelligent tutoring system that uses only knowledge graphs build?

Building intelligent tutoring systems using only knowledge graphs can make use of the knowledge graph that plays the role of a ‘brain’ to preserve the educational contents as well as

the diverse teaching skills and student learning progresses, thus making it very flexible ITSs [82].

A brief summary of the process of constructing an intelligent tutoring system using only the knowledge graph is as follows [83].

a. Constructing the domain model as a knowledge graph.

The instructional contents of the subject are constructed in a structured network consisting of nodes (meaning knowledge concepts) and edges (relationships).

b. Integrating the student model into the KG

The student is represented as a dynamic node that exists within the graph, not as a separate database.

First, we create "knowledge-level" (or "mastery") edges between student nodes (the nodes created for student evaluation) and knowledge concept nodes. Then, the weights of the edges are updated in real time based on the student's academic performance evaluation through the student nodes (evaluation nodes). For example, the system updates the knowledge level (or mastery level) from 0.0 to 1.0.

c. Pedagogical logic is defined by graph queries.

Then, the logic of the instructional approach is implemented by an algorithm using the graph traversal rules. The following are the rules for graph traversal.

Path-finding: If a student fails to learn a complex concept, search for "precedence" edges to ensure that the student is not familiar with the basis of that failed concept.

Recommendation: If a student has succeeded in learning the current concept, then he uses a subgraph search to find the next more informative problem to solve. System then chooses the premise concepts that are strongly connected to the current learning goal, not the edges whose connectivity strength is weak.

d. Automatic generation of questions for assessments and tutoring contents

Tutoring content that will be used for tutoring and queries that will be used for interaction are automatically generated using a knowledge graph.

Dynamic learning sequences generation: The system presents the student with learning contents along the edges of the learning path defined for a particular student only.
Answer finding questions: System uses a graph traversal algorithm to extract correct answers directly from nodes and edges, ensuring correctness without external additional support.

4. PREVIOUS RESEARCH ON ITSs AUTHORING TOOLS, CHECKERS (OR PLAYERS), AND PLATFORMS DEVELOPMENT

Generally, ITS authoring tools are needed for the creation of contents of intelligent tutoring, players for delivery, checkers for evaluation, and platforms allow the intelligent tutoring systems to experience in an interactive way in the cloud environment. This paper does not address previous work on platforms because platforms and players are generally in the same mode of operation. Authoring tools must include checkers or players. Thus, in Section 1, which is concerned with the authoring tool, we summarize and analyze the typical authoring tools and checkers (including players).

4.1. Authoring Tools

These tools aim to enable any professional or teacher to develop an intelligent tutoring system for their subjects without developing programs and without knowledge of AI. The following are the tools for authoring ITSs.CTAT (Cognitive Tutor Authoring Tools) is one of the most popular tools, which supports the so-called "example-tracing" and "model-tracing" [36].SimStudent is a machine learning agent developed to help beginner creators build intelligent tutoring systems (ITS) without the need for program development [38].

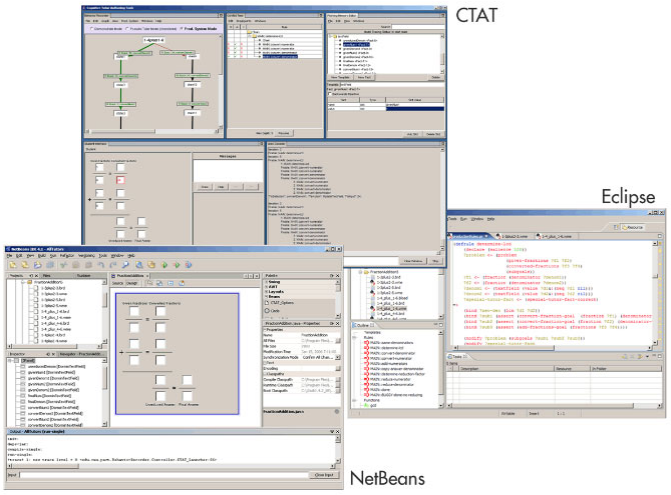


Figure 11. The Cognitive Tutor Authoring Tools[40]

xPST (Extensible Problem Specific tutor) is an authoring tool used to develop an ITS for geometry and statistics [32].
The GIFT (Generalized Intelligent Framework for Tutoring) is a framework developed to enable rapid and standard creation of adaptive training [37].

XAIDA & REDEEM is tools that enable the development of intelligent tutoring systems such as manufacturing and classroom instruction [32].

ACTS (AI-Based Classroom Teaching Simulator) is Conversational ITSs developed by the University of Virginia that simulates the classroom teaching of human teachers.

DomainBuilder is the name of the authoring tool included in the TutorBuilder system, which allows for test and evaluation of creations associated with medical education [39]. The functions are DomainBuilder and TutorBuilder. DomainBuilder has visualized test functions, and TutorBuilder has a playback function.By comparing their solution and a constraint checklist set, it creates a tutor based on constraints that automatically evaluate students. This software verifies the domain model to ensure that the constraints are defining the right solutions [41].

4.2. Checkers (or Players)

The checkers (or players) check and evaluate whether the production artifacts actually improve student learning and whether there is no error in student interaction [33].These tools have the following functions. It functions as a storage and interpretation of interaction information with the student. For example, the Tutorshop tool keeps the information by tracking the actions that the student interacts with the system, and researchers use this information to test instructional models or try to improve their performance [36]. There is also a testing function for natural language. For example, ConceptGrid checks the correctness of the question response by the natural language that the student is given and received, comparing with the results of human student interaction [35].Test methods of checkers are as follows [39]. A simulation-based approach using a tool such as SimStudent is presented, which creates a synthetic student to test the expert model. A tool like TutorBuilder can play or preview the results created by direct interaction. DomainBuilder checks the consistency and integrity of the domain content created using the validity engine [42-45].

4.3. GIFT (Generalized Intelligent Framework for Tutoring)

Our research and development focused and investigated GIFT specifically for the design and development of intelligent tutoring systems. The results are as follows.

The Generalized Intelligent Framework for Tutoring (GIFT) is an open source AI-driven platform developed by the United States DEVCOM Soldier Center to create, manage, and evaluate an adaptive and personalized tutoring system. Also, GIFT is a service-oriented modular framework based on the experience of tools, methods and standards that facilitate the development of adaptive tutoring systems (AIS) [47, 59].The goal of GIFT is to develop an authoring tool that can develop new ITS, ITS components (sensor model, instructional model, user interface, sensor interface), tools and methods, and creation standards that can be used or reused in an external training environment. It also provides instructional managers that include the best teaching principles, strategies and tactics for ITS. It also provides an experimental tool to analyze the effects of components, tools and methods of ITS [48].

The GIFT has three functions: Authoring, Instructional Management, and Experimentation and Evaluation [48].Authoring provides a convenient tool (learner model, teaching model, user interface, sensor interface) for development of ITSs in the field by subject or topic experts who is not aware of information technology, computer science, programming technology, artificial intelligence principles and techniques. Thus, authoring tools play the role of instructional designers through graphical user interfaces, such as routing, evaluation, establishment of adaptation rules, etc., including CAT (Course Authoring Tool), DAT (Domain Knowledge Tool), and SAS (Survey Authoring System) [51, 59].The instructional management function provides an instructional engine (eMAP) to adaptively manage the instructional flow based on real-time data for the learner, and GIFT uses real-time data such as learner learning performance, biometric features, and survey response to adaptively use the learner's cognitive state in the instructor [52].

Experimentation and Evaluation Function: ITSs Researchers and developers work as test tools to evaluate the effectiveness of various tutoring methods. We use this function to evaluate new instructional strategies, test cognitive models, and compare learning outcomes through the analysis of a set of learners [53].

The four components of the above-mentioned intelligent tutoring system are as follows from the viewpoint of the configuration module in GIFT [54].Learner Module tracks and evaluates current learning states, such as student ability, temperament, and learning performance [55].Pedagogical Module determines the best instructional strategy (e.g., feedback or hint) based on the learner state [56].Domain Module converts instructional strategies within a given subject into specific learning actions [57].Sensor Module integrates external data such as physiological sensors, emotions and pose recognizers to complete the learner model [60].GIFT has completed, published and distributed its design documents annually through international academic seminars, which are as follows [61-68].

Learner Modeling documentation was completed in 2012 and the document was released in 2013 as Volume 1: ISBN 978-0-9893923-0-3.
The instructional management was completed in 2013 and the document was published in 2014 as Volume 2: ISBN 978-0-9893923-2-7.
The authoring tools documentation was completed in 2014 and the document was distributed in 2015 as Volume 3: ISBN 978-0-9893923-6-5.
The authoring tools documentation was completed in 2014 and the document was distributed in 2015 as Volume 3: ISBN 978-0-9893923-6-5.
The Assessment Methods document was completed in 2016 and was published in 2017 as Volume 5: ISBN 978-0-9977257-2-8.
The team tutoring document was completed in 2017 and the document was distributed as Volume 6: ISBN 978-0-9977257-4-2 in 2018.

The self-improvement systems (self-improvement systems) document of ITS was completed in 2018 and the document was released in 2019 as Volume 7, International Standard Book Number: 978-0-9977257-7-3.
The operation flow diagram of each module of GIFT is as follows [62].

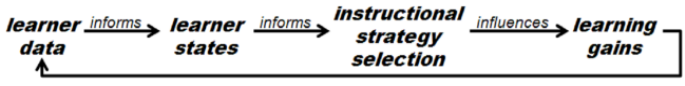


Figure 12. The Overall Flow Diagram of GIFT Platform[63, 74]

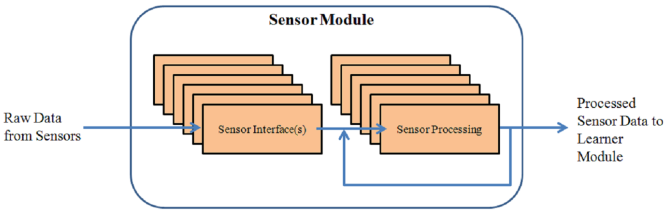


Figure 13. The Flow Diagram of Sensor Modules in GIFT Platform [64]

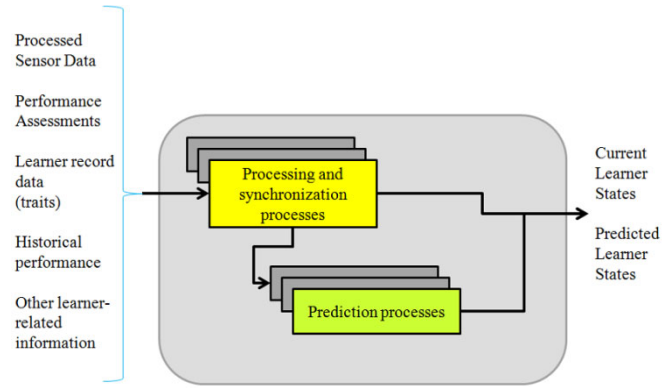


Figure 14. The Flow Diagram of Learner Modules in GIFT Platform[67]

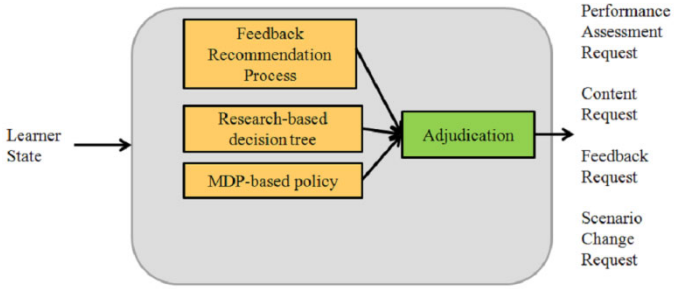


Figure 15. The Flow Diagram of Instructional Modules in GIFT Platform[66]

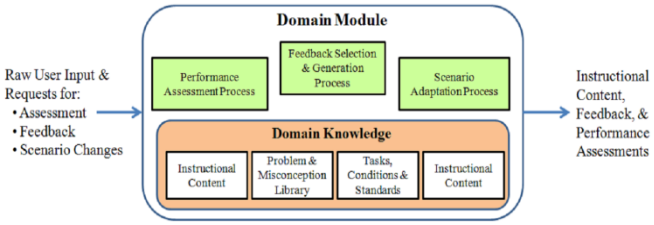


Figure 16. The Flow Diagram of Domain Modules in GIFT Platform[65]

Approaches to GIFT and the development and authoring modes are as follows [66-68, 75, 76].

GIFT cloud: A way to access a cloud that is already built online on the Internet without GIFT on its own computer, and to enable instructional design or course design using the latest updated version.

GIFT Local: This is the way you download an offline version and develop it, providing local service management in Windows, etc. operating systems.

Integration: A collaborative approach to traditional media such as external simulators, games, and PowerPoint.

Domain knowledge file (DKF): In the GIFT design, authors of courses decide which instructional strategies should be implemented based on transitions of learner states, which are authored in the Domain Knowledge File (DKF).

Gateway module: The gateway module provides a connection between the external training application (such as VBS2, PowerPoint, TC3 Sim/Vmedic) and the GIFT in add-in (plug-in) mode.

For example, GIFT 4.0 provides a set of authoring tools such as the author's curriculum (CAT, Course Authoring Tool), the author's adaptive feedback and instructional strategies (DAT, Domain Knowledge File Authoring Tool), and the author's survey (SAS, Survey Authoring System).

The creation of the curriculum of GIFT is as follows [68].

.xml Editor: All the authoring tools of GIFT use the.xml editor or visualized interface.
CAT (Course Authoring Tool): The author uses CAT to design the required course flow.
Node in CAT: Added and extended within CAT to add a transition between concepts, surveys, guidance provided to external training applications and learners.

CAT's link function: The author links training applications in the course to the specific DKF files they authored.
Export Tool: After the course is completed, it is possible to play on a computer without the installation of the full GIFT version.
The following Figure illustrates the authoring process in GIFT [68].

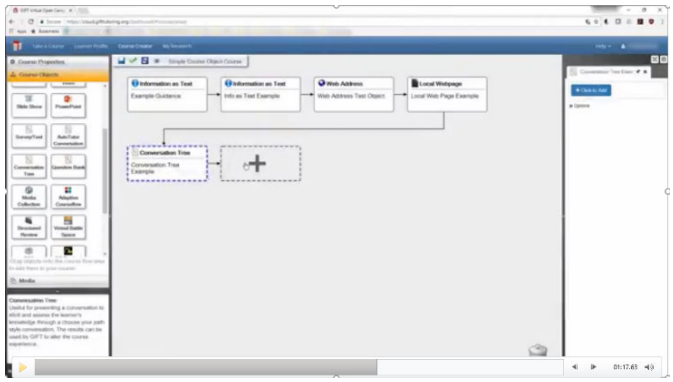


Figure 17. An Example of an Instructional Model Authoring Flow Diagram in the GIFT Framework (1)[68]

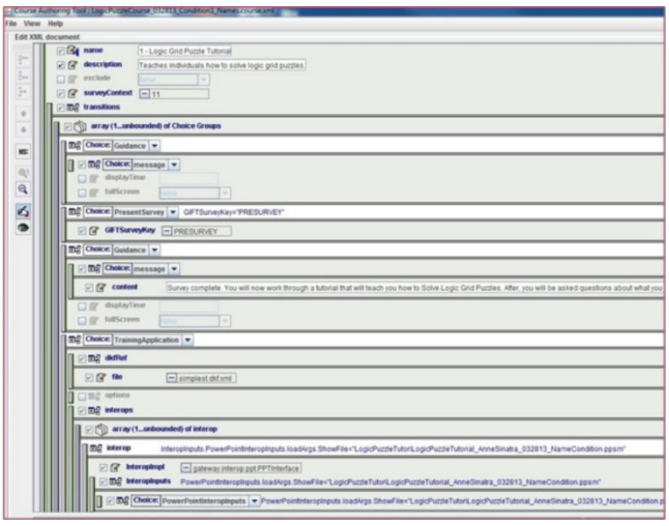


Figure 18. An Example of an Instructional Model Authoring Flow Diagram in the GIFT Framework (2)[68]

The architecture and operation flowchart of GIFT is as follows [62].

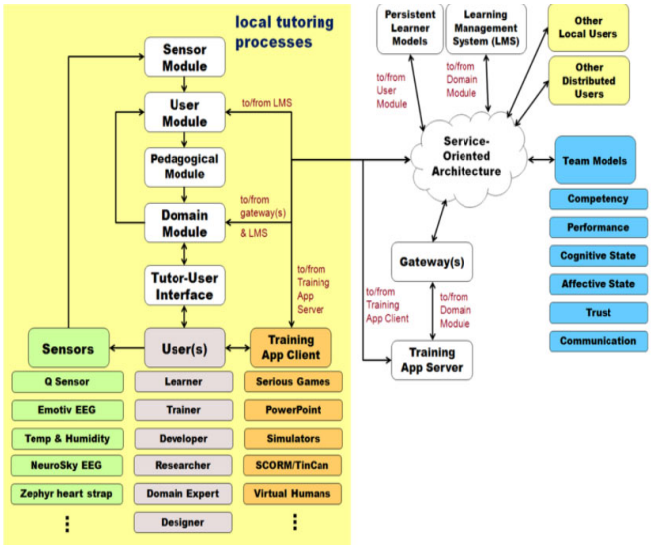


Figure 19. The Architecture and Operation Flow Diagram of GIFT Platform[65]

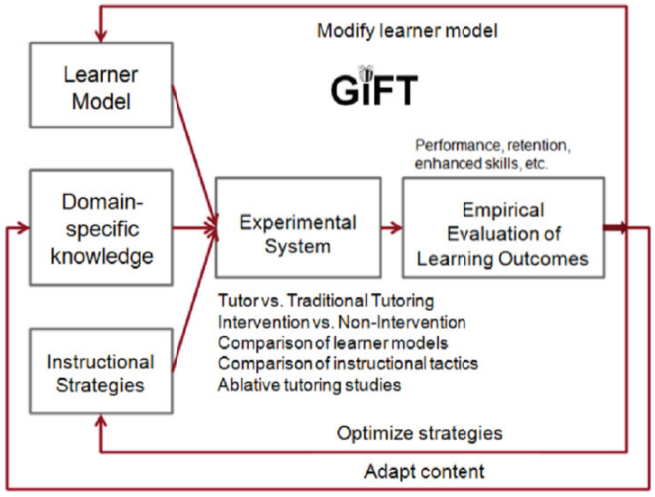


Figure 20. The Architecture and Operation Flow Diagram of GIFT Platform (Overview)[65]

In GIFT, the interaction between learner, tutor and environment takes place as follows [67].

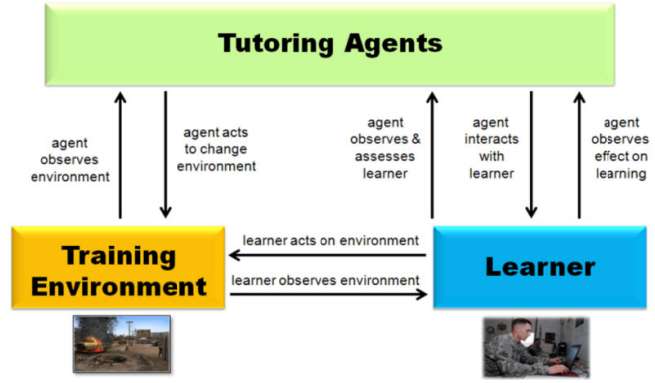


Figure 21. Interaction between Learner, Tutor and Environment (from an Agent Perspective) on the GIFT Platform[66]

In GIFT, the interaction between the elements of the system, the individual learner and the external environment is done as follows.

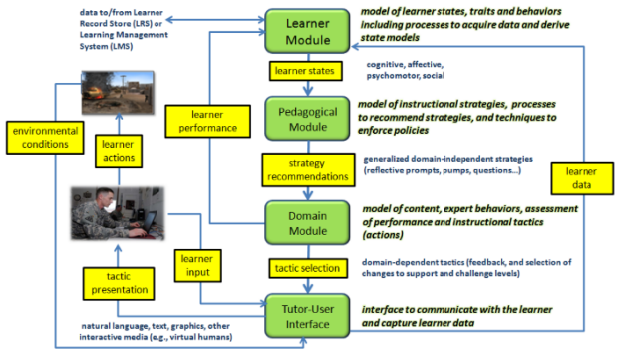


Figure 22. Interaction between System Components, Individual Learner and External environment on GIFT Platform[66]

In GIFT, the Learning Effect Model (LEM) for individual learners is as follows [67].

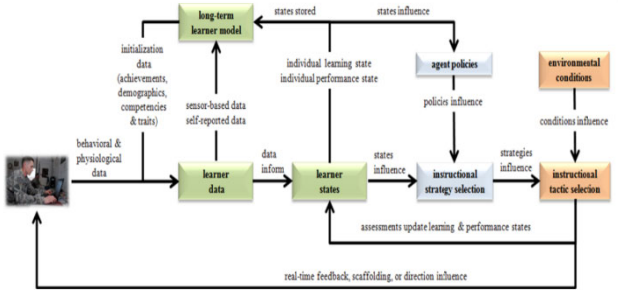


Figure 23. Learning Effect Model (LEM) for Individual Learners on the GIFT Platform[67]

The Learning Effectiveness Model (LEM) for Group Learning in GIFT is as follows [68].

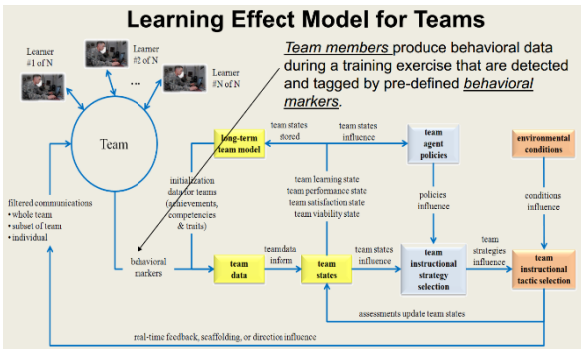
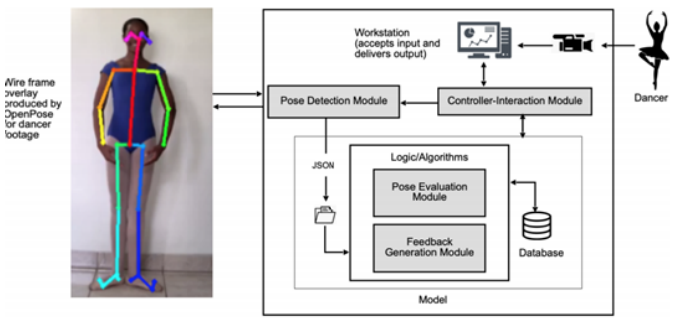


Figure 24. Learning Effect Model for Group Learning on GIFT Platform (LEM)[68]

GIFT is used for multi-uses and multi-purposes, and is also connected to Kinect, Game, and PowerPoint.



High level sketch of the overall architecture of DanceTutor.

Figure 25. The Connection between GIFT and Kinect[68]

5. POMDP-BASED ITSs

5.1. Examples of ITSs Based on POMDP

In Fokaet al. [94], a Robot Navigation - Hierarchical POMDP(RN-HPOMDP) is proposed. This approach allows us to efficiently model complex and large environments using state-space and action-space hierarchies and using a fine-grained decomposition approach to the task. This hierarchical POMDP approach maintains all the information regarding robot movement and sensor uncertainty, thus making the memory of the hierarchical structure efficient and fast the reinforcement learning of parameters.In Theocharouset al. [95], the advantages of representing hierarchical partially observable Markov decision processes (HPOMDPs) as dynamic Bayesian networks (DBNs) are investigated. The results showed that DBN representation of H-POMDPs can learn significantly faster than the learning algorithm in HPOMDPs or equivalent single-layer POMDPs and requires much less data. The DBN formulation can also be easily extended to parameter tying and factoring of variables, which further reduced the complexity of time and sample.In Hansen et al. [96], a hierarchical approach is proposed to plan a partially observable Markov decision process (POMDP) where the policy is represented by a hierarchical finite-state controller. They proposed a scheduling algorithm using programmer-defined task hierarchy to constrain the search space of finite state controllers and proved that this algorithm converges to a ε-optimal hierarchical finite state controller in terms of recursive optimality.In MAHADEVAN et al. [97], the authors studied methodological issues that can be addressed in hierarchical probabilistic models for decision making, such as simultaneous interaction, multiagent control, and hidden state estimation in stochastic environments. In particular, they studied the relationships between Markov chains, Markov decision processes, semi-Markov chains, semi-Markov decision processes, hierarchical hidden Markov models, and partially observable Markov decision processes.

In Pineauet al. [98], an algorithm for scheduling in structured partially observable Markov decision processing (POMDP) is proposed. Then, a new algorithm, called Policy-Contingent Abstraction (PolCA), uses action-based decomposition to decompose complex POMDP problems into a hierarchy of smaller subproblems. Low-level subtasks are solved first and their sub-rules are used to model abstract actions in the context of high-level sub-tasks.In Salomonset al. [99], a Bayesian Tracking-Partial Observation Markov Decision Processing (BKT-POMDP) method is proposed as a fast-selection method for modeling learner expertise for tasks involving multiple expertise.In Daubigneyet al. [100], an optimization method for building ITSs using POMDP is proposed.In Ramachandran [101], the student learning gains are significantly increased when using a personalized robotic tutoring system using Assistive Tutor POMDP (AT-POMDP).In Folsom-Kovariket al. [102], they propose a method to reduce the size of states and observation sets when using POMDP in tutorials.

5.2. Challenges in building POMDP-based ITSs

We analyze the problems of converting the knowledge graphs into POMDPs and the problem of constructing hierarchical POMDPs.

5.2.1. Knowledge Graphs and POMDPs

1) One-dimensional knowledge graphs and POMDPs

We can also construct POMDPs directly, but if we already have a knowledge graph, we can use this knowledge graph to convert it to POMDP, using a one-dimensional knowledge graph. The conversion of one-dimensional knowledge graph or directed acyclic graph to POMDP has been previously studied by Fangju[84]. Here, they used the nodes of a directed acyclic graph to states in POMDP and the “former-latter” relation in a directed acyclic graph to define the transition probability in POMDP. In this paper, they propose an approach to construct POMDP directly using a one-dimensional directed acyclic graph. However, no previous work has been reported on the automatic or semi-automatic conversion of constructed directed acyclic graphs to POMDP.

2) Hierarchical POMDPs

Hierarchical Partially Observable Markov Decision Process (HPOMDP) construction involves the process of decomposing a complex large-scale task into a multilayer structure consisting of smaller and more tractable subtasks Toussaint et al. [92]. In general, hierarchy can abstract state and action spaces, thus saving the ability to handle uncertainty and greatly reducing the time spent in the planning process.

The hierarchical POMDP construction is followed in detail [85].

a. Decompose the task hierarchically
Here, a large task is decomposed into a tree-like structure, where each internal node is an abstract POMDP and each leaf node is a primitive action.

b. Define Level-Specific Models at each level.

Low-level models (concrete models): Modeling direct interactions such as robot navigation and motor control with high resolution and observation space.
High-level model (abstract model): This uses a rough representation to schedule sub-goal sequences (Not like steering to the left, but like moving to a room).

c. Establish the transition functions and the rewards functions.

Then, the sequence of actions is abstracted.
Abstract actions: A complete low-level action strategy corresponding to one abstract action in the upper layer is addressed.
State abstraction: To simplify decision making at a high level, idealized states are matched to abstract ones.

d. implementing beliefs tracking.

Any belief state (probability distribution for true states) that exists between the hierarchies (between levels) is preserved. Advanced methods such as Tru-POMDP use a large-scale language model (LLMs) to build a Trees of Hypotheses to better track beliefs in open-ended environments.

e. Coordinate Execution (Bottom-Up or Top Down)

All strategies are solved in bottom-up order. Here, the optimal values at low level are rewards values for the abstract actions at high level.

What are the frameworks and tools for hierarchical POMDP?

a. Knowledge-based HPOMDPs
This is a framework for a service robot that automatically constructs a hierarchy using encoded environmental knowledge [86].

b. Object-Oriented POMDPs (HOO-POMDP):A modular approach for multi-objective rearrangement in multi-room environments

c. POMDPs.jl Ecosystem
This is a Julia-based framework providing tools (Julia), a solver for construction, and models decision making in the presence of uncertainty.

The hierarchical POMDP construction is performed following a flowchart [87].

a. Defining the hierarchy
Identify the tree structure for task H.

b. Definition of sub-tasks
Define local observations, actions, and termination conditions for each leaf node task.

c. Model transition and reward functions
Define the transition function $T(s, a, s')$ and the reward function $R(s, a)$ for low and high levels.

d. To solve the low level POMDPs
Use either on-line solvers such as DESPOT and POMCP or strategy iteration algorithms.

e. Assembling the hierarchy
Sub-strategies are connected to a comprehensive controller.

3) Hierarchical knowledge graph and hierarchical POMDP

The transformation of a hierarchical knowledge graph into POMDP involves matching the semantic and structured knowledge, such as the relation between objects or the hierarchy of tasks, to states, actions, and reward functions. When a large task, subject or topic, is decomposed into smaller sub-problems with edges defining transitions or relations, entities or nodes with abstract states, we use the hierarchical relationship of this knowledge graph.

5.2.2. Hierarchical Knowledge Graphs and Hierarchical POMDPs

The method of transforming a hierarchical knowledge graph into a hierarchical POMDP is as follows [88].

a. The abstraction of the state space is done through a hierarchical decomposition of the state space [89].

Structure (structure): A hierarchical knowledge graph is used to build a hierarchy of states, where leaf nodes are primitive states, and parent nodes are abstract states [90].
Abstraction: At each level of abstraction, the action space is reduced by decomposing a large complex task into sub-goals.

b. Knowledge-based rewards modeling

Hybrid rewards: The knowledge graph can easily obtain a hybrid reward value that combines low-level immediate reward signal values (obtained from task steps) with high-level semantic reward values (based on task goals) [91].
Graph-Proir Decision: An agent uses the embeddings obtained by Graph Attention Networks (GAT) to rank important tasks based on the relationship.

c. Knowledge-guided belief state

Representation: The knowledge graph serves as a compact belief state within POMDP, which acts as a “memory” for past observations that are essential in a partially observed environment. The system must infer hidden states from incomplete observations.
Dynamic update: When the system moves to another new learning state, the graph is updated to reflect the new observations [92].

d. Hierarchical task scheduling (bottom-up)

Procedure: Abstract actions are modeled as partial POMDPs for transitions between abstract states at the same hierarchical level.
Transition probability: This is computed to simulate low-level strategies.

6. STEP-BY-STEP MODE IN ITS FOR TUTORING

RQ 1: What is a step-by-step tutoring mode?

Step-by-step mode in intelligent tutoring systems is a pedagogical approach designed to guide the learners by decomposing tasks into small and manageable steps and solving complex problems. Unlike systems that only give the final answer at once, this approach promotes ‘active learning’ by scaffolding the learning process. Step-by-step mode realizes immediate feedback for each action of the student and gives hints whenever necessary [77-81].

RQ 2: What are the key features of step-by-step intelligent tutoring system?

a. Scaffolding & Decomposition
Large tasks are decomposed into small tasks, which reduces the cognitive burden and allows the learner to solve a subproblem at one time [77].

b. Immediate Feedback (Inner Loop)
The system does not wait for the student to complete the large task, but evaluates each step so that the defect can be corrected and completed immediately [78].

c. Adaptive Hints
If a student fails at a certain learning stage, the ITS provides hints from general reminders to explicit instructions based on the students’ level [79].

d. Proactive Intervention
The systems can intervene with the student’s learning if a student’s learning is “stuck”. This provides direct training data related to the early step of the learning step that are currently facing difficulties or presents examples [80].

RQ 3: How is the step-by-step mode implemented in modern ITSs?

In the following, we illustrate various systems in which a step-by-step tutoring approach is implemented.

a. AI Tutoring Platforms

Platforms such as “Study Mode” in ChatGPT or Gemini do not allow direct answers of users. Instead, they activate a guidance mode that uses questions that decompose problems [81].

b. Educational Software

Step-by-step tutoring systems such as Hawkes Learning use interactive steps to guide students in subjects such as mathematics or microeconomics [80].

c. Programming Supporting

These systems (e.g., Stitch) analyze the student's code, compare it with the goal solution, and provide a step-by-step guide to correct errors.

d. Example-Based Feedback

For example, a system like Step Tutor. Students see example codes or case problems before and after their activities, compare their results or reflect on their results.

RQ 4: (Pedagogical Advantages) What advantages do the step-by-step tutoring mode have over other methods from the pedagogical point of view?

a. It increases the effect of positive direction of learning [79].

The step-by-step tutoring mode has a greater effect on student learning than the traditional computer-based approach. This is possible only when the student is actively involved in the learning activity, not simply providing an answer.

b. This mode is very effective in correcting errors [78].
This mode often uses positive reinforcement for successful learning steps and constructive correction for incorrect steps.

c. Adjusting individual learning pace

Students are taught with their own learning speed. More skilled learners receive very little or very subtle hints, but relatively less skilled learners are more directed [77].
d. When using the step-by-step mode, conceptual understanding is further facilitated.
By focusing on the problem-solving process, students can better understand the underlying principles.

From Question 1 to Question 4, it can be seen that the step-by-step tutoring mode has some features compared to the other tutoring methods [77-81]. One is the difference in the “answer” mode. The traditional approaches are to provide immediate final answers, while the step-by-step approach is to encourage “student to do the work”. The other is the difference between the step-by-step and computer-aided instructions. While in Computer-Aided Instruction (CAI) feedback is simply implemented, the step-by-step provides more precise and adaptive inner-loop learning guidance.

7. THE PROPOSED INTELLIGENT TUTORING SYSTEM SPECIFICATIONS

7.1. Intelligent Tutoring System Requirements

Our research team analyzed the requirements of an intelligent tutoring system which type of intelligent tutoring system needs through consultation with many teachers in the country. The results are summarized as follows.

a. Develop a one-to-one tutorial system that reproduces the classroom teaching style of human teacher.
The reasons for this requirement are as follows. This is because some students want to give a lecture that reproduces classroom lecture styles for themselves when they do their own learning in the afternoon after receiving a human teacher’s lecture in the morning.
b. Make sure that any subject teacher can develop an intelligent tutoring system for his/her subject without developing a program and without knowing AI.

7.2. Intelligent Tutoring System Specifications

Our developer side worked out the following specifications through consultation with teachers of universities and schools at all levels:

a) The intelligent tutoring system to be developed is composed of following three applications: Intelligent Tutoring System Authoring Tool, Checker (or Player), and Platform. In the future, we will call the Intelligent Tutoring System Authoring Tool (including Checker) the Intelligent Tutoring System framework. Intelligent Tutoring System Authoring Tool, Checker and Platform are developed for applications running on Web and personal computers, respectively. Here, the authoring tool is needed for creation, and the checker is needed for reproduction and testing of the production results. The developers use a checker to test the results of their authoring tool in the same environment as the platform, when they check whether they can distribute their creative results by interacting with the system from the teacher’s and student’s point of view.
b) The intelligent tutoring system to be developed is an adaptive, knowledge-tracking tutoring system that tracks the student model and then performs tutoring based on the knowledge graph.
c) The player (or checker) and platform of the intelligent tutoring system to be developed is a one-to-one tutorial system that reproduces the classroom teaching style of human teacher.
d) Develop an intelligent tutoring system authoring tool to enable any subject teacher to develop an intelligent tutoring system for his/her own subject.
e) Use a hierarchical decomposition of the knowledge graph to allow teachers to create a knowledge graph for their subject easily and logically without conflict.
f) We develop and provide a visualized, graphical knowledge graph authoring module that enables teachers to easily create a hierarchically partitioned knowledge graph for their subjects. This module is embedded in the authoring tool. Then we must add an acyclic violation check function in the directed acyclic graph.

g) The tutoring models to be developed are the step-by-step tutoring model, the tutoring model using only the knowledge graph, and the tutoring model using POMDP.
h) The authoring tool includes the ability to convert manually constructed hierarchical knowledge graphs into POMDP in a semi-automatic way.
i) Finally, we add a function that reduces the computational complexity in the POMDP model.

8. THE OVERALL ARCHITECTURE, ACTION FLOW DIAGRAM OF THE PROPOSED ITS, AND UI

8.1. ITS Architecture Diagram and ITSs Development Process

8.1.1. ITS Architecture Diagram

The proposed intelligent teaching system is largely composed of three sub-systems. The following are the three main components of the system: the ITS Authoring Tool, the ITS Checker (or Player) and the Platform. The description of the subsystems of the ITS is given in Table 1. Figure 26 shows the overall architecture of the "Intelligent Tutoring System that reproduces the human teacher's classroom lecture style". The ITS is largely composed of a framework and a platform. The framework consists of an ITS authoring tool and an ITS checker (player). The framework and platform were developed for application and web, respectively. The framework is used by the teacher and the domain expert, and the platform provides interaction with the learner.

Table 1. The Subsystems of ITS

	Function names	Explanation	User
Sub-system 1	ITS authoring tools	The teacher creates one model and basic data for several models in the tutoring system of his subject. In particular, we author (or create) domain models, basic data of student model, and basic data of instructional model.	teachers, experts
Sub-system 2	ITS checkers (or players)	When the goal is to distribute the product authored (created) using the authoring tool, the player is used, and the tester is used when the model is aimed at improving performance.	teachers, experts
Sub-system 3	ITS platform	Learner interacts with this platform to enhance learning activities.	learners

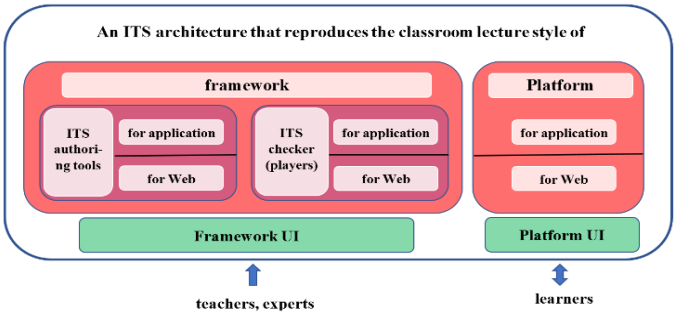


Figure 26. An ITS Architecture that Reproduces the Classroom Lecture Style of Human Teacher

8.1.2. ITSs Development Process

When discussing ITSs, there are developers who develop systems and learners who use it. Developers are divided into programmers (including system designers) and authors who creates tutoring contents using the ITS authoring tools. Programmers develop the framework and the platform in a programming way. The framework development includes development of authoring tools and development of checkers (players). When the authoring tools, players and platforms are developed by the developer, authors use the authoring tools to create ITS contents of his subject. The created tutoring contents are distributed through appropriate checker and player. The platform calls the distributed intelligent tutoring contents and uses it for interaction with the learner. Figure 27 shows the development process of the intelligent tutoring system.

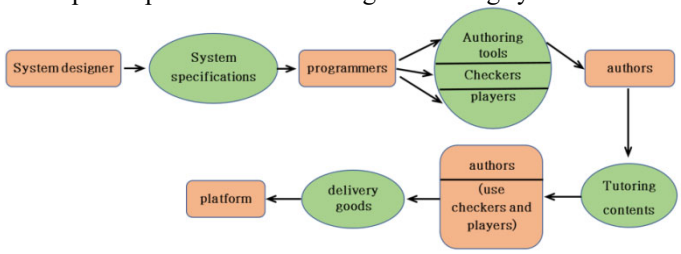


Figure 27. Development Process of the Intelligent Tutoring System

8.2. ITS Framework

The ITS framework includes authoring tools and checkers (players). Figure 28 shows the proposed ITS framework structure and action flow diagram.

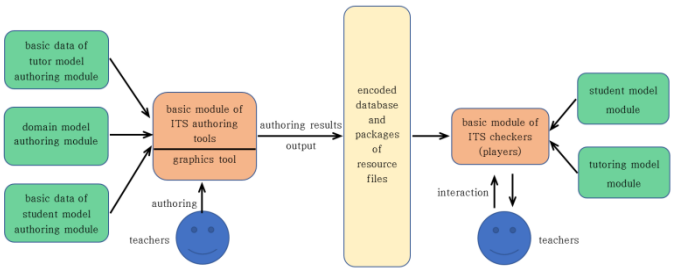


Figure 28. The Framework Architecture of "ITSs Reproducing Human Teacher's Classroom Lecture Form"

8.2.1. ITS Authoring Tools

The goal of the ITS authoring tool is to enable any subject teacher without knowledge of programming experience and artificial intelligence to develop an ITS for his/her subject simply by keyboard editing and mouse-based drawing. The authoring tool we propose is a manual. The functions of this authoring tool include the hierarchical partitioned knowledge graph construction using a graphics tool, the domain model (expert model) construction for the subject, the basic data creation of student model and the basic data creation of tutoring model. We use a hierarchical partitioning approach to construct knowledge graphs using a graph drawing tool. That is, a hierarchically partitioned knowledge graph is obtained.The domain model for the subject consists of pictures, videos, and comments (txt format) used in the subject.The student model is a comprehensive assessment of the learning process towards the student's positive direction, which means information about the learning concepts that the student fully realizes, information about the learning concepts that he or she is incorrectly aware, and quantitative knowledge of specific concepts. Generally, the student modeling methods include the empirical linear regression model, Instruction Factor Model (IFM), Performance Factor Model (PFM), Item Response

theory (IRT) as the student tracking model, stochastic Bayesian model, Hidden Markov Model belonging to the time interval model, etc. These models require input data for sensory information such as student response, body posture, eye movements and states, head movements, etc. In particular, the system's query content and the assessment of student responses to it should be constructed manually by experts or teachers in the field. We define the evaluation measures of the responses to the questions and the questions that will be used for student evaluation as basic data of student model.If feedback is required during a student's learning process, the instructional material and supplementary explanatory material for the feedback points, suggestions (text, pictures, questions and evaluation of the results of the responses to the questions) should also be constructed manually by experts or teachers in the field. We define the construction of such data as the authoring of basic data of pedagogical model.The creation results of the authoring tools are hierarchical partitioning knowledge graphs manually constructed by the teacher or expert, domain models for the subject, basic data for student model, and basic data for pedagogical model. We call these artifacts created by the ITS Authoring Tool the Intelligent Tutoring Contents. Intelligent tutoring contents are a database and various types of files that have information about the knowledge graphs and the pedagogical contents. For example, txt, .doc, .png (.jpg), and .mpeg (.mpg, .mov, .3gp, …). The intelligent tutoring contents are encoded in a suitable way for delivery and stored in a single file. Figure 29 below shows the flow diagram of the creation of the intelligent tutoring contents (relating to the contents of the step-by-step intelligent tutoring).

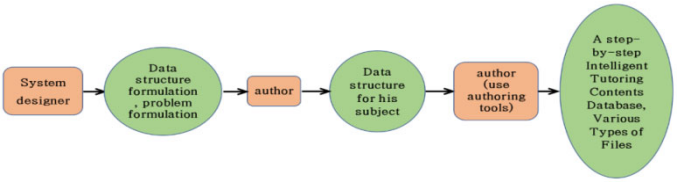


Figure 29. An Intelligent Tutoring Content Flow Diagram

The intelligent tutoring contents are divided into three broad categories: the step-by-step intelligent tutoring contents, the intelligent tutoring contents using only the knowledge graph, and the intelligent tutoring contents using POMDP. The intelligent tutoring contents using only knowledge graph and the intelligent tutoring contents using POMDP are created based on hierarchical partitioning knowledge graph. The authored intelligent tutoring contents is the input of the ITS checkers (or players).

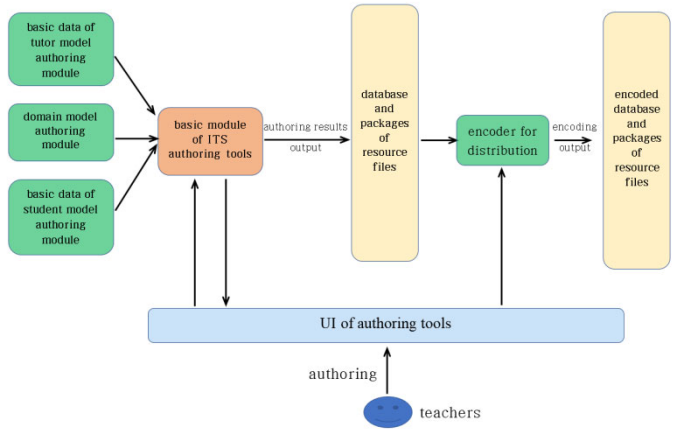


Figure 30. ITS Authoring Tools Architecture and Flow Diagram

8.2.2. ITS Checkers (Players)

The ITS authoring tool uses a player for delivery purposes and a checker for further improvement of the current production results. As already described in previous studies, there are cases where the checker and player are implemented by different tools and implemented by one tool. In this paper, we propose a method to implement the checker and player by a single tool.

The functions of the checker and player include decoding of the encoded intelligent tutoring contents, predicting and estimating the student's learning status, and generating the tutoring path. This is implemented by the decoder module, student model module, and tutoring model module.

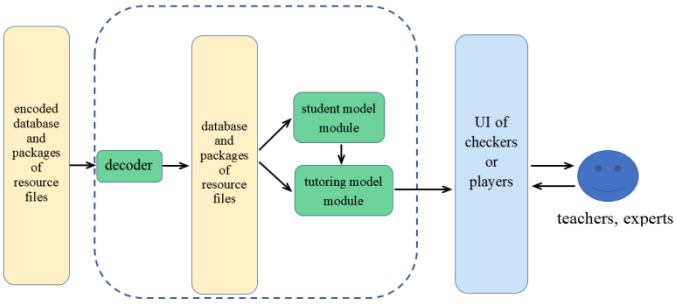


Figure 31. ITS Checkers (Players) Architecture and Flow Diagram

8.3. ITS Platform

First, the learner's needs, and the learner's sensory information (raw data) are entered into the student model, then the analysis is performed to obtain the learner's learning-related state information. When this information is entered into the tutoring model, the pedagogical model refers to the domain model, generating the right pedagogical strategy and the instructional path. The interaction with the student is done by the instructional strategy and the tutoring path, and consequently, the improvement of the learner's learning status is achieved. The following Figure s (Figure s 32 and 33) show the proposed ITS platform flow diagram and architecture.

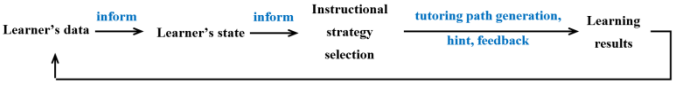


Figure 32. A Proposed ITS Platform Flowchart

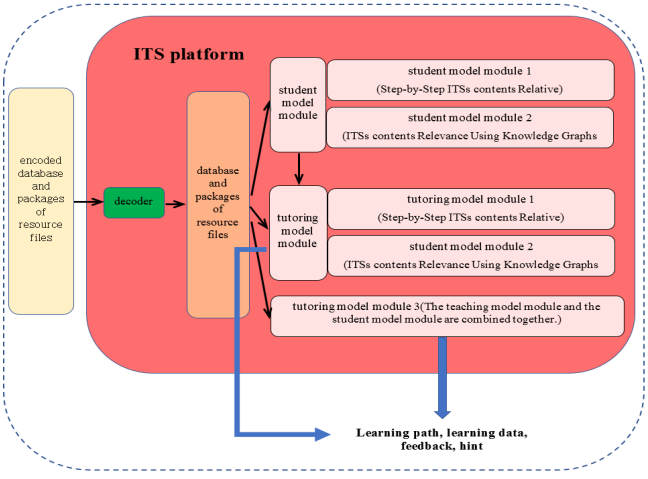


Figure 33. A Proposed ITS Platform Architecture

8.4. ITS UI

8.4.1. ITS authoring UI

The UI when the ITS started is as follows.

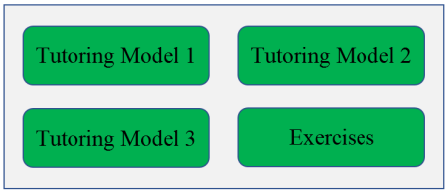


Figure 34. A First Start Screenshot of ITS Authoring Tool

The Tutor Model 1 implements the step-by-step tutoring contents authoring function, Tutor Model 2 implements the tutoring contents authoring function using only the knowledge graph, and Tutor Model 3 implements the tutoring contents authoring function using POMDP.

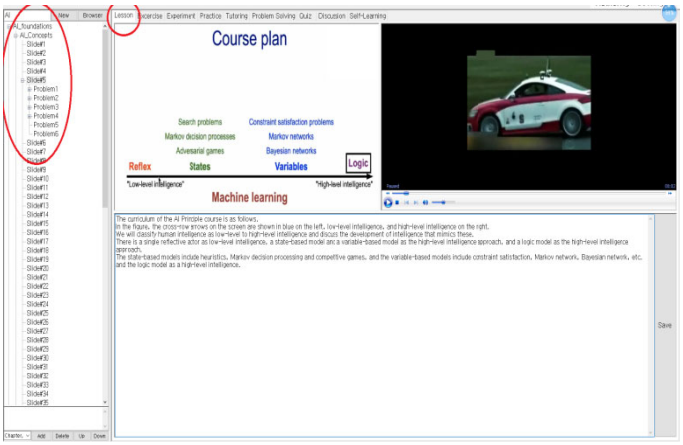


Figure 35. An UI of Step-by-Step Tutoring Contents Authoring Functions

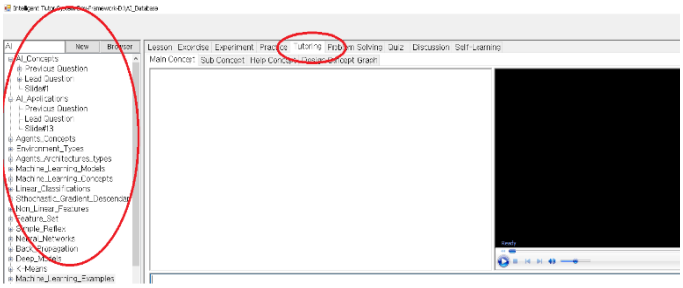


Figure 36. An UI with Hierarchical Partitioning Building Functions of Knowledge Graphs

If you have selected tutoring models 2 and 3 on the first startup screenshot of the authoring tool (or when you have drawn a knowledge graph using the graphics tool), the list that floats to the left of the screenshot is as follows [36].

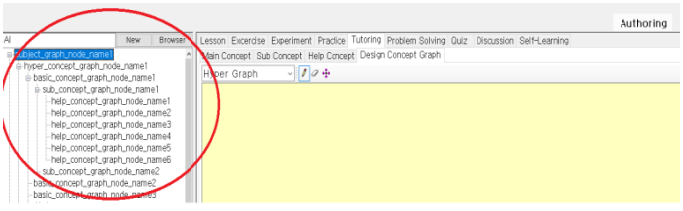


Figure 37. A List that Floats to the Left of the Screenshot when you Draw a Knowledge Graph using a Graphics Tool

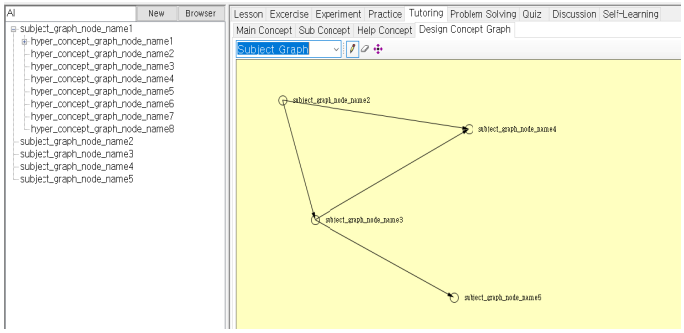


Figure 38. An UI with Hierarchical Partitioning Constructs of Knowledge Graphs using Graph Drawing Tools

8.4.2. An UI of the ITS Checker (Player)

A first starting screenshot of the ITS checker (Player) is shown in Figure 39 below.

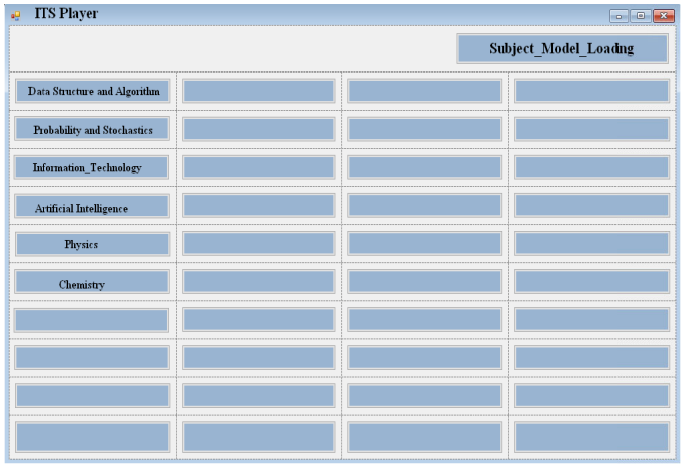


Figure 39. A First Starting Screenshot of the ITS Checker (Player)

After the AI course is selected, the UI is.

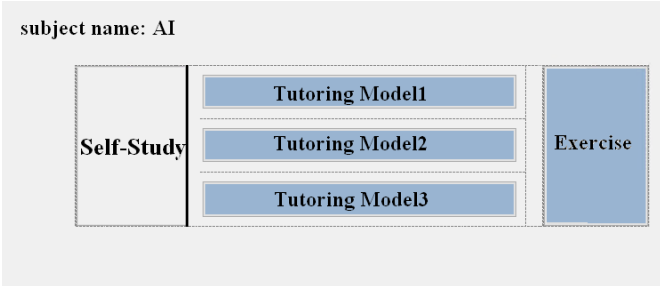


Figure 40. An UI After the AI Course is Selected

The following Figure 41 shows the UI of the case of the choice of the tutoring model 1. Currently, the UI is composed of four parts. It is marked with red circles. The region in which the circle is written with a number of 1 is the area where the academic name of the chapters and sections in the subject is represented. Region 2 is the part that represents the display of electronic visual aids (Slide or pictures) on TV when a teacher is in class. Region 3 is the part where the teacher reproduces the processing of video-projected sessions on the lecture topic. Regions 2 and 3 enable the display of the full screen with a single mouse click of the learner. Region 4 is a part of the representation of the teacher's voice, which the learner may understand with the eye or if you press the "Reading" button, it is output to the audio by the speech synthesizer.

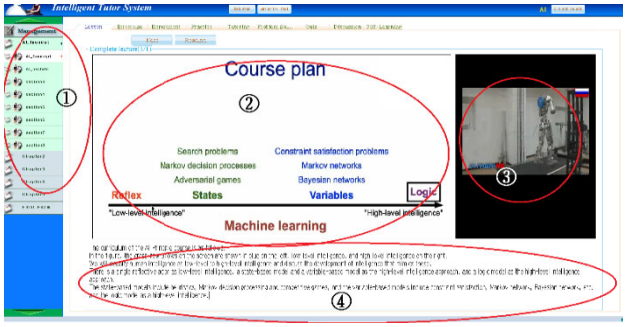


Figure 41. UI Related to the Step-by-Step Tutoring Contents Testing Function in a Checker

Figure 42 shows the UI associated with the tutoring contents testing function using only the knowledge graph in the checker. Differentiating from UI related to the step-by-step tutoring contents testing function is only one, which means that domain 1 is different. In UI, the first area is not the chapter or section, but the so-called 'basic concepts' that exist in the subject.

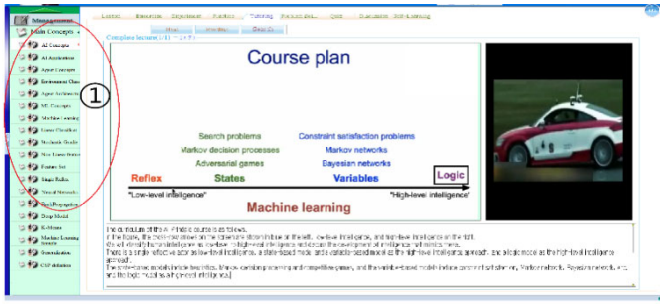


Figure 42. An UI Related to Tutoring Contents Testing using only Knowledge Graph in Checker (Player)

9. A HP PRINCIPLE OF KNOWLEDGE GRAPHS AND A HPKG CONSTRUCTION METHOD USING GRAPH DRAWING TOOLS

We propose an approach for creating hierarchical partitioning knowledge graphs (HPKGs) for intelligent teaching using visualized drawing tools without knowledge of AI or programming by experts or teachers in any field.

9.1. A Hierarchical Partitioning Principle of Knowledge Graphs

As explained earlier, we will use a hierarchical partitioning of the knowledge graph to allow any subject teacher to construct a knowledge graph for his or her subject without discrepancies and easily, and provide a graphics drawing tool that enables him to implement it. The knowledge graph to be treated is a graph that has the property of directed acyclic graphs with only a single relation called the "former-latter". Here, the concept of "former" is a prerequisite to be known before any knowledge concept is learned. The term "latter" is either a knowledge concept that, if already learned, is believed to succeed through the learning process, or a learning concept chosen by the learner.

We define the knowledge graph as follows.

G= (V, E), V is the set of nodes (or vertices), E is the set of edges (relations). V = $\{v_i|\ i=1,2, \ldots, n\}$ is a set of possible knowledge concepts present in the subject. When E = $\{e_i|e_i=<v_{i1}, v_{i2}>, v_{i1}, v_{i2} \in V, j=1, 2, \ldots, m\}$ is a set of possible directed acyclic edges with a former-latter concept relation, the graph G = (V, E) is called a knowledge graph of the subject. In this knowledge graph, vertices reflect the knowledge concept present in the subject, and edges reflect the former-latter order relation between those knowledge concepts. This knowledge graph must be acyclic. If we have to learn concept v1 for two concepts v1 and v2, then we have to learn concept v2, we denote it by the directed edge e=<v1, v2>.

We start with a knowledge graph from a "subject graphs" and construct a hierarchical partition into "hyper-concept graphs", "basic concept graphs", "sub-concept graphs" and "help-concept graphs". These graphs have edges with vertex, which indicates educational or cognitive sequences. The definition of knowledge graphs belonging to each hierarchy of this hierarchical partitioning-type knowledge graph is as follows.

9.1.1. Subject Graphs

We define the subject graph as $G_{sub}= (V_{subi}, E_{subi})$. The index *i* is an integer index symbol, $i \in [1, N_{sub}]$, N_{sub} is the total number of vertices of the subject graph, V_{subi} represents one vertex of the subject graph, and E_{subi} represents one relation. A vertex of a subject graph defines the set of the largest concept blocks in the subject and the relationship defines the sequential relationship between the vertices. Figure 43shows an example of a subject graph.

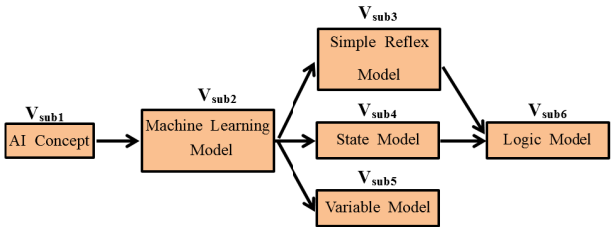


Figure 43.Subject Graph Example

9.1.2. Hyper-Concept Graphs

A hyper-concept graph is further constructed by decomposing the vertices of a subject graph, which is defined as follows:
$G_{sub} = (V_{subi}, E_{subi}) \approx G_{hyper} = (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{hyperl})$, $i \in [1, N_{sub}]$, $j \in [1, N_{hyper}]$, $l \in [1, N_{subi}]$ *i*, *j*, *l* is the total number of vertices of the hyper-concept graph, and N_{subi} is the number of child vertices belonging to the *i*th node of the s subject graph. The symbol ≈ indicates that the meaning of the two graphs is equivalent to each other. The following Figure s show the hyper-concept graph.

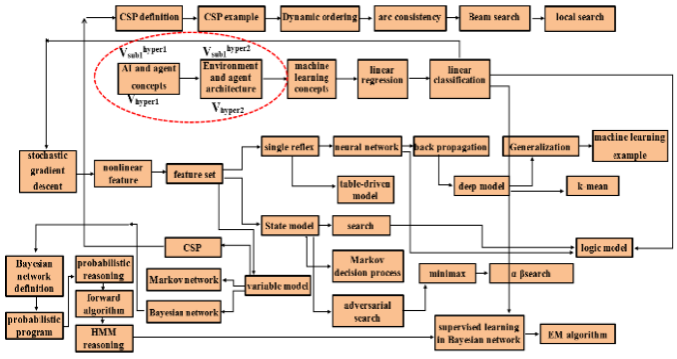


Figure 44.Hyper-Concept Graphs Example (1)

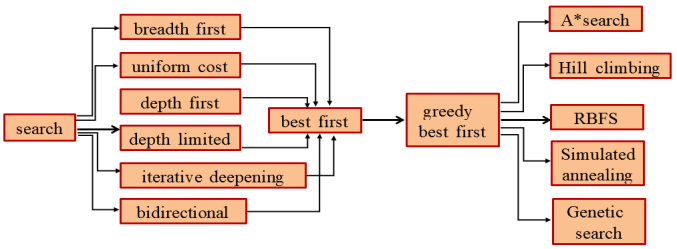


Figure 45.Hyper-Concept Graphs Example (2)

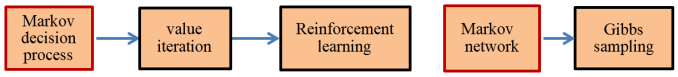


Figure 46.Hyper-Concept Graphs Example (3)

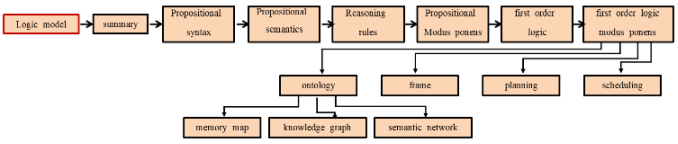


Figure 47.Hyper-Concept Graphs Example (4)

9.1.3. Basic Concept Graphs

The basic concept graphs are further decomposed into vertices of the hyper-concept graphs, which is defined as follows. $G_{sub} = (V_{subi}, E_{subi}) \approx G_{hyper} = (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{superl}) \approx G_{basic} = (V_{basick}, E_{basick}) = (V_{subi}^{superl,basicm}, E_{subi}^{superl,basicm})$, $i \in [1, N_{sub}]$, $j \in [1, N_{hyper}]$, $l \in [1, N_{subi}]$, $k \in [1, N_{basic}]$, $m \in [1, N_{subi,\ hyperl}]$

Let i, j, k, m be integers, N_{basic} be the total number of vertices of the basic concept graph, and $N_{subi,\ hyperl}$ are the number of children of vertices V_{subi}^{hyperl} of the hyper-concept graph. The basic teaching unit of tutoring is the basic concept. Eventually, in the intelligent tutoring system, all the instructional nodes of tutoring are composed of basic concepts. The selection list displayed on the left-most part of the user interface of the developing intelligent tutoring system is the basic concepts in the subject, not the chapters and sections in the subject, as in the step-by-step tutoring (Figure 49). Figure 48 shows an example of a basic concept graph corresponding to the nodes (V_{sub1}^{super1}) and (V_{sub1}^{super2}) (represented by a red circle) of the hyper-concept graph in Figure 44.

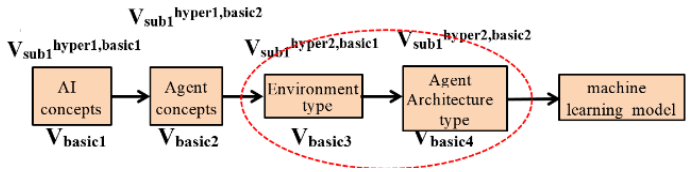


Figure 48.Basic Concept Graphs Example

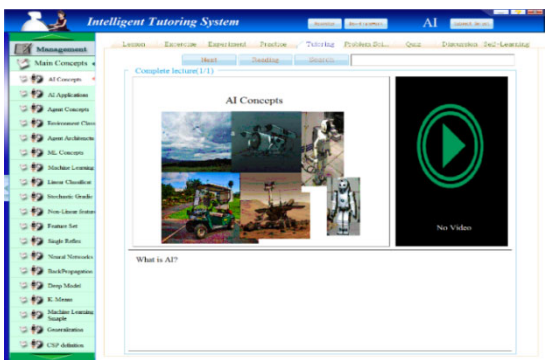


Figure 49.In Tutoring System, the Choice is a Basic Concept

9.1.4. Sub-concept Graphs and Help-concept Graphs

Figure 50 shows the sub-concepts and help-concepts between the basic concepts "Environmental Classification" and the basic concept "Active Structure Classification" in AI. In tutoring using only the knowledge graph, the actual tutoring flow of intelligent teacher as tutor is implemented by this graph. The computation of generating the specific tutoring paths for each student on the ITS platform is done using this graph.

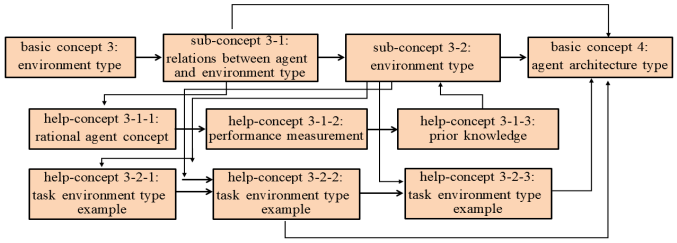


Figure 50.Sub-Concept and Help-Concept Graphs Example

9.2. A Hierarchical Partitioning Knowledge Graph Construction Approach using Graph Drawing Tool

The hierarchical partitioning approach and examples of the knowledge graph presented above are only documents on paper. This principle should be implemented in the ITS authoring tools. In this embodiment, there are ways to create databases directly, create xml documents, and create documents using a document editing function (which enables building hierarchical, partitioned knowledge graphs), a function of the authoring tools. We have already provided users with a tool that creates a hierarchical partitioning knowledge graph using the document editing function in the authoring tool (Figure 51). Many users have been asked to make a more convenient tool to create a hierarchical partitioning knowledge graph.

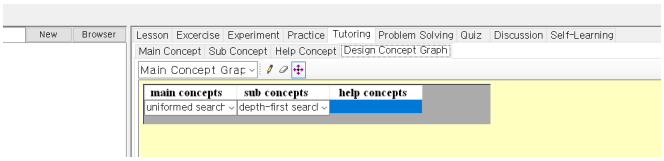


Figure 51. A Screenshot for Tools Creating Hierarchical, Partitioning Knowledge Graphs Using Document Editing Functions

We have updated the original authoring tool to create a knowledge graph using graphically and visualized graphics tools. When using this graphics tool, authors can create hierarchical partitioning graphs, just as they usually use only the mouse to draw pictures. Figure 52 below shows a scene constructed by drawing a knowledge graph by a mouse.

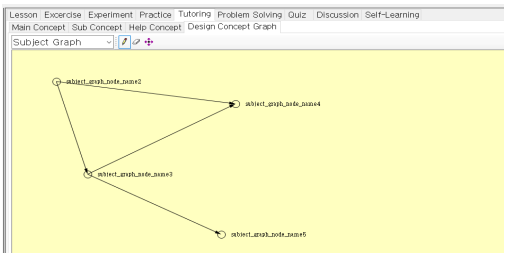


Figure 52.A Screenshot of Graph Drawing by Mouse

10. TUTORING CONTENTS AUTHORING AND CHECKING (PLAYING)

The above classifies intelligent tutoring systems as frameworks and platforms. This intelligent tutoring system also consists of student models, tutoring models, domain models, and interfaces. The tutoring model can be viewed pedagogically, systematically, and AI. The tutoring model is considered only from a pedagogical point of view as follows. The tutoring model plays the role of "how" in the system. The meaning of this 'how' is interpreted in two ways. One is the pedagogical mode, and the other are the instructional contents that is necessarily necessary or is considered to be a student's solution. In relation to the pedagogical approach, the terms Active Learning, Scaffolding, and Proactive Intervention are used. The term "active learning" does not mean that the student immediately gives the entire answer to the problem, but that the student himself will be able to elicit the answer by himself through the process of solving the problem. The term "scaffolding" is used to divide a problem that must be solved into several subproblems, meaning that each subproblem is solved by a student's effort completely through an internal iterative loop. For this reason, when using the term scaffolding, the term "Scaffolding for Problem-Solving" is often used. The term scaffolding also means that only when each subproblem is completely solved, does the solution to the other subproblem at its upper level. The system can interfere with the student's learning if the student's learning is challenged. This provides direct learning data related to the early state of the learning state that are currently facing difficulties or presents examples.

This is a "proactive intervention". This section presents the methods of intelligent tutoring authoring and checking using only step-by-step tutoring and knowledge graph. The step-by-step tutoring model mainly focuses on the application of pedagogical modes, which embody the method and principle of "active learning", "scaffolding", and "proactive intervention". Thus, as the pedagogical modes, hints, scaffolding for problem-solving, and proactive intervention are used.In our paper, we focus on predictive contents presentation in the construction of a tutoring model of an ITS using only knowledge graph. That is, based on predicting the current state of the student, we focus on implementing a method that presents learning problems or learning contents that the student is expected to be able to solve in the future.

10.1. Authoring and Checking (Playing)for Step-by-Step Tutoring

10.1.1. Step-by-Step Tutoring Principle

In the past, the research and development of intelligent tutoring systems has proposed step-by-step in the process of developing a research to implement the teaching methods of human teachers in machines (Vande Pol et al. [46]). This approach is a way of implementing a framework called KCDs(Knowledge Construction Dialogues), which guides all students according to a pre-planned "guide direction of inference", taking into account that students' abilities are different, KCD applies how to exit from the main course of the course and return to the main path before it, when the student has a wrong answer, for the sub-session for "therapeutic" teaching. This approach requires repeated learning for a long time and does not take into account the individual student's level of knowledge, which can be disappointing and inefficient for some students.

Our research and development of intelligent tutoring systems seeks to implement the step-by-step intelligent tutoring function that implements this step-by-step tutoring principle. We assume that the intelligent teacher using the step-by-step pedagogical approach performs the following instructional activities. The teacher constructs few static student models manually using his or her teaching experience and academic knowledge, apply a step-by-step approach to teaching, and return to the lesson when judged to have failed to understand the content of the current teaching. That is, the lecture on the subject that is not understood is repeated, we revisit the supplementary lecture on the wrong problem and then the previous question was re-presented and the 10 points were passed to continue the next topic. In this way, the teacher's unilateral teaching flow prevails because he is not in the mind of the student, the teacher is motivated to take education and the student is in a passive state. It is also a unilateral teaching mode that does not take into account each individual student, i.e., each student will have different learning efficiencies and different learning outcomes.

10.1.2. Step-by-Step Tutoring Contents Data Structures

The subject teacher edits the necessary data in text, creates it using the provided authoring tool, and then imports the created step-by-step tutoring contents into the ITS Checker (or Player) to test or evaluate it. First, the system designer must submit authoring specifications for the users of the authoring tools. The authoring specifications include data structure formulations and problem formulations for users of authoring tools. Next, the author creates data structures for his subject by editing it on a computer or writing it on a paper. Then, the author uses this data structures and the authoring tools to create step-by-step tutoring contents. The process flow from the presentation of system designer's authoring specifications to the creation of the tutoring contents is shown in Figure 53.



Figure 53.The Process Flow from Submitting Authoring Specifications to Creating Tutoring Contents

This data structures consist of domain model data, basic data of student model, and basic data of pedagogical model. In the data structure shown below, the labeled rows 1 refer to domain model data, the labeled rows 2 refer to basic data of student model, and the labeled rows 3 refer to basic data of pedagogical model.

Developers (programmers who develop authoring tools) refer to a database that is a product of authoring, in addition to data structures formulations for authoring tool users, and problem formulations for authoring tool users. Data structures formulations, problem formulations, and example data structures are as follows.

a. Data Structures Formulations
{[Topic 1(///Domain model data///)], [Question and correct Answer pair in topic 1 (QA pairs)(/// basic data of student model ///), Supplementary lecture material for the question and

answer pair in topic 1 (///basic data of pedagogical model ///)] (///There may or may not be a pair of question and correct answers on the topic ///)}, … , …, {[Topic n(///Domain model data///)], [Question and correct Answer pair in topic n (QA pairs)(/// basic data of student model ///), Supplementary lecture material for the question and answer pair in topic n (///basic data of pedagogical model ///)] (///There may or may not be a pair of question and correct answers on the topic ///)}, The question and answer pair (QA pair) in the section (///basic data of student model ///)}

b. Problem Formulations
The form of the problem is as follows.
Type 1: Single-choice mode
Type 2: Multiple selection
Type 3: Blank filling
Form 4: Combination of Combo-box and Blank Filling
Type 5: Response mode with natural language

c. Data Structure Examples
[Chapter 1] Basics
1: [Section 1] Concepts
/////// a .ppt (or a picture), necessary videos and comment texts on subject 1. Several .ppt (or pictures), several videos, and several comment texts are available. There must be at least one .ppt for the topic i. ///////////
/*---It's a ppt (or a picture) of topic 1 and domain model data. ----*/
1:(topic 1 - ppt 1) Slide1.png
/*--This is the comment text on the first ppt of topic 1. --*/
1:(Explanation 1) I will study the subject from this time.
/*--This is the second ppt of topic 1. --*/
1:(topic 1 - ppt 2) Slide5.png
/*---This is the comment text corresponding to the second ppt of topic 1. ---*/
1:(Explanation 1)The course of this subject is as follows.
/*---This is a video on topic 1. At this point, only one video is allowed. ---*/
1:(video 2)
/*---At this point, only one video is allowed. This is QA pairs in Subject 1. ---*/
2:(Question and correct answer pair in topic 1)
/*---Several problems can be presented. The form of the problem is of various forms. ---*/
/*---The following is a multi-choice problem. ---*/
2:[Question 1, 2] The English source of AI is artificial intelligence. What is the meaning of English artificial?
①imitation
②spurious
③human
[answer] ① 5 ② 5 ③ 0
/*---The following is a multi-choice problem. ---*/
2:[Question 2, 2] What is the meaning of English intelligence?
① the ability to acquire and apply knowledge
② the ability to think and reason
③ ability to understand and benefit from experience
[answer] ① 4 ② 3 ③ 3
/*---The following is a multi-choice problem. ---*/
2:[Question 3, 2] What are there in human intelligence?
① Human intelligence includes thinking, reasoning, oral speaking, ear listening, language, visual understanding, nose smell, and skin feeling.
② Intelligence of man involves intelligent behavior and action.
[answer] ① 5 ② 5
/*---The following is a multi-choice problem. ---*/
2:[Question 4, 2] Why do AI technologies and products need?
①To replace what man does
②To enhance human intelligence.
③To make an autonomous being like a human being.
[answer]① 3 ② 3 ③ 4
/*---The following is a combination of COMBOX and Blank Filling. ---*/
2:[Question 5, 4] Theory and development of ②? systems that perform tasks of increasing ① such as perception, classification, learning, abstraction, reasoning, and behavior
[answer]① human' ability, machine intelligence ② computer
/*---The following is a natural language-based response model. -*/
2:[Question 6, 5] What is AI defined in a broad and general sense?
/*---This are the supplementary lecture materials, a basic data for tutoring model, corresponding to the QA pairs in topic 1. The format is the same as the domain model data type.---*/
3:(ppt 2-1) Slide6.png
3:(Explanation 2-1) Artificial intelligence means counterintelligence.
3:(video 2-1) 1-1.mp4
3:(ppt 2-2) Slide7.png
3:(Explanation 2-2) Human intelligence includes perception, classification, learning, abstraction, reasoning, behavior, and so on.
3:(video 2-2) 1-1.mp4
3:(ppt 2-3) Slide7.png
3:(Explanation 2-3) Artificial intelligence technology and products began to develop from the human needs to make machines that replace human intelligence and function or implement them at a higher level, and then, as technology and computing resources rapidly evolve, they are moving towards the realization of the desire to develop fully autonomous mechanical agents.
3:(video 2-3)
3:(ppt 2-4) Slide7.png
3:(Explanation 2-4) AI is about the theory and development of computer systems that, to a narrow extent, perform tasks that augment human intelligence such as perception, classification, learning, abstraction, reasoning, and behavior, and to develop fully autonomous agents in a wide range.
3:(video 2-4)
……
/*--- It's a ppt (or a picture) of topic 2. ---*/
1:(topic 2 - ppt 1) Slide8.png
1:(Explanation 1) .....
1:(video 1)
/*---This is the final test entry for the section. It is basic data for tutoring model.---*/
2:(Chapter 1. Section 1. Final test)
……

10.1.3. Step-by-Step Tutoring Contents Database for Authoring Tools Developers

The Step-by-Step Tutoring Contents data structures used by the authoring tool and the checker is implemented using the database, which is as follows.

Table 2.Step-by-Step Tutoring Contents Database

Record name	Data type	main key	meanings
Chapter table			
ChapterID	INTEGER	√	Chapter number
ChapterTitle	TEXT		Chapter title
Section table			
ID	INTEGER	√	unique identifier of section
ChapterID	INTEGER		the number of the chapter in which the section belongs
SectionID	INTEGER		the number of the section in the chapter
SectionTitle	TEXT		section title
Lecture table			
comment: Each slide is represented as a line of this table and is separated by the value of the SectionID field. In order to obtain the slide information for Section 1.2, we obtain the unique identifier(ID) of the row corresponding to Section 1.2 in the table and return only the rows with the same value and SectionID.			
ID	INTEGER	√	unique identifier of slide
SectionID	INTEGER		Unique number of a section containing a slide, in what section is the slide?
SlideName	TEXT		"Subject name/group teaching/slide" Name of the picture file in the path (including the extension)
Comment	TEXT		comment
VideoName	TEXT		"Subject name/group teaching/slide" Name of a movie file in the path (including an extension)
ShowOrder	INTEGER		Numerical values for slide display order
hasTest	INTEGER		If there are questions for slide, then 1, 0, if no.
FinalTest table			
comment: In group teaching, there is a final question at the end of each section. The information is a table. As in this table, in all test-related tables, the test data is entered in XML format. The test editor that prints the test problem in XML has been developed and is therefore used. Each row represents one test problem. Thus, the way to obtain the necessary test data is as in the table above, so that ParentSection only obtains the same rows as the unique number of the section.			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSection	INTEGER		corresponding section number
Table for SlideTest			
comment: In the slide of the Lecture table, if "hasTest" is 1, there is a slide. Then, we get the same lines of ID value of the slide line and the same value of ParentSlide in the SlideTest table and then ask.			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSlide	INTEGER		unique number of the slide, The ID of slide in the Lecture table.
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/group teaching/slide"
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/group teaching/slide"

10.1.4. Step-by-Step Tutoring Contents Checking (Playing) Algorithm for Authoring Tools Developers

The step-by-step tutoring contents checking (playing) algorithm works in units of LS_t, each time a lecture topic.
[Algorithm 2. Step-by-Step Tutoring Contents Checking (Playing) Algorithm]

```
1:function Step-by-Step_Tutoring_Contents_Checking
(GT_MDB)
2:{
3:      for(i=0; i<n_max; i++) //n_max=max{n_s, n_e, n_v, n_q}
4:      {
5:              ith slide display;
6:              ith movie playback;
7:              for(j=0; j<n_q; j++)
8:              {
9:                      jth question;
10:                     jth answer;
11:                    while (result mark < threshold mark)
12:                     {
13:                             Additional lecture on the
jth question;
14:                             jth query presentation ;
15:                     }
16:             }
17:     }
```

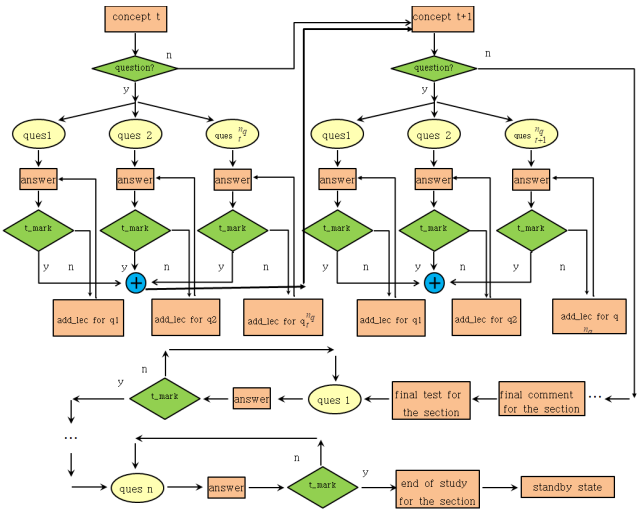


Figure 54.Step-by-Step Tutoring Contents Flow Diagram

10.2. Authoring and Checking (Playing) Tutoring Contents using only HP Knowledge Graphs

The tutoring systems using knowledge graph-only works as follows. The student model outputs the value of the learning process of the premise concept of the basic concept that the student has selected (requested) and the checker (player) or platform's tutoring model computes the learning path that is specific to each student based on this value.

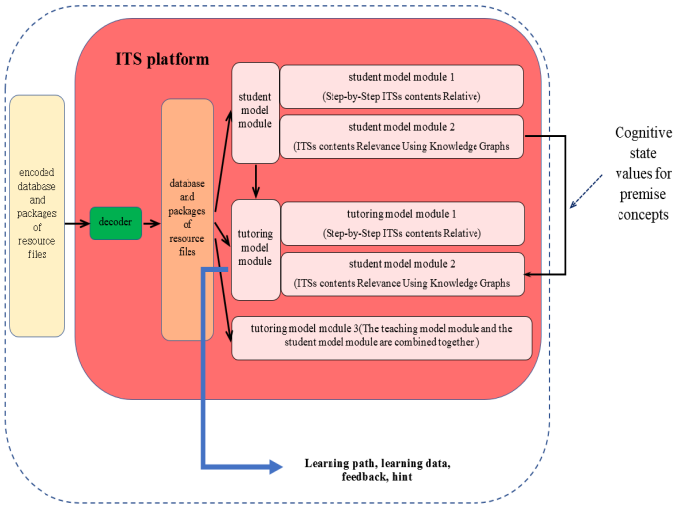


Figure 55.A Framework Diagram and Action Flow Diagram Corresponding to Tutoring Using only Knowledge Graph on the Proposed ITS Platform

10.2.1. Authoring Tutoring Contents using only Knowledge Graphs

1) A Review of the Checker (Player) or Platform in Tutoring Systems Using Knowledge Graphs only

To design the tutoring contents authoring tool that uses only knowledge graphs, we need to design the workflow on the checker (player) or platform in this respect, especially the learning path generation principle. We want to make the checker (player) or platform of a tutoring system that uses only the knowledge graph to work as follows. And we want to create the tutoring contents that enables this behavior in the authoring tool of the system using only the knowledge graph. The proposed action is as follows.

In an ITS that uses only a knowledge graph, it is assumed that a student requests a learning activity for one of the key concepts present in the subject. The following Figure illustrates the case when the student asked for a study of the basic concept of "Active Structure Classification".

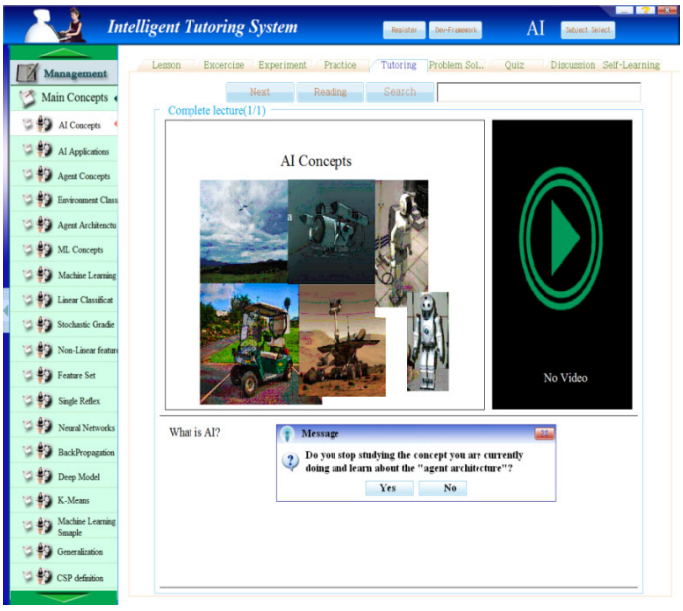


Figure 56.AScreenshot of Cases where a Student has Chosen the Basic Concept of "Active Structure Classification"

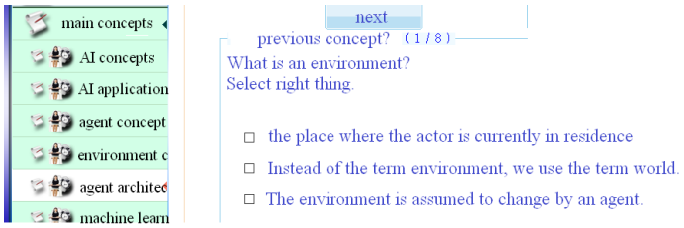


Figure 57.A Question Example to Understand the Perception of the Basic Concept of Environment Classification, a Premise Concept of the Basic Concept of Active Structure Classification

At this time, the intelligent tutoring system asks questions to understand the learner's perception of the immediate antecedent concept (antecedent concept) of the concept "Active Structure Classification", the current key concept on the knowledge graph. An example is shown in the Figure below. The following Figure illustrates an example of a question to understand the perception of the basic concept of "environment classification", a preconception of the basic concept of "Active Structure Classification". Of course, authoring tools do not want to impose a great restriction on the number of such questions. Thus, we have developed an authoring tool that enables the number of queries as many as we need.The next problem to be solved in the system design is how to do it after the results of the responses to these questions are given. The next system design is as follows. If the perception of the basic concept of a student's choice is scored at a score of 1.0, the system presents guided questions that can be derived by itself from the basic concept that the student requested (e.g., the basic concept of "Active Structure Classification"), then iteratively proceeds to the supplementary sessions according to the degree of response, and finally gives a rich enough lecture on the basic concept that the student requested (e.g., Active Structure Classification). The type of questions is five, as in the previous step-by-step tutoring. In this case, if the basic concept chosen by the student as an exception is the first root concept of the knowledge graph, there is no clear understanding of the previous premise concept, only the elicited questions, supplementary lectures, and a complete lecture structure.Finally, we need to design a system for cases where the proficiency of the basic concept of the student's request is below the score of 1.0. In this case, the design should be designed to allow individual instruction to begin with the premise of the basic concept requested. The system then computes the individual tutoring paths probabilistically using the sub-concepts and help-concept graphs that exist between the premise concept and the basic concepts that the student currently chooses.

To implement such a system design, the data structure design corresponding to the basic concepts of knowledge graph and the data structure design corresponding to the sub-concepts and help-concepts must be preceded.

2) Data Structure Design Corresponding to the Basic Concepts of Knowledge Graphs

In the following, we show an example of a data structure corresponding to the basic concept of knowledge graph in the same processing flow as above.

/*---A data structure example corresponding to the basic concept of knowledge graph ----*/

(basic concept 4) A classification by structure of the agents

/*---------A question to judge the cognitive performance of basic concept 3 ----------*/
[Question 1,1] What is an environment? Choose the right ones.
a. Environment? Where the agents inhabit.
b. Instead of the term environment, we also use the term world.
c. The environment is assumed to change by agents.
[Answer] a. 3, b. 4, c. 3
[Question2,1] By definition of the concept of the agents, is there a link between the environment and the agents?
1 Related
2. No correlation
a. Related.
b. No correlation.
[Answer] a. 10, b. 0
[Question3, 4] What is the link between the agents and the environment? Try to use your ideas.
[Answer(suggestion)] a. perception, b. action
[Question4, 5] An agent is said to perceive data or information from its surroundings, but what types of perception or sensing of such data?
[Answer(suggestion)] Eyes, vision (images), ears, auditory sounds (voice, speech), nose, smell, tongue, skin
/*-Guided questions and supplementary lecture on "Classification according to the structure of an agent". ---*/
[Question4-1, 5] An agent is called a mechanical entity that acts on its surroundings after acquiring perceptual data from its surroundings using sensors. An agent is implemented in a program, which is what will be implemented? In this respect, redefine the agent.
[Answer(suggestion)] architecture, agent program
/---*A supplementary lecture on Question 4-1. The instructional model then returns Problem 4-1 after this supplementary session, and this cycle is repeated until the score for question 4-1 is 1.0. ----*/
(explanation 4-1) An agent is useless if it has a program. It must have a mechanism to operate, which is called the architecture. In detail, when considering architectures, we consider the CPU or GPU that indicates computational power and the memory capacity that indicates storage power.
(video 4-1)
[Question4-2, 2] There is a very simple way to implement an agent program. What would you like?
[Answer] a. 3, b. 3, c. 4
(ppt 4-2) slide1.png
(explanation 4-1) …….
(video 4-1) …….
//////////////////
/*------A complete lecture example of "Classification according to the structure of agents" --*/
(slide 4-3) slide10.png
(explanation 4-3) ……
(video 4-3) ……
(slide 4-4) slide11.png
(explanation 4-4) ……
(video 4-4) ……
//////////////////////

3) Data Structure Design Corresponding to the Sub-Concepts and the Help-Concepts of Knowledge Graphs

In the following, we show examples of data structures corresponding to the sub-concepts and help-concepts of the knowledge graph in the same processing flow as above.
/*--- Data structure design examples corresponding to the sub-concepts and the help-concepts of knowledge graphs ----*/
/*-------A example of data structures of sub-concepts and help-concepts that exist between the basic concepts "Environment Classification" and "Active Structure Classification" in Figure 48. ----------*/
(sub-concepts 3-1) the relationship between the agents and the environment
(ppt 3-1-1) Slide2.png
(explanation 3-1-1)… …
(video 3-1-1)
(Questions for Slides 3-1-1)
[Question 1, 1] …
[Question 2,2] …
(help-concept 3-1-1) rational agent concepts
(ppt 3-1-1-1) Slide5.png
(explanation 3-1-1-1) ……
(video 3-1-1-1)
(Questions for Slides 3-1-1-1)
[Question 1, 2] ……
(help-concept 3-1-2) …….. ///(ppt 3-1-2-1) Slide5.png///(explanation 3-1-2-1) …… ///(video 3-1-2-1) ///(Questions for Slides 3-1-2-1)[Question 1, 2] ……
(sub-concepts 3-2) environment specifications///(ppt 3-2-1) Slide2.png///(explanation 3-2-1)… … ///(video 3-2-1)///(Questions for Slides 3-2-1)///[Question 1, 1] …///[Question 2,2] ……///(ppt 3-2-2) Slide32.png///(explanation 3-2-2)… … ///(video 3-2-2)///(Questions for Slides 3-2-2)///[Question 1, 2] …///[Question 2,2] ……///……///(ppt 3-2-n) Slide32.png///(explanation 3-2-n)… … ///(video 3-2-n)///(Questions for Slides 3-2-n)///[Question 1, 2] …///[Question 2,2] ……///
//////////////////////////////////////
(help-concept 3-2-1)
//////////////////////////////
(help-concept 3-2-2)
////////////////////////

10.2.2. An Acyclic Violation Checking Algorithmfor Hierarchical Partitioning Knowledge Graph Construction

When constructing a hierarchical partitioning knowledge graph using a graph drawing tool, acyclic violations can occur. This can happen when adding a new node or a new edge to the existing graph. This means that we have to implement an acyclic violation checking algorithm when developing a graph drawing tool. In general, these algorithms use a method to check whether a reverse path from v to u occurs when creating a new arc u → v [69].

1) Acyclic Violation Checking Algorithms
The following are acyclic violation detection algorithms [73]: depth-first search, topological classification based on the Kahn algorithm (Topological Sort-based on Kahn's Algorithm), and Reachability Analysis. An application example is an acyclic violation test in Compiler Construction.

(1) Depth-First Search
Acyclic detection by depth-first search involves depth-first search (or color-based method) with Back-Edge detection and recStack[71].

a. Depth-First Search with Back-Edge Detection

This algorithm is also called loop detection in directed graphs using color. This algorithm performs a depth-first search

starting from v when a new edge (u, v) is added, checking whether it is reachable from v to u. The states of the state space are classified into three categories. That is, we distinguish between the states that have not yet been visited (marked in white), the states currently visited (marked in gray), and the states that have already been visited (black) if all processing of the node and all its child nodes has been completed. Here, the currently visited states are represented by the nodes present in the current recursive stack. The previously visited states are represented by nodes that are called in the open list and have already finished processing and their child nodes. The system judges that there exists a loop when it encounters a node that exists in the current search tree during the depth-first search. The time complexity of this algorithm is O (V + E), and the space complexity is O (V).

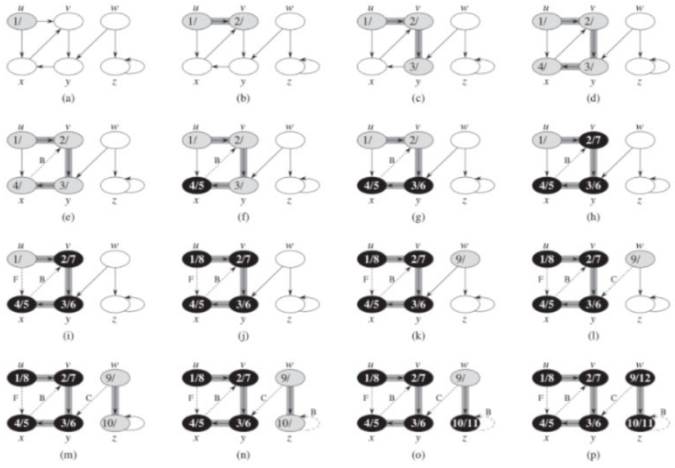


Figure 58. Depth-First Search with Back-Edge Detection in Directed Graphs (from [71])

Figure 58 shows the cycle generation test process by depth-first search in the directed graph. The edges searched by the algorithm are shaded by a solid line if the edges are the edges of the search tree, and are indicated by a dashed line if not the edges of the search tree. The edges denoted by B, C, and f denote back, cross, and forward edges. Numbers displayed inside the vertices indicate the search and the end times.

The acyclic violation checking algorithm using depth-first search, expressed as pseudo-code, is as follows.

[Algorithm3: An Acyclic Violation Checking Algorithm with Depth-First Search with Back-Edge Detection] (from [70])

DFS (G)
1: for each vertex $u \in G$, V
2:u. *color* = WHITE
3:$u.\pi$ = NIL
4:*time* = 0
5: for each vertex $u \in G$, V
6: if u. *color* ==WHITE
7:DFS-VISIT (S, u)

DFS-VISIT (S, u)
1:*time* = *time* + 1 // a just-searched white vertex u
2:u. d = *time*
3:u. *color* = GRAY
4: for each $v \in G$. *Adj* [u]// Search for the edge (u, v).
5:if v. *color* == WHITE
6:v. $\pi = u$
7:DFS-VISIT (S, v)
8:u. *color* = BLACK// The vertex u, which has just been searched, is marked in black.
9:*time* = *time* + 1
10:u. f = *time*

(2) Topological Sort based on Kahn's Algorithm

The search principle of this algorithm is as follows. the algorithm first computes the number of edges (denoted in-degree) that are entered for each node, and adds to a queue all nodes with an in-degree value of 0. Then, the algorithm iteratively deletes a node from the queue, which deletes all the edges that start and extend from that node, and reduces the in-degree value of its neighbors. If the in-degree value of a neighbor is 0, add that node to the queue. Consequently, if the number of processed nodes is less than the total number of nodes present in the graph, then we judge that there exists a loop. The time complexity of this algorithm is O(V + E).

(3) Reachability Analysis

This method is called incremental checking. When a new edge $v \rightarrow w$ is added, the algorithm checks whether w could already be reached in v before it.If w can be reached at v, v → w generates a cycle. To determine whether v is reachable, algorithm run either a breadth-first search algorithm or a depth-first search algorithm starting from w.

(4) Compiler Construction

As an example of an application of acyclic violation checking, we can take the directed acyclic graphs (DAGs) construction example in compiler design.

2) Acyclic Violation Checking forHierarchical Partitioning Knowledge Graph Construction

As we have already considered above, the problem arises if we use visited [ ] only in depth-first search. The solution is to use a further definition of the array recStack[] in addition to the array called visited[].

We propose an acyclic violation checking method using recStack to construct hierarchical partitioned knowledge graph. Our proposed acyclic violation checking method in hierarchical partitioned knowledge graph construction is a modification of the acyclic violation checking method using recStack, which was previously proposed. Below is an acyclic violation checking algorithm on the subject graph.

[Algorithm 4: AcyclicViolation Detection Algorithm Using recStack in DFS (on the Subject Graph)]
input: subject graphG_{sub}= (V_{subi}, E_{subi}), adj(v)
output: cyclic true or false

isCyclic(Vector adj in G_{sub})

// Function to detect cycle in a directed graph
function isCyclic (Vector adj)
{
V. visited = false // The visit information for neighboring nodes is denoted by false.
V. recStack = false // The recStack information for neighboring nodes is denoted by false.

```
// Run DFS from every unvisited node
for each every unvisited node i□ V
 {
if i. visited = false &&isCyclicUtil (adj, i, visited, recStack) = true then do
return true
}
return false
}
-----------------------------------------------
// Utility DFS function to detect cycle in a directed graph
function isCyclicUtil (Vector adj, v, u, visited, recStack)
 {
 // node is already in recursion stack cycle found
if u. recStack is true then do
return true

 // already processed no need to visit again
if u. visited is true then
 return false

u. visited = true
u. recStack = true

// Recur for all adjacent nodes
for each v □ adj[u]
 {
if isCyclicUtil (adj, v, visited, recStack) is true then do
return true
}
 // remove from recursion stack before backtracking
 u. recStack = false
return false
}
```

The acyclic violation checking algorithm when the knowledge graph is hierarchically partitioned is as follows.

[Algorithm 5: Hierarchically Knowledge Partitioning and Acyclic Violation Detection Algorithm]

/*----Creating a subject graph ---*/
Creating $G_{sub}=(V_{subi}, E_{subi})$
G_{sub}– isCyclic // Performing an acyclic violation detection on the subject graph
/*----Creating a hyper-concept graph by dividing each vertex of the subject graph. ---*/
for each v □V_{subi}
Creating$G_{sub}= (V_{subi}, E_{subi}) \approx G_{hyper}= (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{hyperl})$
G_{hyper}– isCyclic // Performing an acyclic violation test for hyper-concept graphs
/*----Creating a basic concept graph by dividing each vertex of the hyper-concept graph. ---*/
for each v □V_{hyper}
Creating$G_{sub}= (V_{subi}, E_{subi}) \approx G_{hyper}= (V_{hyperj}, E_{hyperj}) = (V_{subi}^{hyperl}, E_{subi}^{hyperl})$
$\approx G_{main}= (V_{maink}, E_{maink}) = (V_{subi}^{hyperl, mainm}, E_{subi}^{hyperl, mainm})$
G_{main}– isCyclic // Performing an acyclic violation detection on a basic concept graph
/*----Creating a sub-concept graph by dividing each vertex of the basic concept graph. ---*/
for each v □V_{main}
Creating a sub-concept Graph $G_{sub\text{-}concept}$
$G_{sub\text{-}concept}$ – isCyclic // Performing an acyclic violation detection for a sub-concept graph
/*----Creating a help-concept graph by dividing each vertex of the sub-concept graph. ---*/
for each v □$V_{sub\text{-}concept}$
Creating a help-concept Graph $G_{help\text{-}concept}$
$G_{help\text{-}concept}$ – isCyclic // Performing an acyclic violation detection for a help-concept graph

10.2.3. Tutoring Contents Checking (Playing) Using Knowledge Graphs only

To design and implement the authoring tool, checker and platform of an ITS using only the knowledge graph, we design and implement the student model module 2 and the tutoring model module 2 in Figure 55.

1) A Design of Student Model Module 2

This leads to the problem of designing and implementing a method to quantify the degree of awareness of the premise concept.
Let us consider the student's perception of the basic concept of time t to be TD.

Then, if the score TD = 1.0 is called <A_M = 1 interval(ex)>, if TD = [0.8,1), then it is said to be <A_M = 2 interval(v_g); if TD = [0.6,0.8), then it is said to be <A_M = 3 interval(g); if TD = [0.4,0.6), then it is <A_M = 4 interval(aver); if TD = [0,0,0,4], then it is <A_M = 5 interval(fail)>.

2) Tutoring Contents Database Using Knowledge Graphs Only
The tutoring contents database using only the knowledge graph is as follows.

3) A Design of the Tutoring Model Module 2
This design leads to the problem of proposing and implementing a learning path generation algorithm.

a. Learning Path Generation Algorithm
[Algorithm6. Learning path generation algorithm]
1: if ($V_{basic\ k\text{-}1}$ = ex)
2: {
3: (If the student's learning state prediction result is ex, then the derived query for the *k*-1th concept is repeated until the score is 1.0, then a rich and complete lecture on the *k*-1th concept is given);
4: }
5: else if ($V_{basic\ k\text{-}1}$ = v_g)
6: {
7: (Choose the shortest path among paths consisting of the sub-concepts between the *k*-1th concept and the *k*th concept);
8: }
9: else if ($V_{basic\ k}$ = g)
10: {
11: (choose a path among *m*-1 paths consisting of sub-concepts between *k*-1th concept and *k*th concept using probabilistic inference algorithm);
12: }
13: else if ($V_{basic\ k}$ = aver)
14: {
15: For a student with this value, we generate paths that include the sub-concept and help-concept, and then also divide the number of total subpath as above and generate the learning path according to the corresponding probability value.
16: }
17: else
18: {
19: Make a choice of the *k*-1th basic concept.
20: }

Table 4. Tutoring Contents Database Using Knowledge Graphs Only

Record name	Data type	Main key	Explanation
Concept table			
ID	INTEGER	√	unique identifier number
ConceptName	TEXT		concept name
ConceptType	INTEGER		concept type: basic concept-1, sub-concept-2, help-concept-3
RootConcept	INTEGER		root basic concept ID: this value is all zero, otherwise, the unique number of the root basic concept of the concept. For example, if a sub-concept B is a sub-concept between the basic concept A and the basic concept C, then the ID value of the basic concept A is the value of this field.
basic concept relational table			
Each row of the table represents each edge(relation) of the concept graph.			
ID	INTEGER	√	unique identifier number
BeginVertex	INTEGER		precedence ID
EndVertex	INTEGER		ID of the posterior concept
RootConcept	INTEGER		fundamental concept of the relation
Relationship	INTEGER		What are the relationships between concepts. If either the front or rear nodes is a sub-concept, then 2. if one of the two is the basic concept, and one is the sub-concept, then 1. If both concepts are basic, then 0
Lecture table			
Comment: The slide is represented as a line of this table and are separated by the values of the ConceptID field. That is, to get the slide information for concept A, we get the unique identifier (ID) of the row corresponding to concept A in the concept table and return only the rows that are equal in value and ConceptID.			
ID	INTEGER	√	unique identifier of slide
ConceptID	INTEGER		Unique number of concepts belonging to slide is it an slide that reflects which concept.
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/individual teaching/slide".
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/individual teaching/slide".
ShowOrder	INTEGER		Numerical values for slide display sequences
hasTest	INTEGER		If there is a question for slide, then 1, 0, if no.
ReTest			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentConcept	INTEGER		The ID of the parent concept, which concept is the question of understanding.
PreTest			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentConcept	INTEGER		In the Concept Table, the ID of the concept, i.e., is it a leading question that reflects which concept
SlideName	TEXT		The name of the picture file (including extension) in the path of "the subject name/individual teaching/slide".
Comment	TEXT		comment
VideoName	TEXT		The name of the movie file (including extension) in the path of "the subject name/individual teaching/slide".
SlideTest (A understanding rate for a slide)			
ID	INTEGER	√	unique identifier
XMLData	TEXT		XML data for testing
ParentSlide	INTEGER		The ID of the slide, i.e., is it a leading question that reflects which slide.

b. A Description of Line 12 of Algorithm 6

A description of the stochastic path selection method for intervals 3. For the A_M=interval 3(0.6<= TD < 0.8), we randomly choose a path among m-1 paths consisting of the sub-concepts between the nth concept and the n+1th concept. If the distance is between 0.6 and 0.8, divide 1.0 by (n = total sub-learning path m −1) and find the ratio value that one learning path occupies in the whole interval 1. For example, if we have five sub-learning paths, then dividing 1.0 by 4 yields a value of 0.25. This means that each learning path (n = total sub-learning path m-1) has a ratio of 0.25 in the interval (actually, the interval between 0.6 and 0.8). Then, n sub-learning paths are ordered from one to the next, depending on the number of arcs present in each path, from the number of arcs to the least. The probability values of each of the four intervals are expressed as 0.6+0.25×0.2×1=0.6+0.05=0.65, 0.6+0.25×0.2×2=0.6+0.1=0.7, 0.6+0.25×0.2×3=0.6+0.15=0.75, 0.6+0.25×0.2×4=0.6+0.2=0.8.

The value of 0.25 is the ratio of one path to the path consisting of n (m-1) sub-concepts. This number varies with the number of paths that consist of sub-concepts. The value of 0.2 is the value that makes the interval between 0.6 and 0.8 equal to 1.

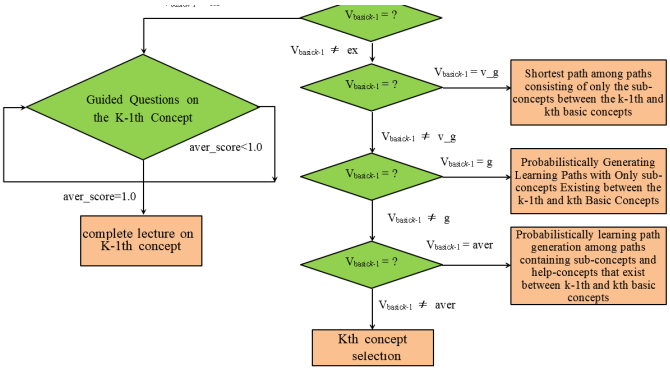


Figure 59.Learning Path Generation Flow Diagram

11. AN ITS BUILDING USING HIERARCHICAL POMDP

The tutoring model using Partially Observable Markov Decision Process (POMDP) provides a mathematical framework for intelligent tutoring systems to decide the optimal teaching strategy in uncertain environments.Unlike standard models that assume that student knowledge is fully observable, POMDP treats student's actual state as an internal state, such as a "hidden variable" that must be deduced from observable behaviors, such as question responses or response times.

POMDP is known as an effective way to deal with the uncertainty of the student's learning state and the uncertainty of the intelligent tutoring system behavior.In addition, the construction of the intelligent tutoring system with POMDP has the advantage of being able to update the system performance in real time and online [72].

In particular, the advantages of using the POMDP approach in the development of intelligent tutoring systems are as follows.

- Personalization: This approach allows tutors to provide personalized support in real time using individual learning curves of individual learners.
- Far-looking planning: Unlike instant reaction systems, POMDP can plan several steps far away.
- This time, it is not possible to provide immediate and positive rewards, but to select actions that lead to success in long-term learning.
- Handling uncertainty: This method allows us to deal robustly with uncertain noise data.

Intelligent tutoring systems using hidden Markov models and partially observable Markov Decision processes have a close connection with knowledge graphs. Hidden Markov models and partially observable Markov decision processes are known to be very effective in modeling uncertainty about the learner's learning state, but they face problems such as sparsity of data, cold-start problems of the system, lack of interpretability, etc.The knowledge graph solves these problems by its nature as structuring, semantics, and relativism.For this reason, knowledge graphs are integrated with the hidden Markov model, POMDP.In this paper, we propose methods to solve the problems when constructing a hierarchical POMDP based on a hierarchical knowledge graph, rather than using POMDP and knowledge graph simultaneously.In particular, the proposed system is a system that reproduces the classroom teaching style of human teacher.

This problem can be divided into the problems of developing an intelligent tutoring system authoring tool and the problems of developing a platform.First, the authoring tool must develop a domain model and a student model.Then, we need to identify how to convert the previously constructed knowledge graph to POMDP.In particular, we need to identify how to convert hierarchical knowledge graphs into hierarchical POMDPs.And the initial values for the parameters of the POMDP model must be determined.We also need to add a function that automatically determines and deletes invalid states.Next, we need to develop a tutoring model for the platform (or tester and player).The tutoring model should be developed to compute the learning path based on the Q learning theory, etc.And we have to solve the computational complexity problem.

In this paper, we will set up and solve the following research items in POMDP-based ITSs research and development that reproduces the classroom teaching style of human teacher.

a. An Action Principle of an ITS Using Hierarchical POMDPUI
b. An Architecture and UI of ITSs using hierarchical POMDP
b.1. Total architecture of the system and UI
b.2. An architecture and UI of Authoring Tools
b.3. An architecture and UI of Platform (or Checkers and Players)
c. Semi-Automatic Transformation of Hierarchical Partitioning Knowledge Graph to POMDP
d. A approach for determining the initial values of hierarchical POMDP parameters
e. Computing strategy and real-time online improvement method in hierarchical POMDP
f. A method to reduce computational complexity in hierarchical POMDP

To build an intelligent tutoring system using POMDP, we first need to perform the POMDP formalism.That is, we must define the internal components of the POMDP tutoring model.The POMDP tutoring model is completely defined by a set of parameters representing the interaction between tutoring machine and learner [72].That is, POMDP is defined by a set of elements $(S, A, T, \rho, O, Z, b_0)$[73].

- Here S is a set of states, representing the learner's internal hidden state condition, including the learner's emotions and psychological states, such as the degree of knowledge proficiency (e.g., the basic concept A is known, the sub-concept B is mistaken) and the confidence or frustration.
- A is a set of actions that give appropriate hints, present questions, or present the learning content that the student will perform.
- $T: S \times A \times S \rightarrow [0,1]$ defines the state transition probability. This defines the probability that a learner will transition from one learning state to another after a particular tutoring activity of the tutoring machine. Here T is the first character of the term transition and we will have to use this T in the state transition probability notation, but since the recognized probability marker P already exists, we denote the transition probability by $P(s'|s,a)$ without labeling $T(s'|s,a)$ separately. That is, $P(s'|s,a)$ is the probability of transition from s to s' after performing an action a in a state s. For each s and a pair, $\sum_{s' \in S} P(s'|s,a) = 1$.
- $\rho: S \times A \times S \rightarrow \Re$ is the compensation function, and $\Re$ is set of the reward values. $\rho(s,a,s')$ returns the compensation value after reaching s' after performing a at s.When the current knowledge concept is completed, it gives a high reward value, while when the response time is long or the response is discarded, it gives a low reward value or penalty value.
- O is a set of observations. As a visually observable response of the learner, the correct response, the wrong response, and the request to help are considered as noise signals representing the belief state.
- $Z: A \times S \rightarrow O$ defines the observation probability.This is the probability value given a specific observation (e.g., incorrect answer) given the student's current hidden state. $P(o|a,s')$ represents the probability that an agent observes o after performing a and reaching state s'.For each a and s' pair, $\sum_{s' \in S} P(o|a,s') = 1$.Note that $P(o|a,s')$ is not only s' but also a probability value of o given a and s'.
- b and b_0: b are the probability distribution values for all possible states, based on the learner's learning history, and the current optimal guess information of the tutor's machine on the learner's knowledge level.b_0 is the initial belief state.

The working principle of intelligent tutoring system using hierarchical POMDP is as follows.
To construct an intelligent tutoring system using partially observable Markov decision process, there must be uncertainty

in the system's behavior and uncertainty in the system's perception through sensing, and it must be formalized.That is, we have to define what uncertainty exists during system operation, what is the hidden state (variable) and the observed state (variable).To do this, the system must formulate what actions it performs and what it perceives.

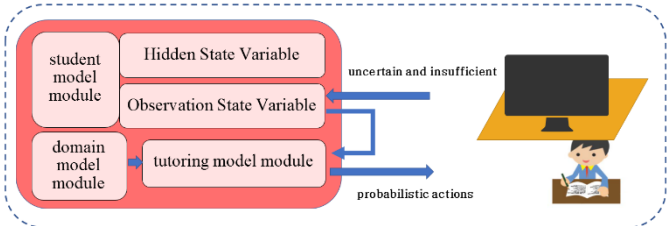


Figure 60. Uncertainty in perception and action of ITSs using POMDP

It is assumed that the intelligent tutoring systems using POMDP, which is a reproduction of human classroom lessons, works as follows.

First, the student asks the intelligent teacher to learn something.This is done by clicking on the concept that the student requests in the UI ① zone of the interface in our intelligent tutoring system.This selection action of the student is entirely observable.What is the concept that a student has chosen?Concepts floating in the left region (zone 1) of the UI that use hierarchical POMPDs were inherited from the hierarchical knowledge graphs; therefore, these concepts are subject concepts of the subject graphs, hyper-concepts of the hyper-concept graphs, and basic concepts of the basic concept graphs.If we choose a subject concept of the subject graph, this subject concept is the goal of the student and the intelligent teacher, and if we choose the hyper-concept of the hyper-concept graph, the hyper-concept is the goal, and if we choose the basic concept, the basic concept is the target.

The intelligent teacher "recognizes" the student's cognitive state and performs the appropriate teaching activities.Here, we must define and implement "recognizing" and "teaching behavior".It must be determined whether or not the student can pass the course of his/her current knowledge concept completely, depending on the results of the examination of his/her cognitive state.It is then divided into a "fully observable" state, which assumes that the student's cognitive state is completely known, and a "partially observable" state.In this section of the paper, we describe a Markov decision-making approach under the assumption that "only partially observable" is known.The instructional behavior of an intelligent teacher is to achieve the goal of fully learning the concepts chosen by the student, i.e., the subject concept or the hyper-concept or the basic concept.

12. CONCLUSION

This paper presents a comprehensive set of challenges in building a one-to-one tutoring system that reproduces the classroom teaching style of human teachers.We analyzed the types of preceding tutoring systems and classified the development of intelligent tutoring systems into authoring tools, testers (or players), and platform development.The research has defined the development purposes, goals, and functions of the developed intelligent tutoring system through the specifications.The paper elaborates on the functions and modules of the proposed intelligent tutoring system through the structure of the developed ITS and the proposed UIs.The developed system consists of a step-by-step tutoringmodel, a tutoring model using only knowledge graphs, and a tutoring model using POMDP.The system also consists of authoring tools, players, and platforms.The developed authoring tool can use the visualized knowledge graph graphics tool to construct a hierarchically partitioned knowledge graph.Then, anacyclic violation checking algorithm was used.

We also briefly propose a method for building tutorial systems using POMDP.However, due to the surface constraints, the POMDP is not described in detail and only the problems to be solved are revealed.A detailed proposal for building tutoring models by POMDP is to be reported in a subsequent paper.This paper does not specifically set up an experimental project, as it is the main objective of the design proposal for the development of an intelligent tutoring system.

We define the state for POMDP based on the hierarchical knowledge graphs defined in Section 9.This state represents the knowledge of the intelligent tutoring systems (agents) about the knowledge state of the student.The state can be a subject concept, a hyper-concept, and a basic concept.Each state has two values, known and unknown.

The Intelligent Tutoring System executes a sequence of actions to provide a learning activity that allows the student to achieve the desired learning goal.At this time, the intelligent tutoring system asks the student to know exactly the student's cognitive state, i.e., the value of the hidden variable (state).

Abbreviations

ITSs	Intelligent Tutoring Systems
MDP	Markov Decision Process
HMMs	Hidden Markov Models
POMDP	Partially Observable Markov Decision Process
LMSs	Learning Management Systems
KG	Knowledge Graph
UI	User Interface
HPKGs	Hierarchical Partitioning Knowledge Graphs
HP	Hierarchical Partitioning
QA	Question and Answer
DAGs	Directed Acyclic Graphs

Author Contributions

Gwang-Hyok Kim: Conceptualization, Methodology
Yu-Il Kim: Software
Song-Hwan Kwon: Writing-original draft preparation, Project administration
Gyong-Jin Kim: Investigation, Software
Gang-Bom So: Software
Sin-Hye Jang: Verification
Ha-Gyong Kim: Resources, Validation
Jong-Nam Rim: Investigation, Software
Chung-Song Ko: Software, Formal analysis
Song-Il Pak: Writing-review & editing
Chol-Jun Hong: Software, Data curation
Myong-Son Choe: Software

Conflict of Interest

The authors hereby declare no conflict of interest.

REFERENCES

1. Xiaohan Z. (2020). A Survey on Application of Knowledge Graph. IOP Conf. Series: Journal of Physics: Conf. Series 1487 (2020) 012016, https://doi.org/ 10.1088/1742-6596/1487/1/012016.
2. Chen, P., Lu, Y., Zheng, V.W., Chen, X., Yang, B. Knowedu (2018): A system to construct knowledge graph for education. IEEE Access 6, 31553–31563 (2018).
3. Khazanchi, R. (2021). Impact on Intelligent Tutoring System (ITS) on Mathematics Achievement Using ALEKS, SPRINGER NATURE Link, 2021, pp 471-475.
4. V, T, K. A. (2021). Intelligent Tutoring Systems for Personalized Mathematics Learning: A TPACK Perspective, SPRINGER NATURE Link, 2021, pp 206-220.
5. Van K. L. (2026). ASSISTments Dataset 2009-2010, IEEE DataPort, DOI:10. 21227/xgb2-tb53.
6. Kinshuk (1996). Computer aided learning for entry level Accountancy students. PhD Thesis, De Montfort University, England, July 1996.
7. Corbett A. T., Koedinger, K. R., & Anderson, J. R. (1997). Intelligent Tutoring Systems. In M. G. Helander, T. K. Landauer, & P. V. Prabhu (Eds.), Handbook of human-computer interaction (pp. 849-874). Amsterdam: Elsevier.
8. Nwaha, H. S. (1990). "Intelligent tutoring systems: An overview". Artificial Intelligence Review 4(4): 251-277. doi: 10.1007/bf00168958. https://dx.doi.org/10.1007%2Fbf00168958.
9. Freedman, R. (2000). What is an intelligent tutoring system? Intelligence 11(3): 15-16. doi: 10.1145/350752.350756. https://dx.doi.org/10.1145%2F350752.350756.
10. Nkambou, R., Mizoguchi, R., &Bourdeau, J. (2010). Advances in intelligent tutoring systems. Heidelberg: Springer.
11. Mitrovic, A., Ohlsson, S. (2006) Constraint-Based Knowledge Representation for Individualized Instruction. Computer Science and Information Systems, 3(1), 1-22.
12. Ohlsson, S., Mitrovic, A. (2007) Fidelity and Efficiency of Knowledge representations for intelligent tutoring systems. Technology, Instruction, Cognition and Learning, 5(2), 101-132.
13. Anderson, H., Koedinger, M. (1997). intelligent tutoring goes to school in the Big City. International Journal of Artificial Intelligence in Education, 60(1), 264-272.
14. Sottilare, R. (2012). Considerations in the development of an ontology for a generalized intelligent framework for tutoring. In 12M defense and homeland security simulation Conference (DHSS 2012).
15. McLaren, B. M., DeLeeuw, K. E., & Mayer, R. E. (2011). Polite web-based intelligent tutors: Can they improve learning in classrooms? Computers & Education, 56(3), 574-584. doi: 10.1016/j.compedu. 2010.09.019.
16. ZerKouk, M. et al. (2025). A comprehensive Review of AI-based Intelligent Tutoring Systems Applications and Challenges, arXiv: 2507.18882v1.
17. Sghaier, G. et al. (2025). A systematic literature review to implement large language model in higher education: issues and solutions, Discover Education (2025) 4:35 | https://doi.org/10.1007/s44217-025-00424-7.
18. Carbonell, J. R. (1970). Mixed-initiative man-computer instructional dialogues (Ph.D. thesis), Massachusets, USA: Massachusetts Institute of Technology. Dept. of Electrical Engineering.
19. 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.
20. Clancey, W. J. (1979). Tutoring rules for guiding a case method dialogue. International Journal of Man-Machine Studies, 11(1), 25–49.
21. 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.
22. Pablo, A. G. et al. (2025). A framework for adapting conversational intelligent tutoring systems to enable collaborative learning, Expert Systems With Applications 271 (2025) 126663, https://doi.org/10.1016/j.eswa.2025.126663.
23. Kinshuk, and Ashok, P. (1997) A Conceptual Framework for Internet Based Intelligent Tutoring Systems. Knowledge Transfer, II, 117-24.
24. Tommaso, C. et al. (2024). Towards Educator-Driven Tutor Authoring: Generative AI Approaches for Creating Intelligent Tutor Interfaces, arXiv: 2405. 1471v1, https://doi.org/10.1145/3657604.3664694.
25. Priynka, S., Mayuri, H. (2022). Designing an intelligent tutoring system for computer programing in the Pacific, Education and Information Technologies (2022) 27:6197–6209, https://doi.org/10.1007/s10639-021-10882-9.
26. Erica, M. et al. (2001). ACTIVEMATH: A Generic and Adaptive Web-Bas ed Learning Environment, Carsten Ullrich DFKI Saarbr ¨ucken, D-66123 Saarbr ¨ucken, Germany November 20, 2001.
27. Muhammad, A. Et al. (2020). The transition from intelligent to affective tutoring system: a review and open issues, doi 10.1109/access.2020.3036990, ieee access.
28. Linqing, L. et al. (2023). Knowledge Graph Enhanced Intelligent Tutoring System Based on Exercise Representativeness and Informativeness, arXiv: 2307.15076v1.
29. Guixia, W., Zehui, Z., Shouyuan, Q. (2025). Synergizing Knowledge Graphs and LLMs: An Intelligent Tutoring Model for Self-Directed Learning, Educ. Sci. 2025, 15, 1102 https://doi.org/10.3390/educsci15091102.
30. Chenxi, D. et al. (2023). How to Build an Adaptive AI Tutor for Any Course Using Knowledge Graph-Enhanced Retrieval Augmented Generation (KG-RAG), arXiv:2311.17696.
31. Mei-Ling, C. et al. (2026). Knowledge Graph–Enhanced Retrieval for Personalized AI Tutoring, arXiv:2611.17696.
32. Tom, M. (1999). Authoring Intelligent Tutoring Systems: An Analysis of theState of the Art, International Journal of Artificial Intelligence in Education (1999), 10, 98-129.
33. Vincent, A. et al. (2025). An Integrated Platform for Studying Learning with Intelligent Tutoring Systems: CTAT+TutorShop, arXiv:2502.10395.
34. Kenneth H. et al. (2018). Opening Up an Intelligent Tutoring System Development Environment for Extensible Student Modeling, AIED 2018, LNAI 10947, pp 169-183, 2018, https://doi.org/10.1007/978-3-319-93843-1-13.
35. Abu Naser, S. (2016). ITSB: An Intelligent Tutoring System Authoring Tool, Journal of Scientific and Engineering Research, 2016, 3(5):63-71.
36. Aleven, V., Sewall, J., McLaren, B. M. & Koedinger, K. R. (2006). Rapid authoring of intelligent tutors for realworld and experimental use. In Kinshuk, R. Koper, P. Kommers, P. Kirschner, D. G. Sampson & W. Didderen (Eds.),

Proceedings of the 6th IEEE international conference on advanced learning technologies (ICALT 2006) (pp. 847-851). Los Alamitos, CA: IEEE Computer Society.
37. Robert A. S. (2014). Examining Opportunities to Reduce the Time and Skill for Authoring Adaptive Intelligent Tutoring Systems, Proceedings of the 2nd Annual GIFT Users Symposium (GIFTSym2), 44-48.
38. MacLellan, C. J., Wiese, E. S., Matsuda, N. & Koedinger, K. R. (2014). SimStudent: Authoring Expert Models byTutoring. GIFTSym2, Pittsburgh, PA.
39. Diego, D. et al. (2018). Authoring Tools for Designing Intelligent Tutoring Systems: A Systematic Review of the Literature, Int J ArtifIntellEduc (2018) 28:336–384, https://doi.org/10.1007/s40593-017-0157-9.
40. Aleven, V., McLaren, B. M., Sewall, J., & Koedinger, K. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary evaluation of efficiency gains. In M. Ikeda, K. D. Ashley, & T. W. Chan (Eds.), Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS 2006), (pp. 61 -70).
41. Keith, B. (2015). Authoring Tools for Adaptive Training – An Overview of Systems and Taxonomy for Classification, Springer International Publishing Switzerland 2015, D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2015, LNAI 9183, pp. 562–569, 2015. DOI: 10.1007/978-3-319-20816-9_54.
42. Robert, S. et al. (2017). Defining Complexity in the Authoring Process for Adaptive Instruction, Springer International Publishing AG 2017, D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2017, Part II, LNAI 10285, pp. 237–249, 2017. DOI: 10.1007/978-3-319-58625-0_17.
43. Benjamin, D. et al. (2014). AutoTutor and Family: A Review of 17 Years of Natural Language Tutoring, Int J ArtifIntellEduc (2014) 24:427–469, DOI 10.1007/s40593-014-0029-5.
44. Tom, M. (2016). Coordinating the Complexity of Tools, Tasks, and Users: On Theory-based Approaches to Authoring Tool Usability, Int J ArtifIntellEduc (2016) 26:37–71, DOI 10.1007/s40593-015-0076-6.
45. Fletcher, J. D. et al. (2018). Shared Mental Models in Support of Adaptive Instruction for Teams Using the GIFT Tutoring Architecture, Int J ArtifIntellEduc (2018) 28:265–285, DOI 10.1007/s40593-017-0147-y.
46. Vande Pol, J., Volman, M., Beishuizen, J. (2010). Scaffolding in teacher-student interaction: a decade of research. Educ. Psychol. Rev. 22, 271–296.
47. Robert A. et al. (2011). Using Learner Data to Influence Performance during Adaptive Tutoring Experiences, adfa, p. 1, 2011. © Springer-Verlag Berlin Heidelberg 2011.
48. Tiffany, P., Robert, S. (2011). Research to Practice: Adaptive Technologies and Approaches to Performance Improvement, adfa, p. 1, 2011. © Springer-Verlag Berlin Heidelberg 2011.
49. Qwaider, S. R., Abu-Naser, S. S. (2018). Excel Intelligent Tutoring System. In: Proceedings (2018).
50. Caleb, V. et al. (2022). Automated Assessment of Team Performance Using Multimodal Bayesian Learning Analytics, 2022 Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC), I/ITSEC 2022 Paper No. 22258 pp 1-14.
51. James, R. S. et al. (2015). Designing Representations and Support for Metacognition in the Generalized Intelligent Framework for Tutoring, D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2015, LNAI 9183, pp. 663–674, 2015. DOI: 10.1007/978-3-319-20816-9_63.
52. Benjamin, G. (2015). Augmenting Instructional Practice in GIFT Using the Engine for Management of Adaptive Pedagogy (EMAP), D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2015, LNAI 9183, pp. 595–604, 2015. DOI: 10.1007/978-3-319-20816-9_57.
53. Benjamin, G. (), Feedback Source Modality Effects in Game-Based Training: A Trade-Off Analysis, Proceedings of the Human Factors and Ergonomics Society 59th Annual Meeting – 2015.
54. James. R. S. et al. Demonstration: Using GIFT to Support Students'Understanding of the UrbanSim Counter Insurgency Simulation, Seventeenth International Conference on Artificial Intelligence in Education (AIED 2015), pp 17-23.
55. Robert A. S. (2011). Modeling Shared States for Adaptive Instruction, adfa, p. 1, 2011. © Springer-Verlag Berlin Heidelberg 2011.
56. Benjamin, G. et al. (2012). Use of Evidence-based Strategies to Enhance the Extensibility of Adaptive Tutoring Technologies, Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.
57. Keith, B. et al. (2012). Recommendations for Modern Tools to Author Tutoring Systems, Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.
58. Robert, A. S. et al. A Modular Framework to Support the Authoring and Assessment of Adaptive Computer-Based Tutoring Systems (CBTS), Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2012.
59. Colin, R., Stephen, G. (). Bringing Authoring Tools for Intelligent Tutoring Systems and Serious Games Closer Together: Integrating GIFT with the Unity Game Engine, Recommendations for Authoring, Instructional Strategies and Analysis for Intelligent Tutoring Systems (ITS): Towards the Development of a Generalized Intelligent framework for Tutoring (GIFT), AIED 2013 Workshops Proceedings Volume 7, pp37-40.
60. Robert, S., Benjamin, G. (2012). Designing Adaptive Computer-Based Tutoring Systems to Accelerate Learning and Facilitate Retention, Cognitive Technology, Volume 17, Issue 1, 2012.
61. Arthur, G. (2013). A Guide to Understanding Learner Models, Design Recommendations for Intelligent Tutoring Systems - Volume 1.
62. Goldberg, B., Sinatra, A., Sottilare, R., Moss, J., &Graesser, A. (2015). Instructional Management for Adaptive Training and Education in Support of the US Army Learning Model-Research Outline (No. ARL-SR-0345). Army Research lab Aberdeen proving ground md human research and engineering directorate.
63. Robert A. S. (2017). Challenges in Authoring, Instructional Management, and Evaluation Methods for Adaptive Instructional Systems, or publication in the Proceedings of the American Education Research Association (AERA) Conference Technology, Instruction, Cognition & Learning Special Interest Group (AERA TICL SIG) Symposium on Intelligent Tutoring Systems, BIG DATA–Learning Analytics, and Automated Humanlike Tutoring: Similarities and Differences in San Antonio, Texas, 28 April 2017.
64. Robert A. S. (2017). Adaptive instructional methods to accelerate learning and enhance learning capacity, Proceedings of the International Defense and Homeland Security Simulation Workshop, 2017, ISBN 978-88-97999-90-4; Bruzzone, Cayirci and Sottilare Eds.

65. Robert A. S. et al. (2018). Special Issue on the Generalized Intelligent Framework for Tutoring (GIFT): Creating a Stable and FlexiblePlatform for Innovations in AIED Research, Int J ArtifIntellEduc (2018) 28:139–151, DOI 10.1007/s40593-017-0149-9.
66. Scott O., Robert A. S., Keith, B. Task allocation and automation level in the design of intelligent tutoring system authoring tools, Proceedings of the International Defense and Homeland Security Simulation Workshop, 2017, ISBN 978-88-97999-90-4; Bruzzone, Cayirci and Sottilare Eds.
67. Benjamin S. G. et al. (2011). Enhancing Performance through Pedagogy and Feedback: Domain Considerations for ITSs, Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC) 2011, No. 11010, pp 3-12.
68. Robert A. S. (2018). A Comprehensive Review of Design Goals and Emerging Solutions for Adaptive Instructional Systems, Tech., Inst., Cognition and Learning, Vol. 11, pp. 5–38, © 2018 Old City Publishing, Inc.
69. Michael, A. B. et al. (2011). A New Approach to Incremental Cycle Detection and Related Problems, arXiv: 1112.0784v1.
70. Edith, C. et al. (2013). A Labeling Approach to Incremental Cycle Detection, arXiv: 1310.8381v1.
71. Armaël, G. et al. (2019). Formal Proof and Analysis of an Incremental Cycle Detection Algorithm, 10th International Conference on Interactive Theorem Proving (ITP 2019). Article No. 18; pp. 18:1–18:20, Leibniz International Proceedings in Informatics.
72. Wang, F. (2015). Handling exponential state space in a POMDP-based intelligent tutoring system, in Proc. 6th International Conference on E-Service and Knowledge Management (IIAI ESKM 2015), pp. 67-72, 2015.
73. Folsom-Kovarik, J. T., Sukthankar, G., and Schatz, S. (2013). Tractable POMDP representations for intelligent tutoring systems. ACM Trans. Intell. Syst. Technol. 4, 2, Article 29 (March 2013), 22 pages. DOI = 10.1145/2438653.2438664 http://doi.acm.org/10.1145/2438653.2438664.
74. Sottilare, R.A., Holden, H.K.: Motivations for a Generalized Intelligent Framework for Tutoring (GIFT) for authoring, instruction and analysis. In: AIED 2013 Workshops Proceedings, vol. 7, pp. 1–9 (2013)
75. Sottilare, R.A., Brawner, K.W., Goldberg, B.S. & Holden, H.K. (2012). The Generalized Intelligent Framework for Tutoring (GIFT). Orlando, FL: U.S. Army Research Laboratory – Human Research & Engineering Directorate (ARL-HRED), 2012.
76. Sottilare, R.A., Brawner, K. W., Goldberg, B., & Holden, H. (2013). The Generalized Intelligent Framework for Tutoring (GIFT). In C. Best, G. Galanis, J. Kerry & R. Sottilare (Eds.), Fundamental Issues in Defense Training and Simulation (pp. 223-234). Burlington, VT: Ashgate Publishing Company, 2013.
77. Yuan, S. et. al. (2025). Stitch: Step-by-step LLM Guide Tutoring for Scratch, arXiv: 2510.266334v1.
78. Wengran, W. et. al. (2020). Step Tutor: Supporting Students through Step-by-Step Example-Based Feedback, ITiCSE'20, June 15–19, 2020, Trondheim, Norway, https://doi.org/10.1145/3341525.3387411.
79. Anohina, A. (2007). Advances in Intelligent Tutoring Systems: Problem-solving Modes and Model of Hints, International Journal of Computers, Communications & Control Vol. II (2007), No. 1, pp. 48-55, https://doi.org/10.15837/ijccc.2007.1.2336.
80. VanLehn, K. (2006). The behavior of tutoring systems. International Journal of Artificial Intelligence and Education, 16, 227–265.
81. Ranganathan, R. et. al. (2014). What do students do when using a step-based tutoring system?, Research and Practice in Technology Enhanced Learning Vol. 9, No. 2 (2014) 323 - 347, Ó Asia-Pacific Society for Computers in Education.
82. Bilal, A. S., et al. (2024). A systematic literature review of knowledge graph construction and application in education, Heliyon 10 (2024) e25383, https://doi.org/10.1016/j.heliyon.2024.e25383.
83. Lina, F. (2025). Curriculum Content Construction and Updating for Intelligent Educational Systems Using Knowledge Graphs, Applied Mathematics and Nonlinear Sciences, 10(1) (2025) 1 -21, https://doi.org/10.2478_amns-2025-1075.
84. Fangju, W. (2014). POMDP Framework for Building an Intelligent Tutoring System, In Proceedings of the 6th International Conference on Computer Supported Education (CSEDU-2014), pa 233-240, DOI: 10.5220/0004801702330240.
85. Sergio, A. S. et al. (2021). Knowledge-Based Hierarchical POMDPs for Task Planning, arXiv: 2103.10642v1, https://doi.org/10.48550/arXiv.2103.10642.
86. Wenrui Z., Weidong C. (2021). Hierarchical POMDP Planning for Object Manipulation in Clutter, Robotics and Autonomous Systems 42 (3-4) (2003) 271–281.
87. Smith, T., Simmons, R. (2006). Heuristic Search Value Iteration for POMDPs, Technical report, Robotics Institute, Carnegie Mellon University.
88. Joelle P. et al. (), A HierarchicalApproac h to POMDP Planning and Execution,
89. Shenghao, J. et al. (2019). An Interactive Lane Change Decision Making Model With Deep Reinforcement Learning, 2019 IEEE 7th International Conference on Control, Mechatronics and Automation, 978-1-7281-3787-2/19/ ©2019 IEEE.
90. Rajesh, M. et al. (2025). Hierarchical Object-Oriented POMDP Planning for Object Rearrangement, arXiv:2412.01348v3.
91. Xuanjin, J. et al. (2025). Hi-Drive: Hierarchical POMDP Planning for Safe Autonomous Driving in Diverse Urban Environments, arXiv:2409.18411v2.
92. Toussaint, M., Charlin, L., Poupart, P. (). Hierarchical POMDP Controller Optimization by Likeliho o d Maximization
93. Huifan, G. et al. (2024). Inducing Individual Students' Learning Strategies through Homomorphic POMDPs, arXiv:2403.10930v1.
94. Foka, A., Trahanias, P. (2005). Real-Time Hierarchical Real-Time Hierarchical POMDPs for Autonomous Robot Navigation, In Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems (IROS), 2005.
95. Theocharous, G. et al. (). Representing hierarchical POMDPs as DBNs for multi-scale robot localization,
96. Hansen, E. et al. (2003). Synthesis of Hierarchical Finite-State Controllers for POMDPs, ICAPS-03 Proceedings. Copyright © 2003, AAAI (www.aaai.org).
97. Mahadevan, S. et al. (). Hierarchical Approaches to Concurrency, Multiagency, and Partial Observability
98. Pineau, J., Thrun, J. (2002). An integrated approach to hierarcy and abstraction for POMDPs,

99. Salomons, N. et al. (2022). BKT-POMDP: Fast Action Selection for User Skill Modelling over Tasks with Multiple Skills, arXiv:2404.10930v1.
100. Daubigney, L. et al. (2013). Model-free POMDP optimization of tutoring systems with echo-state networks, Proceedings of the SIGDIAL 2013 Conference, pages 102–106, Metz, France, 22-24 August 2013.
101. Ramachandran, A. et al. (2019). Personalized Robot Tutoring using the Assistive Tutor POMDP (AT-POMDP), Copyright c 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org).
102. Folsom-Kovarik, J. T. et al. (2010). Scalable POMDPs for Dianosis and Planning in Intelligent Tutoring Systems, AAAI Fall Symposium (FS-10-07).

****