**International Journal of Science Academic Research**
Vol. 07, Issue 10, pp.1603-1625, October 2026
Available online at http://www.scienceijsar.com
ISSN: 2582-6425

**Research Article**

# ITS AUTHORING TOOLS AND PLAYERS DEVELOPMENT USING KNOWLEDGE GRAPHS THAT REPRODUCES THE CLASSROOM TEACHING STYLES OF HUMAN TEACHER

**[1]Yun-Sok Ham, [2]Hyon-Il Son, [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, [1]Ho-Gun Jong, [2]Gwang-Sik Kim and [2]Se-Ryon Kim**

[1]Institute of Information Technology, University of Science, 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

**Received** 10thAugust 2026; **Accepted** 05thSeptember 2026; **Published online** 30thOctober 2026

**Abstract**

We propose an approach to construct an intelligent tutoring system (ITSs) using a knowledge graph (KG) that reproduces the classroom lecture style of human teacher. We propose an approach to divide hierarchical partitioned knowledge graphs into subject graphs, hyper-concept graphs, basic concept graphs, sub-concepts and help-concept graphs. Hierarchical partitioned knowledge graphs can be easily constructed using visualized graphics drawing tools and embedded in the database, the product of the authoring tool. Using the intelligent tutoring system authoring tool, we embed domain model data, basic data of student model, and basic data of pedagogical model into the knowledge graphs. When using the ITS authoring tool, the results are a database and a set of files of various types. The ITS players use the knowledge graph and learning path generator to generate the learning path inherent to the student. The results of the evaluation and analysis of the performance gains of students who studied with the contents of intelligent tutoring using the knowledge graphs were very positive.

**Keywords:**Intelligent Tutoring Systems, Knowledge Graphs, Hierarchical Partitioned Knowledge Graphs, Hierarchical Partitioning of Knowledge Graphs, Learning Path Generation.

## 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. 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.

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

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 User Interface (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. The learning activities of the intelligent tutoring system considered in this paper are driven by tutoring contents using only the knowledge graphs. 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. 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 [18-22, 62-76]. In the following figure, we use the term module instead of model in the sense that we implement the model.

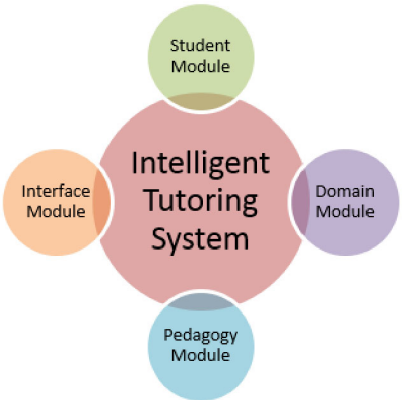



**Figure 1.The Components of ITSs [63]**

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].

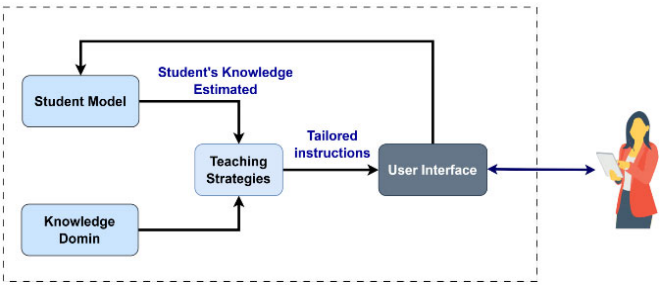



**Figure 2.The General Flow Diagram of ITSs [50]**

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). Researches uses 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, feedforward, 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].

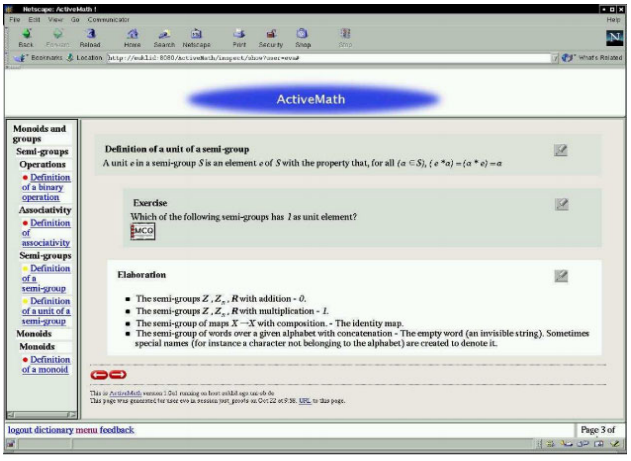



**Figure 3.A Screenshot of an ActiveMath Session [27]**

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 Stellan Ohlsson'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.

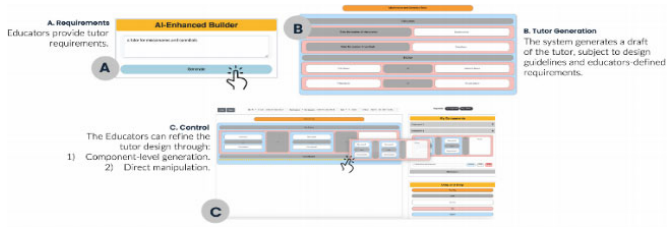



**Figure 4.Illustration of Our AI-Enhanced Tutor Builder System: Educators Provide Input Requirements (A) which Inform the Automated Generation of a Draft Tutor Interface (B), Followed by the Educator's Hands-on Refinement through Component Generation and Direct Manipulation (C), Visualizing the Integration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

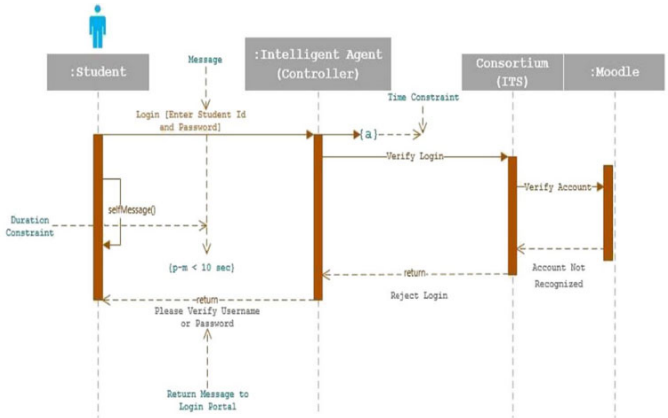



**Figure 5. The Sequence Diagram that Describes a Login Interaction among a Set of Objects Participated in for Example (Student, Intelligent Agent, Consortium, and Moodle) that has a Collaboration (or Scenario on Intelligent Tutoring), Arranged in Chronological Order; It Shows the Objects Participating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

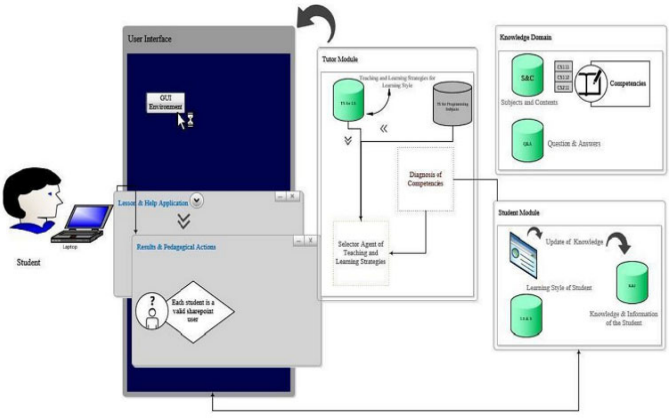



**Figure 6. The Proposed Architecture for ITS. Since the Programming Subjects have been Chosen and Customized, they will be Made Available to the Student through the Moodle Platform. The Assessment will be Applied after the Student has Revised a Subject, and the Database that Stores the Student's Output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

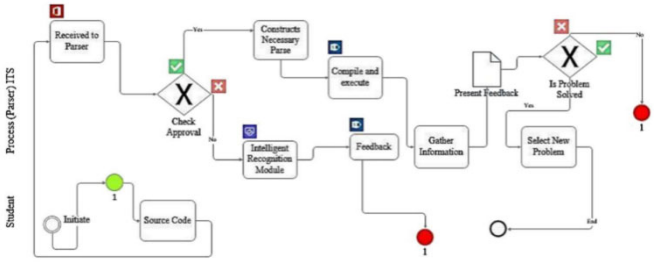



**Figure 7. The Flow Chart Process Model for the Proposed ITS. Due to the Complexities of Semantic Decoding, ITS can only Mentor a Limited Subset of Programming Languages Concurrently. Therefore, at some Point, It Necessitates Making Restrictions on a Small Subset[25]**

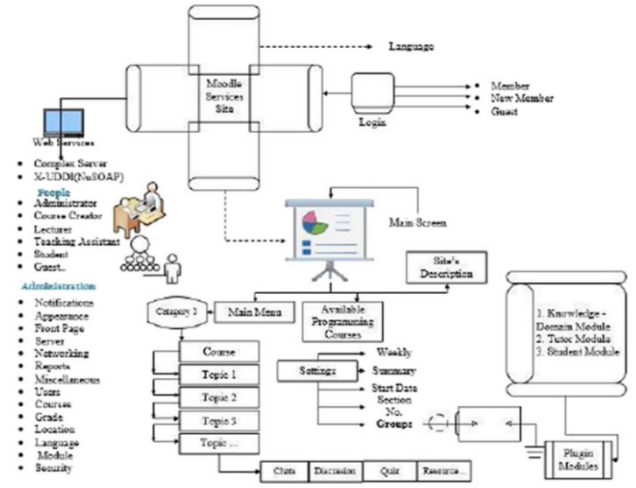



**Figure 8. The Broad-Spectrum Architecture of Moodle and its Functions and how the ITS System will be Embedded into this LMS[25]**

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.

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

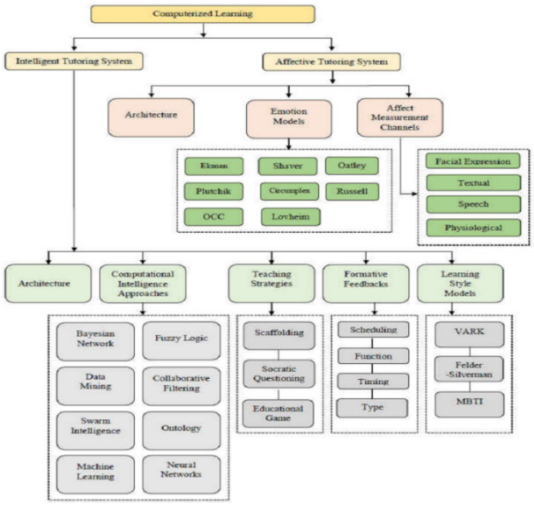



**Figure 9. State-of-the-Art Approaches for Devising ITS and ATS[27]**

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 enables 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.

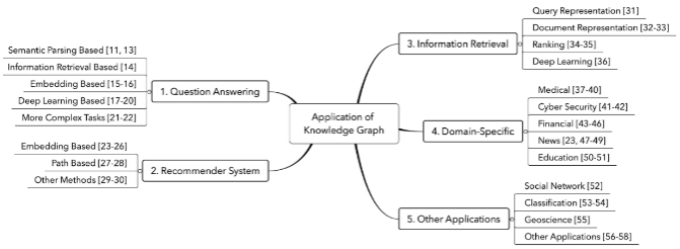



**Figure10. Application Fields of KGs [1]**

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 [78].

A brief summary of the process of constructing an intelligent tutoring system using only the knowledge graph is as follows [78].

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].

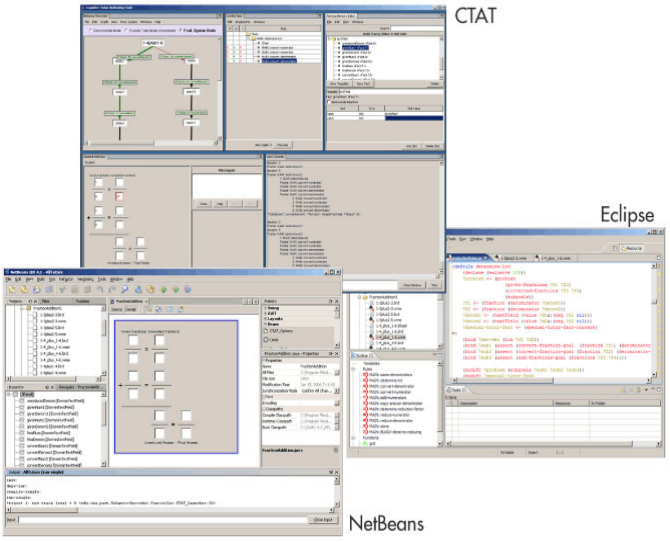



**Figure11. The Cognitive Tutor Authoring Tools[40]**

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].

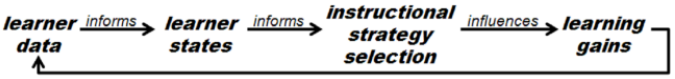



**Figure 12. The Overall Flow Diagram of GIFT Platform[63, 74]**

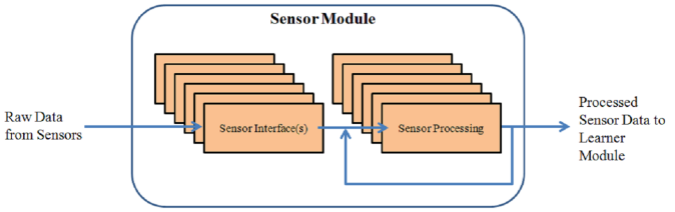



**Figure13. The Flow Diagram of Sensor Modules in GIFT Platform [64]**

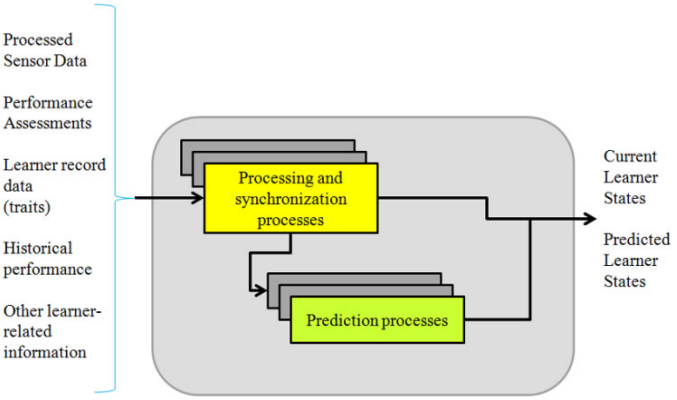



**Figure14. The Flow Diagram of Learner Modules in GIFT Platform[67]**

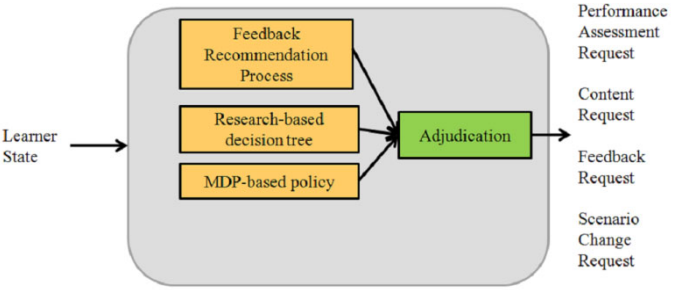



**Figure15. The Flow Diagram of Instructional Modules in GIFT Platform[66]**

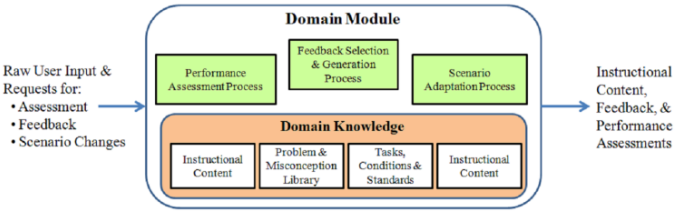



**Figure 16. The Flow Diagram of Domain Modules in GIFT Platform[65]**

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].

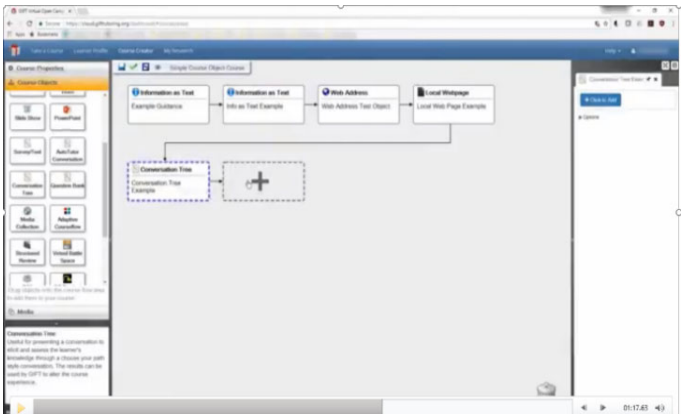


**Figure17. An Example of an Instructional Model Authoring Flow Diagram in the GIFT Framework (1)[68]**

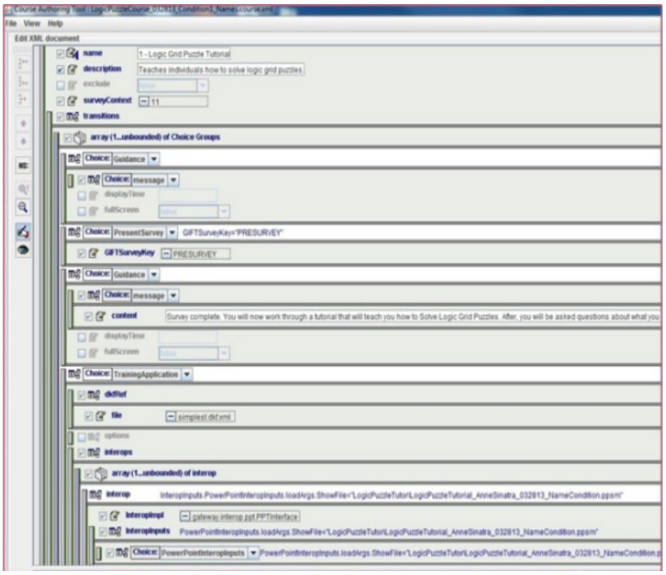


**Figure18. An Example of an Instructional Model Authoring Flow Diagram in the GIFT Framework (2)[68]**

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

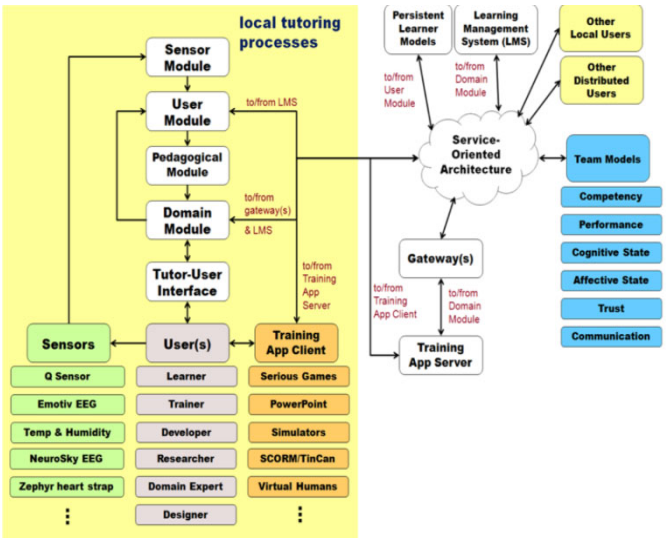



**Figure19. The Architecture and Operation Flow Diagram of GIFT Platform[65]**

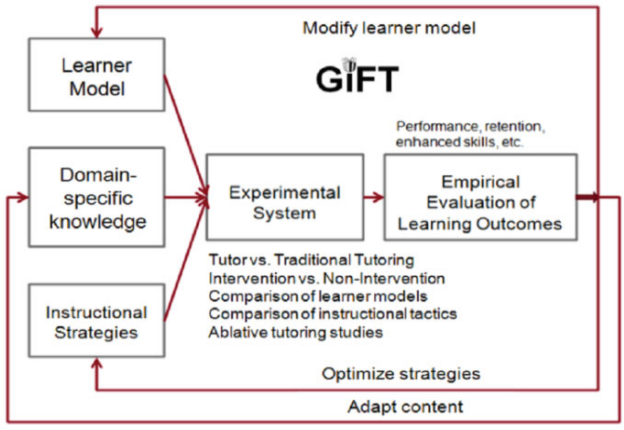



**Figure20. The Architecture and Operation Flow Diagram of GIFT Platform (Overview)[65]**

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

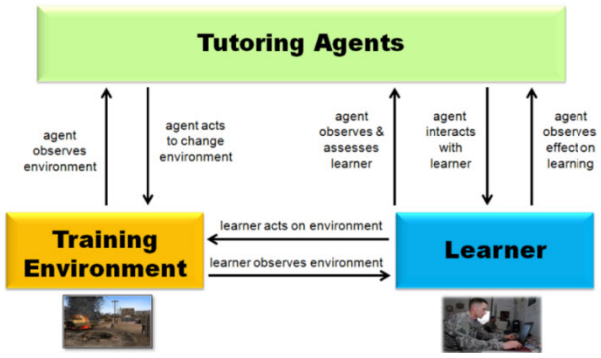



**Figure21. Interaction between Learner, Tutor and Environment (from an Agent Perspective) on the GIFT Platform[66]**

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

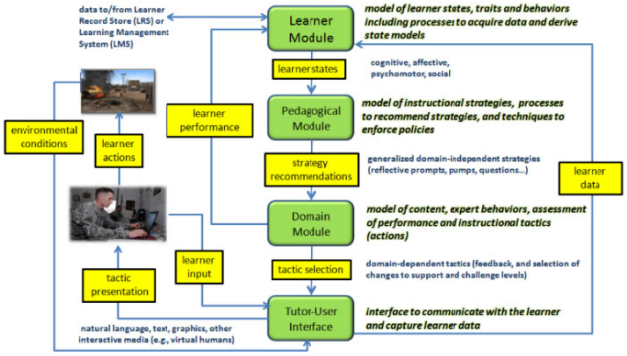



**Figure22. Interaction between System Components, Individual Learner and External environment on GIFT Platform[66]**

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

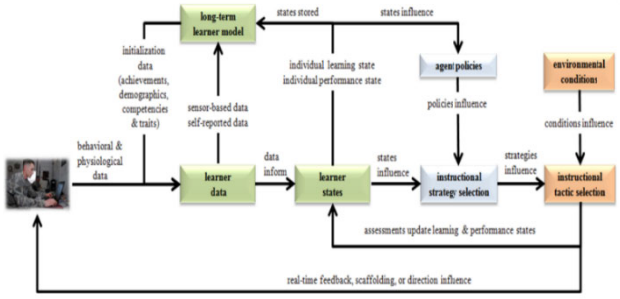



**Figure23. Learning Effect Model (LEM) for Individual Learners on the GIFT Platform[67]**

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

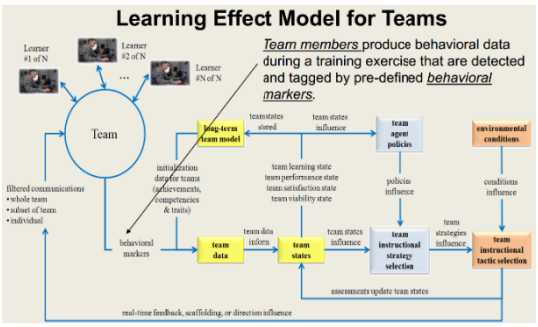
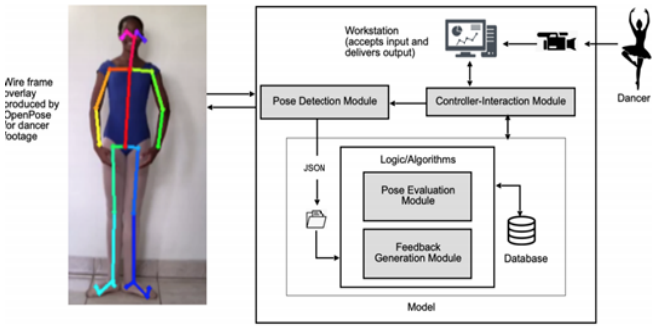



**Figure24. Learning Effect Model for Group Learning on GIFT Platform (LEM)[68]**

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.

**Figure25. The Connection between GIFT and Kinect[68]**

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

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

### 5.1. 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.

### 5.1.1. 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 26shows an example of a subject graph.

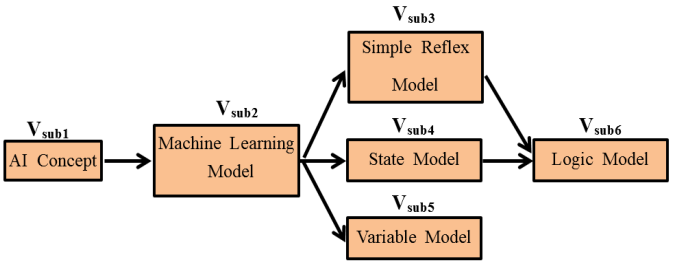



**Figure 26.Subject Graph Example**

### 5.1.2. 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 figures show the hyper-concept graph.

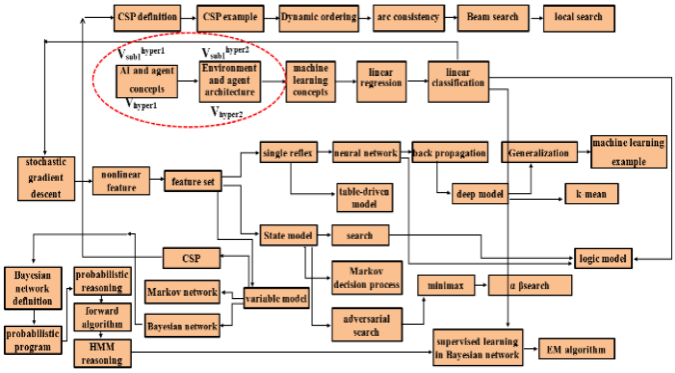



**Figure 27.Hyper-Concept Graphs Example (1)**

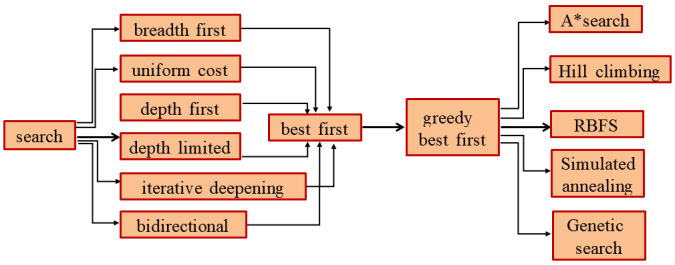



**Figure 28.Hyper-Concept Graphs Example (2)**




**Figure 29.Hyper-Concept Graphs Example (3)**

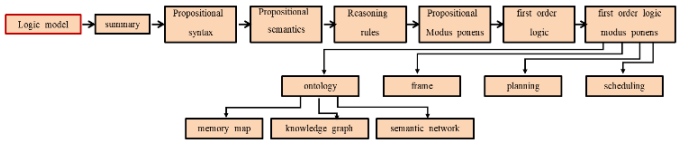



**Figure 30.Hyper-Concept Graphs Example (4)**

### 5.1.3. 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 31). Figure 30 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 27.

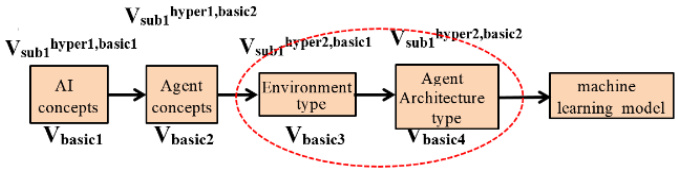



**Figure 30.Basic Concept Graphs Example**

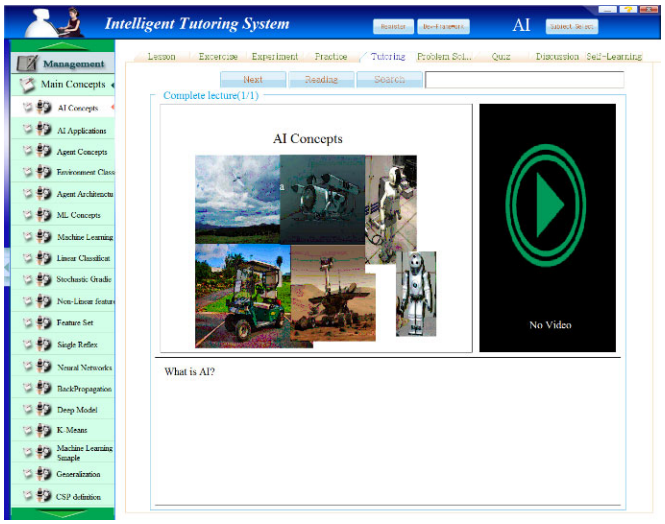



**Figure31.In Tutoring System, the Choice is a Basic Concept**

### 5.1.4. Sub-concept Graphs and Help-concept Graphs

Figure 32 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.

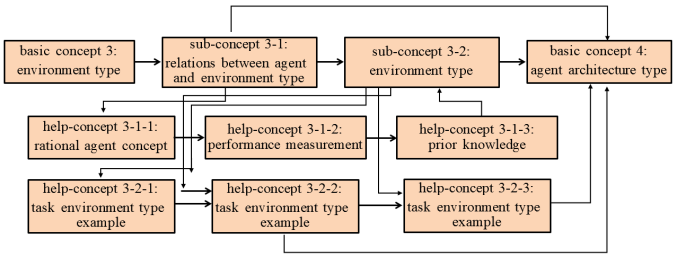



**Figure 32.Sub-Concept and Help-Concept Graphs Example**

### 5.2. 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 33). Many users have been asked to make a more convenient tool to create a hierarchical partitioning knowledge graph.

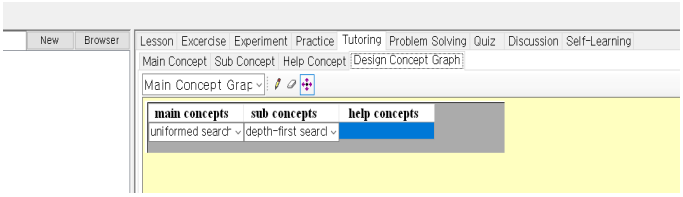



**Figure33. A Screenshot for Tools Creating Hierarchical, Partitioning Knowledge Graphs Using Document Editing Functions**

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 34 below shows a scene constructed by drawing a knowledge graph by a mouse.

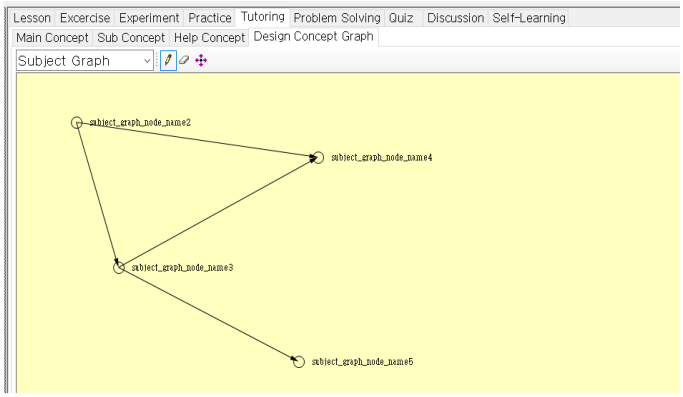



**Figure34.A Screenshot of Graph Drawing by Mouse**

## 6. AUTHORING AND CHECKING (PLAYING) TUTORING CONTENTS USING ONLY 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.

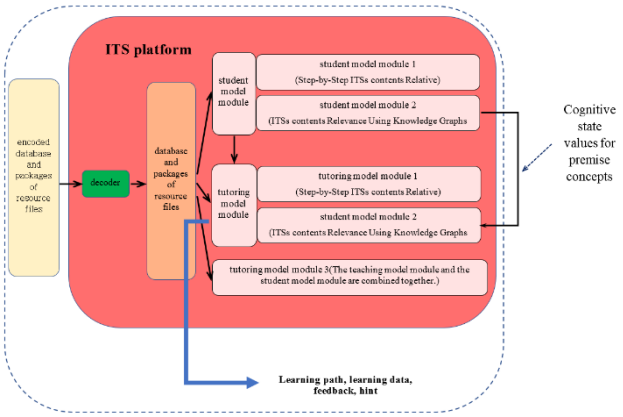



**Figure35.A Framework Diagram and Action Flow Diagram Corresponding to Tutoring Using only Knowledge Graph on the Proposed ITS Platform**

### 6.1. 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".

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.

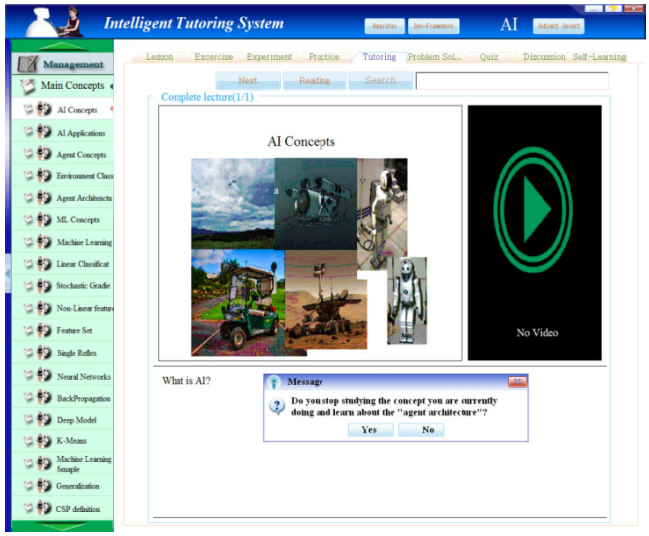



**Figure36.AScreenshot of Cases where a Student has Chosen the Basic Concept of "Active Structure Classification"**

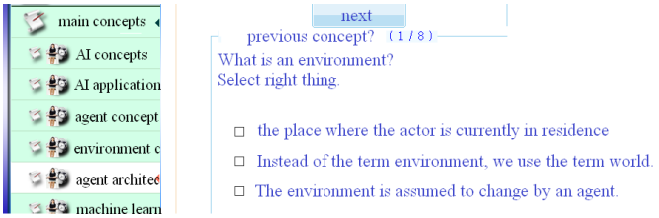



**Figure37.A Question Example to Understand the Perception of the Basic Concept of Environment Classification, a Premise Concept of the Basic Concept of Active Structure Classification**

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)
//////////////////////////

## 6.2. 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).

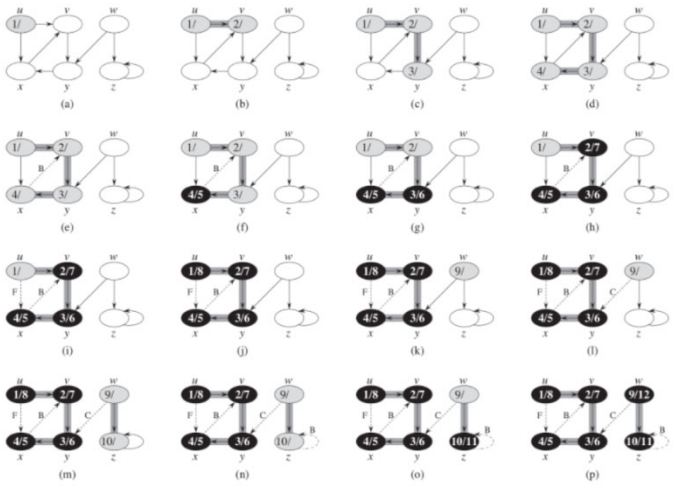


**Figure38. Depth-First Search with Back-Edge Detection in Directed Graphs (from [71])**

Figure 38 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.

[Algorithm2: 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*∈*G*. *Adj* [*u*] // Search for the edge (u, v).
5: if *v*. *color* == WHITE
6: *v*. π = *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*→*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 3: AcyclicViolation Detection Algorithm Using recStack in DFS (on the Subject Graph)]
input: subject graph$G_{sub}$ = ($V_{subi}$, $E_{subi}$), 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 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 4: 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 $G_{sub}$ = ($V_{subi}$, $E_{subi}$) ≈ $G_{hyper}$ = ($V_{hyperj}$, $E_{hyperj}$) = ($V_{subi}^{hyperl}$, $E_{subi}^{hyperl}$)
$G_{hyper}$– 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$G_{sub}$ = ($V_{subi}$, $E_{subi}$) ≈ $G_{hyper}$ = ($V_{hyperj}$, $E_{hyperj}$) = ($V_{subi}^{hyperl}$, $E_{subi}^{hyperl}$)

$\approx G_{main} = (V_{maink}, E_{maink}) = (V_{subi}^{hyperl, mainm}, E_{subi}^{hyperl, mainm})$

$G_{main}$– 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 $\in V_{main}$

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 $\in V_{sub\text{-}concept}$

Creating a help-concept Graph $G_{help\text{-}concept}$

$G_{help\text{-}concept}$ – isCyclic // Performing an acyclic violation detection for a help-concept graph

### 6.3. 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 35.

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

[Algorithm5. 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 sub-path as above and generate the learning path according to the corresponding probability value.
16: }
17: else
18: {
19: Make a choice of the *k*-2th basic concept.
20: }

b. A Description of Line 12 of Algorithm 5

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.

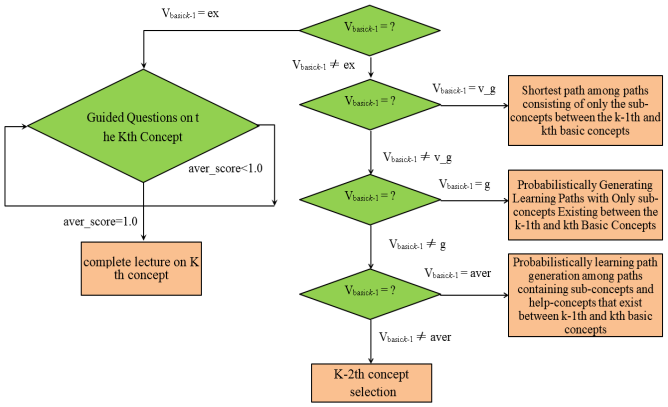



**Figure 39.Learning Path Generation Flow Diagram**

## 7. EXPERIMENT AND RESULT

We analyzed the learning effects of students when using single-layer knowledge graphs and the hierarchical partitioned knowledge graphs. For AI subjects, we constructed a single-layer knowledge graph and a hierarchical partitioned knowledge graph, respectively.

**Table 1. 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<br>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. |

The number of knowledge concepts in the single layer knowledge graph was 150, while in the hierarchical partitioned knowledge graph, the number of nodes in the subject graph was 8, the number of nodes in the hyper-concept graph was 55, the number of nodes in the basic concept graph was 230, the number of sub-concepts graph was 620, and the number of help-concepts graph was 644.When the tutoring model constructed from the single-layer knowledge graph and the tutoring model constructed from the hierarchical partitioned knowledge graph were used for each group, the average score of each group is as follows.

**Table2. Assessment of student achievement using two models**

| | Mean value |
|---|---|
| Single layer model | 7.2 |
| Hierarchical partitioned model | 7.8 |

Evaluation of student's performance gains achievement using both models Mean-value single layer model 7.2 Hierarchical split model 7.8

As a result, we found that the student achievement using the instructional model constructed from the hierarchical split-type knowledge graph was higher than that of students using the single-layer model.

## 8. CONCLUSION

In this paper, we propose an intelligent tutoring system construction approach using hierarchical partitioned knowledge graphs.The hierarchical division of a subject or topic by experts in the field allowed us to construct hierarchical partitioned knowledge graphs for their subject. The hierarchical partitioned knowledge graphsare constructed as subject graphs, hyper-concept graphs, basic concept graphs, sub-concept graphs, and help-concept graphs.This hierarchical partitioned knowledge graphs include domain model data, basic data of student model, and basic data of pedagogical model.The ITS players use the knowledge graph and learning path generator to generate the learning path inherent to the student. The results of the evaluation and analysis of the performance gains of students who studied with the contents of intelligent tutoring using the knowledge graphs were very positive.

**Abbreviations**

| | |
|---|---|
| ITSs | Intelligent Tutoring Systems |
| LMSs | Learning Management Systems |
| KG | Knowledge Graph |
| UI | User Interface |
| HPKGs | Hierarchical Partitioning Knowledge Graphs |
| HP | Hierarchical Partitioning |
| DAGs | Directed Acyclic Graphs |

**Author Contributions**

**Yun-Sok Ham:** Conceptualization, Methodology
**Hyon-Il Son:** 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
**Ho-Gun Jong:** Software
**Gwang-Sik Kim:** Software
**Se-Ryon Kim:** 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 Artif Intell Educ (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 Artif Intell Educ (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 Artif Intell Educ (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 Artif Intell Educ (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. [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 Artif Intell Educ (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. 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.
78. 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.


*********