JKAU: Comp. IT. Sci., **Vol. 11** No. 2, pp: 1 – 24 (1444 A.H. / 2022 A.D.)
DOI: 10.4197/Comp.11-2.1

Agent-based Intelligent Tutoring System for Arabic Grammar

Sumayah Natheer[1], Hanan Elazhary[1,2] , Haneen Al-Ahmadi[1]
[1]Dept.Computer Science & Engineering
University of Jeddah, Saudi Arabia
[2] Electronics Research Institute, Cairo, Egypt
sahmed0023.stu@uj.edu.sa, helazhary@uj.edu.sa, hhalahamade@uj.edu.sa

Abstract. **an Intelligent Tutoring System (ITS) gives tailored teaching to students based on their learning preferences and/or skills to address the problems of learning. These include the difficulty of providing quality learning to students in remote areas, which may be hard to reach by students and/or teachers. Additionally, tracking the progress of each individual student and providing suitable learning material and exercises is not an easy task. One of the best candidates for such systems is Arabic grammar, which is a quite complex subject. Nevertheless, due to this complexity it hasn't been adequately handled by researchers. Also, those systems have generally been developed for elementary-level grammar courses. Thus, this paper aims to develop an intelligent tutoring system to address problems of learning Arabic grammar and make education reachable anywhere based on self-paced learning. The system is developed for an Arabic grammar course at the University of Jeddah. An intelligent agent is a pack of software tools to be merged with various applications and databases to facilitate their operation by making them modular. They also facilitate modification and extension of any ITS without having to rebuild it from scratch. Though agents have been integrated in Arabic grammar tutoring systems, to the best of our knowledge, no framework or methodology has been proposed for developing such agent-based grammar ITSs. We attempt to address this problem in this paper.**
Keywords—Agents, Arabic Grammar, Intelligent Tutoring System

I. INTRODUCTION

In traditional education, the students and teachers must attend school, but many remote areas are very difficult to reach by qualified teachers, and the distance to school and high transportation costs may lead to illiteracy. In addition, the students have different skill levels, and there is difficulty in tracing student errors individually by the teacher [1]. Modifying traditional tutoring to meet student needs is costly and resource wasting. Intelligent Tutoring Systems (ITSs) provide a solution to those problems. An ITS provides content, exercises, and feedback to learners. The primary advantage of ITSs is their ability to interact with learners one-to-one and personalize the instruction and exercises according to their backgrounds and progress.

Ideally, ITSs should contain the domain's concepts, rules, and problem-solving strategies. In addition to providing this material to learners, they should be able to analyze the students' answers to exercises to detect any errors using techniques such as model tracing and constrained-based modeling (CBM). By assessing the students' knowledge, they should generate feedback and learning material and exercises relevant to individual learners. In other words, ITSs should have adaptive capacity to modify feedback and suggested exercises to guide learners to comprehend the presented materials. Besides, ITSs should deliver data and analyses to teachers and developers who are aspiring to enhance teaching practices and methods.

ITSs have been developed for various domains including mathematics, languages, computers, and physics. The Arabic language is a Semitic language spoken by hundreds of millions worldwide and in the Middle East. The main

difficulty in learning the Arabic language is its grammar, which is generally harder than that of many other languages such as English. This makes Arabic grammar a suitable candidate for ITSs. Nevertheless, this complexity led to a shortage in such ITSs. Thus, one of the goals of this paper is to address this problem. Unlike related ITSs that are generally developed for elementary school Arabic grammar, the proposed ITS handles a course at the University of Jeddah.

Intelligent agents provide packs of software tools that can be merged with applications to facilitate their operation and modification by making them modular. Though agents have been integrated with Arabic grammar tutoring systems, to the best of our knowledge, no framework or methodology has been provided to guide development. We attempt to address this problem by exploiting intelligent agents in the functionality of the proposed ITS, while providing a framework for developing such ITSs using agents. To sum up, our main objective is to develop an agent-based ITS for university-level Arabic grammar and a systematic methodology.

Towards achieving this goal, several sub objectives to be achieved include: (1) analyzing a university-level Arabic language grammar course, defining related linguistic skills and sub-skills, and then arranging and sorting them, (2) preparing a set of exercises and relating them with appropriate skills and sub-skills, (3) designing and developing an agent module, a skill module, and a database module using Prometheus design tool (PDT) and JADE, and finally (4) developing a web-based ITS for Arabic language grammar.

The contributions include (a) developing an ITS for a university-level Arabic grammar course that has not been handled before, (b) delivering a methodology and framework for analysis and design of agent-based ITS for Arabic grammar (and possibly other domains), and (c) implementing this ITS to help the students and teachers gain maximum advantages of Arabic grammar tutoring. To the best of our knowledge, no previous efforts have been conducted for associating PDT with Arabic grammar ITS, and its development using JADE. Although agent-based ITS for Arabic grammar is not a new issue, it is a problematic issue that calls for further research because it is still in its infancy. We provide a novel approach for integrating agents in the functionality of the ITS.

The next sections of the paper are organized in the following way: First, the paper provides detailed background and comprehensive literature review related to the topic. This is followed by specifying the research methodology and the proposed design of agent based ITS for Arabic grammar. Finally, it presents the implementation of the proposed ITS, discussion, conclusion, and future work.

II. BACKGROUND

This section presents background about both ITSs and intelligent agents.

A. *Intelligent Tutoring Systems*

The fundamental goal of original intelligent computer-aided instruction or later ITSs is to develop and advance models of tutoring and instruction aiming to grant students state-of-the-art academic tutoring, using a suitable human teaching strategy, on a one-to-one basis. The early ITSs were proposed in the 1970s [2] aiming at granting academic engagements that are customized and tailored to weaknesses, strengths, and the learning style of a given student. Although ITSs can use AI in their operation, the significant difference that makes ITSs unique is the prediction of the state of the learner knowledge and acting accordingly. In the following subsections, we discuss typical ITS architecture and student modeling techniques.

- *Typical ITS Architecture*

Typically, an ITS is formed using three modules: domain knowledge, instructing knowledge, and a model of learner's knowledge state. Dede [3] refers to those modules as knowledge base, pedagogical module, and student model respectively. Recently, a fourth module, which is the user interface, has been proposed due to the

growing interest in allowing communications and conversations among learners and gadgets [4].

Knowledge Base or Expert Module: This base articulates the domain knowledge in terms of declarative knowledge reflecting 'knowing what' and procedural knowledge reflecting 'knowing how'. Several schemes are utilized to represent domain knowledge. These include lists, trees, semantic networks, frames, production rules, logic, and variations or mixtures of them. The most frequently used method for this purpose is the production system that comprises: (1) a production set of if-then rules or procedural memory, (2) a collection of facts or declarative memory, (3) a working memory, and finally, (4) an interpreter that applies the production rules to solve a given user problem. Production systems are generally utilized in expert systems, which in turn, were utilized to implement the knowledge bases of earlier versions of ITSs. For example, GUIDON executed the MYCIN expert system as its knowledge base [5, 6].

Pedagogical Module: Teaching is regarded as a knowledge-based skill that is conducted using specified techniques and strategies selected and actively merged together in response to the learner's actions. Therefore, the pedagogical module should incorporate several skills, including the presentation method, selection of exercises, the balance among teacher and student control, and should encompass feedback whenever required or applicable. A significant role of a pedagogical module is the way it responds to student errors. The teaching techniques and strategies are supposed to be validated successfully so that the pedagogical module becomes capable of successfully responding to the learner's errors.

Student Model: This model provides the student's current state of knowledge by representing facts, concepts, and problem-solving skills that the student has acquired either fully or partially. The student model is generally developed and updated through answers to exercises provided to the student by the system. This requires implementing a mechanism to inspect bugs, misconceptions, inaccuracies, and erroneous information acquired by the student. This data should be collected and analyzed to deliver optimum teaching interventions for intelligent and effective student tutoring.

User Interface: is an important aspect of any successful ITS. Although the other three parts are idealistic, a successful interface makes an ITS usable and useful. Despite the difficulties, an interface should be built in a natural language for better communication while utilizing advanced AI techniques, by implementing controlled language, multiple-choice selection, and graphic interfaces. Finally, the user interface should consider the memory limitations of learners and the capabilities of man information processing [7].

- *ITS Student Modeling Techniques*

This subsection discusses some of techniques employed by ITSs for student modeling.

Overlay Model [8], [9]: This model was developed in 1976. Since then, it has been one of the most widely adopted student models, and numerous ITSs have employed it. It is based on the idea that student knowledge is a subset of domain knowledge. To decide whether or not the student has a knowledge gap, a comparison is conducted between their conduct and behaviour and that of the domain. The objective is to minimize the distance between them as a consequence.

Towards this goal, a straightforward overlay model comprises a set of domain knowledge items and employs a Boolean value to indicate whether or not the student is familiar with each of those items. The contemporary overlay approach estimates the degree of the student knowledge quantitatively using, for example, a Likert scale (good, average, or poor) or probability. The benefit of employing this strategy is that it enables increasing student understanding up to an appropriate level. The drawback of employing it, on the other hand, is that a student might approach a topic in a different way. It ignores the students' incorrect

knowledge, as well as their requirements and learning preferences. This is the justification behind the fact that the majority of ITSs aiming at individualized tutoring also incorporate, in addition to this model, stereotypes, fuzzy techniques, and others as supplementary techniques for student modeling.

Stereotypes Model [10], [11], [12]: This is another approach that is widely used for student modeling. It was first proposed and employed in a system called GRUNDY. This model has been described as follows: "A stereotype represents a collection of attributes that often co-occur in people. They enable the system to make a large number of plausible inferences on the basis of a substantially smaller number of observations. These inferences must, however, be treated as defaults, which can be overridden by specific observation."

This approach is based on a basic assumption regarding the possibility of grouping students according to shared features. These groups are what we refer to as stereotypes. The features of a new student will be compared to those of the different stereotypes and in case a match is found, the student will be assigned to the corresponding stereotype group. This approach is quite similar to classification problems in machine learning.

ITSs generally select, generate, and sequence learning material for the students, taking into consideration their current knowledge and errors. Nevertheless, most ITSs also allow students to decide what they learn and what path to follow in the courseware. This implies that the student can select too hard or too easy parts of the courseware as needed. They may also skip some parts of it. This approach has two main advantages. The first is that it can provide students with suitable individualized learning material and experience, while the second is that it overcomes the cold-start problem of a student model (how to start a new student model), since each such student will be automatically assigned to the matching stereotype group, inheriting its model.

Constraint Based Model [13], [14]: This model was first proposed to analyze the current states of students, based on their solutions, and represent their short-term knowledge. To achieve this goal, CBM represent both domain knowledge and student knowledge using a set of constraints. In order to assess the knowledge level of a student, the student is provided an exercise and the solution is diagnosed first by considering the constraints one by one and matching their relevance conditions to the solution (to figure out which constraints apply). This is followed by matching the satisfaction conditions of the relevant constraints (to figure out whether they are satisfied). Using this approach, the ITS examines each individual step taken by the student, and in case any errors are detected, it provides feedback to the student using error messages. This feedback ideally informs the student that there is an error, specifies the location of this error in the answer, and explains the violated constraints to help in tutoring.

The prominent advantages of CBM include the fact that it eliminates the need for an expert module, leading to easier and faster design and implementation of an ITS. Additionally, it provides a systematic technique for identifying students' bugs and the origins of their errors. In other words, it eliminates the need of complex analysis and reasoning to identify errors and their origin. This explains the reason that the CBM approach is widely used in ITSs in a variety of domains.

B. Intelligent Agents

Agents are generally described as self-managed software programs more like automatic machines. The interactions among agents can either be supportive or selfish. In other words, an agent can follow its own benefit or could share a collective goal with its counterparts [15]. In this subsection, we discuss agent characteristics, architectures, and how they can be exploited.

- *Agent Characteristics*

Agents are generally characterized by four attributes [16] discussed in this subsection.

Autonomy: Agents can activate themselves without explicit involvement of individuals or systems, and they have the ability to control their states and the actions they take.

Social Ability: Agents use Agent Communication Language (ACL) to communicate with their counterpart agents and possibly humans.

Reactivity: Agents are aware of their environments and have the ability to respond on time to any changes occurring within them.

Pro-activeness: Proactive agents perform actions based on prediction by following the belief–desire–intention software model. In this model, belief reflects how the agent models its environment, desire represents what the agent aims to do or simply its goal, while the intention reflects the action selected by the agent accordingly.

- *Agent Architectures*

Agents have different possible architectures [17]. This section discusses four major types.

Simple Reflex Agents: The term 'percept' refers to what an agent perceives. All what a given agent perceives until present time is called the percept history of the agent. Simple reflex agent forgets this history and considers only the current percept when taking an action. Meanwhile, it relies on a set of condition-action pairs to decide what the next action should be. Whenever the condition of a given rule is satisfied, the agent takes the corresponding action, thus changing its state. Success of this approach implies that the environment of the agent is fully observable. Accordingly, if the case of a partially observable environment, infinite loops are inevitable (since the agent has no matching rule to change its state). To tackle this problem, the agent may randomize its actions. Problems of such agents include:

- inadequate intelligence
- deals only with current percepts
- large storage capacity needed for the fully observable environment
- the need to update the set of rules with any change in the environment

Model-based Agents: A model-based agent relies on a model of the world to be able to take actions in a partially-observable environment. In other words, it addresses the main issue of the simple reflex agent. With any percept, and based on the whole percept history, it takes an action to adjust its current state. This state is stored inside the agent using a structure describing the unseen portion of the world. In order to update its state, the agent needs information about:

- the world model and how the world evolves independent of the agent
- how the actions taken by the agent affect the world model

Goal-based Agents: Those agents focus on their goals, and their decisions are formulated according to how much away they are from the goals (description of desirable situations). These agents are more resilient since they can explore various options and select the most suitable. They are proactive rather than simply reactive in their decision-making.

Utility-based Agents: These agents are developed with the end uses in mind. They are used to select the best alternative among a set of possibilities based on users' utilities or preferences. For example, in case of numerous possible paths to a destination, the preference can be the fastest, the safest, or the cheapest. This is expressed in terms of agent happiness, which is described as a utility. The agent takes the action that maximizes the utility. Utility functions map any state to a real number as a measure of the degree of happiness.

Learning Agent: This agent can learn by experience. It starts with its basic knowledge and takes actions accordingly. Nevertheless, through learning it adapts automatically. To achieve this goal, such agents generally have four components:

- Learning component: responsible for learning by experience from the environment
- Critic component: uses performance metrics to provide feedback to the learning component regarding how well the agent is performing
- Performance component: selects the suitable action based on the agent's performance
- Problem generator: suggests actions for new and informative experiences

- *How to Tell an Agent What to Do*

Agents accomplish tasks for us. In order to get the tasks done, we need, in one way or another, to identify a task that is to be accomplished and communicate it to the agent. The questions now are how to identify the tasks and how to express to the agent what is required. The method that comes directly to our minds is to write a program for the agent to execute. The clear benefit of this approach is that it eliminates uncertainty regarding what we want the agent to accomplish, since it will take necessary actions to follow our instructions and only our instructions. Nevertheless, the problem is that we have to plan carefully how the task will be performed by the agent and have to take into consideration all possible circumstances, or else, if unexpected ones occur, the agent will not execute as required and will be unable to respond accordingly. Therefore, typically, we tell the agent what to do without telling it how to do it. This can be achieved by defining the tasks indirectly and utilizing some type of performance measure.

III. Literature Review

This section discusses related work including agent-based ITSs, grammar ITSs, and agent-based grammar ITS (a combination of the former types).

A. *Agent-Based Intelligent Tutoring Systems*

In this section, we discuss some agent-based ITs describing how they employ agents in their architectures.

Remote Intelligent Tutoring System (RITS) [18]: This system has been developed as an environment through which students can learn one or more subjects, by acquiring not only tutoring material, but also assessments to solve, accompanied by timely feedback as needed. Agents in RITS are divided into three layers. The first layer is the user interface agent responsible for communicating with the students. The second layer is responsible for generating tutoring materials using key-points learning agent, test and assessment agent, and a set of automatic problem-solving agents. Finally, the third layer is the control layer comprised of two agents, one responsible for control of learning and assessment and the other responsible for control of automatic problem solving.

Intelligent Tutoring System with Emotional Pedagogical Agents [19]: This ITS is formed of a set of modules in addition to emotional pedagogical agents for better interaction between students and the ITS. The modules are the user interface, the student model, the domain knowledge, and pedagogical knowledge maps that map a set of knowledge concepts to a set of knowledge descriptions. The student model reflects student's characteristics such as knowledge level, cognitive abilities, learning style, and psychology. The emotional pedagogical agent recognizes student's expressions, analyzes corresponding emotions, and communicates with both the student model and knowledge map to generate relevant personalized tutoring material to the student.

FlexiTrainer [20]: This is an ITS authoring framework. It provides a set of tools to aid in specifying domain knowledge (principles, skills, and tasks), exercises, student model based on Bayesian inference, and tutor behaviour (teaching and assessment strategies under different conditions). This latter editor specifies visually the behaviour of tutoring agents to carry out the educational goals. FlexiTrainer has been successfully used to develop an ITS for tutoring flying skills to helicopter pilots.

MathTutor [21]: This is another multi-agent ITS authoring tool. It utilizes a set of formalisms to

facilitate the task of specifying domain knowledge, student model, and the pedagogical model implemented as expert system rules. In addition to the teacher authoring interface, it provides a student user interface and a set of tutoring agents. Each agent is controlled by a coordinator module. Based on the expert system rules and the student model the suitable learning material is generated from the domain knowledge. The result of an interaction may result in an update to the student model. Feedback is also provided to the student as needed. Control may transfer from one agent to another in case the student needs to move to an advanced level or another curriculum for example. The tool utilizes Petri net for the interaction between the various agents and the students.

A Multi-agent ITS for Learning Computer Programming [22]: This is an ITS for computer programming based on multiple agents. It relies on teaching the students several programming languages based on the assumption that teaching supporting domains in addition to a target domain reinforces learning especially in knowledge-rich domains like computer programming. It utilizes two types of agents in each domain bank: searching agent and conveyor agent. The former is responsible, as the name implies, for searching the corresponding bank. The latter, on the other hand, is responsible for communication among agents.

The above systems utilized different types of agents and various architectures for developing agent-based ITSs. Nevertheless, none of them was intended for grammar tutoring. In this research study, we aim to develop an agent-based ITS suitable for this domain in particular. Similar to some of the above systems, we also aim at developing a systematic method for helping researchers design and implement similar ITSs. Additionally, we will take into consideration the functionalities of the above systems and agents such as domain knowledge, pedagogical module, student model, user interface, learning and assessment, databases, and feedback.

B. Grammar Tutoring Systems

Grammar tutoring systems or Language Tutoring Systems (LTSs) have been established for several languages. They differ in their capabilities, strategies, student models, and linguistic skills. This section presents some of these systems.

Web-based Intelligent Language Tutoring System for German Grammar [23]: This web-based ITS is developed for a grammar course in German. Its intelligence is exhibited in its capability to parse students' inputs. In response, students may be given relevant feedback and possibly suggested exercises. To evaluate the ITS, 19 college students sat for a one-hour test. In this test, 84% of those students thought that the ITS was useful and reliable due to supplying them with instant remarks and free grammar exercises.

English Tutor Using Data Mining Techniques and Jackson's Learning Styles [24]: This is an online English grammar ITS, which takes into consideration learning styles when presenting learning material. It has a database layer storing the student model, which is updated according to the students' behavior. The second application layer contains learning material to be presented to the students, videos, and exercises. Finally, the last client layer is the user interface targeting the students and the system administrator.

According to Jackson's learning styles, there are four basic phases in learning: speaking, grammar, writing, and reading. These phases must be studied, and their quizzes taken by each student in any order they prefer. This implies 24 different possible learning patterns. Data mining techniques are exploited to cluster students' learning styles according to this model. Experiments were conducted to figure out, among the 24 patters, the most common and the most successful. The experiments showed that in both cases, the pattern was speaking - reading - grammar - writing, with scores of 87.4% and higher.

AG Tutor [25]: This is an ITS for teaching Arabic grammar to students at the fourth grade of elementary schools in Egypt. Domain knowledge covers several corresponding lessons. Linguistic

skills are identified under each lesson with corresponding questions. Additionally, CBM is utilized to define a set of constraints representing grammar rules under each skill. The advantage is that this eliminates the need for a bugs' library. To model the student, after answering each question, a list of satisfied and violated constraints is updated. Suitable questions are generated accordingly. AG Tutor has a graphical user interface with multimedia to ease the interaction with the students and make it interesting to them.

Arabic Intelligent Call (ICALL) [26]: This ITS is intended to teach the Arabic language to Egyptian school students in their first year in addition to foreigners learning Arabic as a second language. ICALL has been developed through a number of versions for this purpose. It provides exercises to the students and allows them to write answers as solutions to them. Based on stored model answers and morphological, syntax and error analyzers, it is able to detect and specify any errors in the answers and provide relevant feedback in response. In the case of failure of such approach, a set of buggy rules is exploited for parsing ill-formed input sentences. This is in addition to a number of constraints to locate and diagnose incorrect verbs.

The above ITSs have different architectures and target grammar tutoring in various languages including Arabic emphasizing the need for a systematic approach to develop grammar ITSs. Meanwhile, it is clear that there is a shortage of ITSs for Arabic grammar. The proposed research aims to address both issues by developing agent based Arabic grammar tutoring system and providing a framework for developing similar ITSs.

C. Agent-Based ITS For Grammar

This section discusses an ITS which is most similar to ours since it employs agents in its operation and tutors students Arabic grammar.

AG Tutor [27]: This is an enhanced version of the original AG Tutor discussed above [25], which is intended to teach Arabic grammar to fourth grade students at elementary schools in Egypt. It has been modified by integrating it with a number of agents. The learning material is organized in a prerequisite structure according to lessons and sub-lessons or topics and sub-topics. Questions are added by the system administrator such that each question belongs to exactly one sub-topic. This system is first structured into four modules: pedagogic module (providing learning material), question selector module, student module (using CBM to diagnose student answer and specify satisfied or violated constraints), and interface module. A set of five agents are then used to connect those modules together. These are teaching assistance agent (including expert module and pedagogy), constraints and hints agent (working with the student module to generate suitable hints based on satisfied/violated constraints), learning strategy environment agent (to adapt learning strategy), secretary agent (updating student model and specifying next action based on it), and finally, an interface agent for interaction with users.

The problem with this research is that it does not provide any details of how the system has been designed and how the functionality has been distributed among the various modules and agents. Additionally, no examples have been provided demonstrating the use of CBM. The same is true about adapting the learning strategy of the students. The fact that the learning material is organized according to lessons and sub-lessons and that questions are added by the system administrator such that each question belongs to exactly one sub-topic implies that the system is inflexible and cannot be easily modified. Finally, the system is intended for elementary school students.

The proposed ITS tries to address those issues by providing a framework for designing agent-based grammar ITSs and distributing the system's functionality among intelligent agents. Accordingly, the system is structured mainly using agents realizing its modules, rather than modules connected by agents as in agent-based AG tutor. The learning material is organized into skills and sub-skills rather than topics and sub-topics to be more flexible. To increase flexibility

even further, it allows teachers and/or administrators to specify skills and subskills and add questions through the user interface. Finally, the system addresses a university-level Arabic grammar course. The use of CBM to diagnose students' errors and provide relevant feedback is postponed to future versions. In the current version, the learning material organization into skills and sub-skills facilitates this task via corresponding error messages.

To sum up, the literature review discussed above showed a shortage in ITSs for Arabic grammar though Arabic is a Semitic native language of about hundred million around the globe. Besides, Arabic grammar is generally a complex subject that needs ITSs. This tempted us to propose an agent-based ITS for Arabic grammar. As shown in Table 1, unlike its counterparts developed for elementary school Arabic grammar, the proposed system is developed for a university-level course. Specifically, the material is a course offered by the University of Jeddah. To the best of our knowledge, this is the first ITS for this course. The learning material is organized into skills and sub-skills rather than topics and sub-topics to be more flexible and to facilitate updating the student model and providing feedback in absence of domain knowledge. It is also designed to allow teachers and/or admins to define the material organization and add questions rather than only admins like its counterparts. Finally, we provide a framework for systematic design of agent-based grammar ITSs and for distributing the system's functionality among intelligent agents. Utilizing CBM to represent domain knowledge, diagnose student answers and provide relevant feedback will be implemented in future versions. This will also help in extending the student model by defining it in terms of violated and satisfied constraints in addition to satisfied and unsatisfied skills and sub-skills (as in the current version).

IV. METHODOLOGY

The research methodology consists of four main steps as shown in Fig. 1.

The first step is formulating the proposed framework using three modules: agent module, skill module and database module. This is followed by analyzing and designing the proposed system using PDT. The third step is implementation of the system using JADE. The last step is providing the discussion and conclusion, in addition to directions of future research to complete the system.

TABLE I. COMPARISON OF THE PROPOSED SYSTEM AND THE MOST RELATED SYSTEMS

Features	**ICALL [26]**	**AG tutor [27]**	**The proposed system**
Learning material	primary schools	fourth-grade	*university course*
Agent based	×	✓	✓
Organization	by topic	by topics and sub-topics	*by skills and sub-skills*
Adding questions	by admin	by admin	*by teacher or admin*
Domain model	✓	✓	×

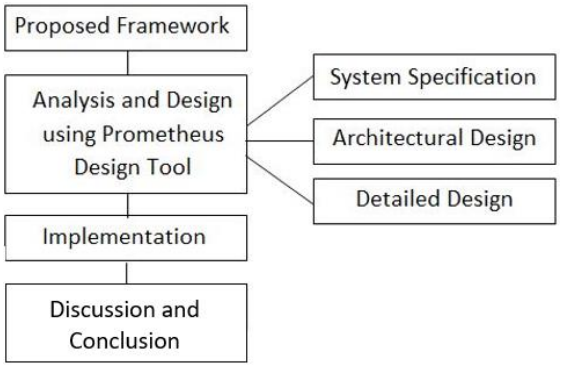

Fig.1. Research methodology steps

A. *Proposed Framework*

As shown in the block diagram in Fig. 2, the proposed ITS is composed of three modules: an agent module, a skill module, and a database module. The database module consists of Student DB and Question DB for students and questions respectively. The agent module, in turn, consists of four agents: question agent, teacher agent, student agent, and user interface agent.

The learning material of the proposed ITS is an Arabic grammar course taught at the University of Jeddah. The proposed system is the first ITS developed for this course. Its material is divided into skills and sub-skills of Arabic grammar. For example, as shown in Fig. 3, the verb form changes based on subject type in Arabic language. Accordingly, we divided the *past verb* skill into sub-skills based on subject type: masculine or feminine, and singular, dual, or plural.

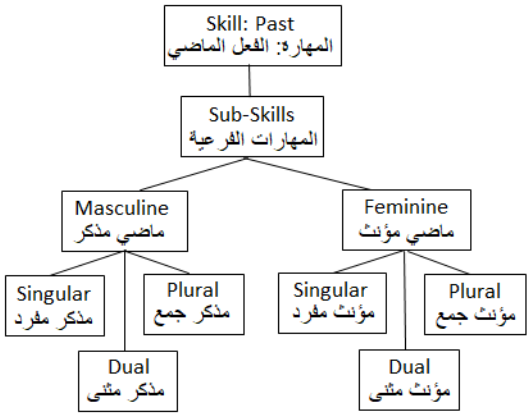

Fig. 3. Skill and sub-skills example of past verb

- *Database Module*

The database module consists of two different databases, which are Question DB and Student DB for the questions and the students respectively as shown in Fig. 4.

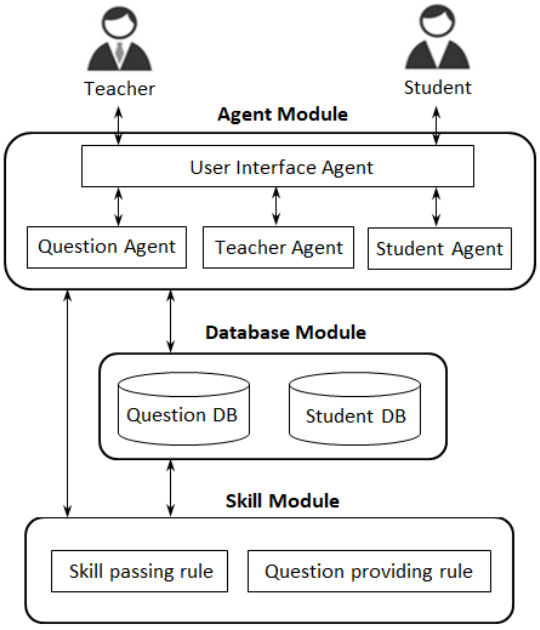

Fig. 2. Block diagram of the proposed system

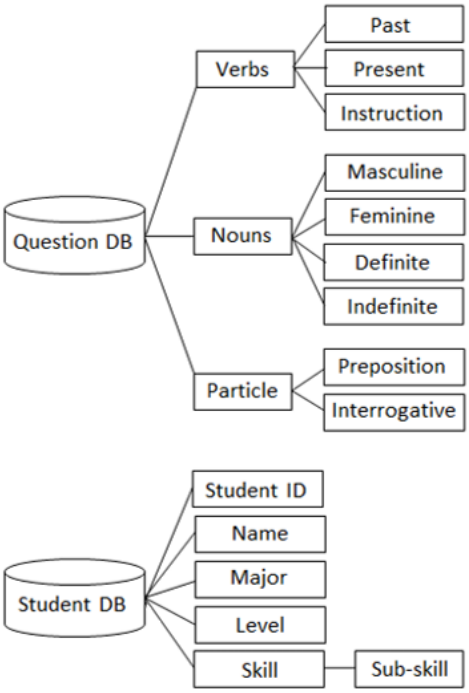

Fig. 4. Database module design

The Question DB consists of a group of questions to assess various skills and sub skills such as verbs, nouns, and particles. Under the verbs, for example, the questions can assess past tense, present tense, instruction verbs.

The Student DB, on the other hand, contains data about the students such as student ID, name, major, and level. This is in addition to passed skills and sub-skills or their current scores.

- *Agent Module*

The agent module consists of four agents: question agent, teacher agent, student agent, and user interface agent. The benefits gained from utilizing agents in our ITS can be summarized as follows:

- Using agents is time-saving since they shorten the required time for building a new ITS.

- Agent design is flexible; changeable without changing the whole system coding.
- As previously noted, an agent in Agent-Oriented Programming (AOP) has a degree of autonomy. Agents can know when to do a given behaviour and how and when to interact with other agents without external interaction. This is unlike Object-Oriented Programming (OOP), in which the object is completely controlled by the programmer. The programmer decides when an object is created, what the object can do, when the object can do a given behaviour and when to interact with other objects and how.

The goals and responsibilities of the agents in the agent module are provided in Table 2. As indicated in the table, through the *user interface*, the teacher, aided by the teacher agent, defines/modifies the Arabic grammar skills and sub-skills that are part of the *domain knowledge*.

TABLE II. AGENTS' GOALS AND RESPONSIBILITIES

Agent	Goals and Responsibilities
User Interface Agent	• provides an interface available for the users to interact easily with the system over the network • interacts with the other agents • records student skills and scores
Teacher Agent	• adds/modifies skills, sub-skills, and questions in the Question DB • defines rules for passing skills and providing questions • creates a student account and saves it in Student DB
Student Agent	• calculates the student score for each skill and sub-skill based on solutions to provided questions • defines the student's progress
Question Agent	• determines the next question based on the student's sub-skill • provides a question to the student based on the student's sub-skill

Questions are also added and attached to the relevant sub-skills, in addition to rules for passing a given skill or sub-skill and for providing questions accordingly. Questions are stored in the Question DB, while the rules are stored in the skill module. This information is part of the *pedagogical module* of the ITS. The teacher also adds the students to the ITS by creating an account for each student. This information is stored in the Student DB. This database also stores for each student information regarding passed skills and sub-skills or corresponding scores, following the overlay *student model*.

Through the user interface, the student logs into the system to solve the questions of the first primary skill. Under this skill, the questions of the first sub-skill must be correctly solved (or a certain percentage of them as defined by the teacher in the skill module) before being moved by the system to the next sub-skill. This continues until all sub-skills of the primary skill are passed. Control then moves to the next skill. The student agent calculates the student score for each skill and sub-skill based on solutions to provided questions. This information is communicated to the skill module to determine current sub-skill. The user interface agent records skills and scores of the students.

Finally, the question agent, is responsible for providing questions to the student and determining next question under current sub-skill guided by information received from the skill module.

- *Skill Module*

The skill module simulates the instructor in tutoring. To achieve its goals, it interacts with both the student agent and the question agent. It takes students' score as input from student agent to determine the current students' sub-skill by applying two rules. It then sends the current student sub-skill to the question agent to provide an appropriate question. As shown in Fig. 5, the rules it uses are as follows:

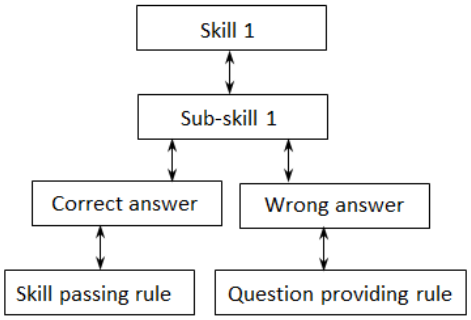

Fig. 5. Rules of skill module

Skill passing rule: This rule will apply if students' answer to a given question is correct**.**

The percentage of passing each sub-skill is determined by the teacher as previously noted. The student must achieve the percentage of current sub-skill to jump to next sub-skill and/or next skill. The current students' sub-skill will be sent to the question agent to determine next appropriate question.

Question providing rule: This rule will apply if the student fails to solve the question. This rule makes the question agent provide another question under the current sub-skill.

B. Analysis and Design using Prometheus Design Tool

The proposed framework follows Prometheus methodology for analysis and design of the grammar ITS. This is achieved through three phases: system specification (as an initial pre-design stage), architectural design, and then detailed design. As shown in Fig. 6, there are three types of diagrams corresponding to the three phases: the first type is dynamic diagrams (scenarios, protocols, and process), the second is overview diagrams (system goals, system, agent, and capability), and the third is detailed form diagrams (agent, capability, event, data, and plan).

To follow this methodology, we used PDT which supports intelligent agent design via graphical development that determines and specifies agent entities [28]. Table 3 shows PDT symbols and describes the meaning of each symbol and its role in designing agent-based systems.

- *System Specification*

The system definition step begins with a high-level system description, followed by the identification of system goals and scenarios, and a design analysis overview diagram. We identified the percepts that are input to each scenario, as well as the actions performed by the system, in the *analysis overview diagram* (see Fig. 7). For example, the teacher enters the questions into the system as a percept (input), and the system adds the questions to Question DB. Also, when a student logs into the system, his skill record is sent to the system as a percept and the system responds by asking the student questions depending on his skills. As a result, the analysis overview diagram describes the system's interface when it comes to percepts (inputs) and actions (outputs).

The next step is specifying the details of the scenarios that were identified in the analysis overview diagram. A scenario is a collection of structured phases, each of which might be a goal, action, perception, or sub scenario. The designer can also specify the roles at each step, and the data accessible, as well as providing brief explanation of procedures. We can start by designing the system scenarios overview and then specify the details of each scenario. The *scenarios overview diagram* of the proposed ITS consists of four main scenarios: create student account scenario, skill/question adding scenario, question providing scenario and skill determining scenario. Each of these scenarios has sub-scenarios as shown in Fig. 8.

PDT produces a goal for each scenario by default, using a name that's the same as that of the scenario, although this can be altered. This is the goal that the scenario is aiming for. The goals developed as a result of the scenarios are generated and added to the *goals overview diagram* (see Fig. 9).

The goals are organized roles that are divided into coherent groups and assigned to agents. They are designed to be compact, easy-to-define chunks of agent capability. The percepts and actions are then suitably assigned to the roles in order for them to fulfil their goals. This is done using the *system roles diagram*. For example, the 'determine student skill' role is responsible for the goals of analyzing student answers and calculating the score, as shown in Fig. 10. To fulfil these goals, the role requires student answer as input, then the role performs the action of 'Determine the skill'.

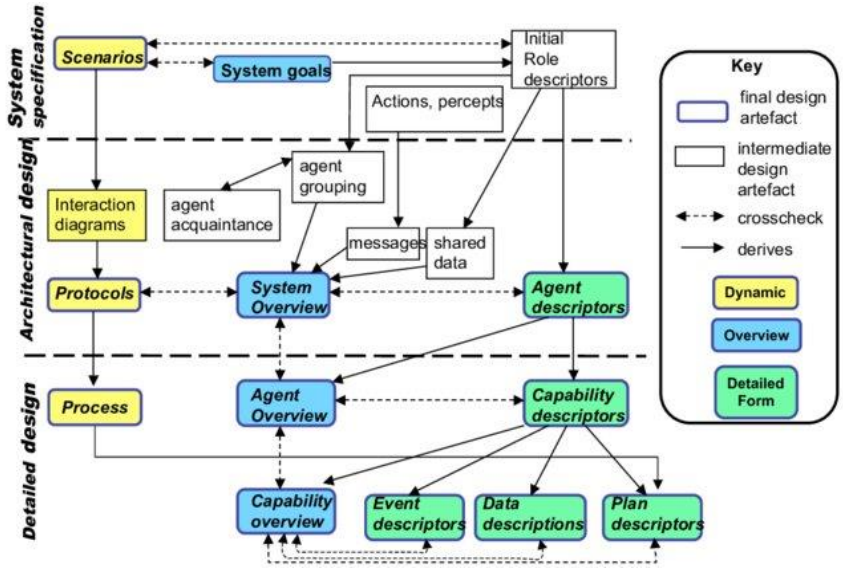

Fig. 6. Phases of Prometheus methodology [28]

TABLE III. THE PDT NOTATION SYMBOLS AND MEANINGS

Name	Symbol	Description
Agent		agent symbol
Action		what the agent does that has an effect on the environment or other agents
Role		roles or group of roles for agents
Protocol		the interaction between agents (protocols are specified using textual notations that map to AUML2)
Data		the belief (internal knowledge model) or external data (used for agent read/write)
Messages		a message for communication between agents
Percept		the input coming from the environment to the agent
Scenario		an abstract description of a sequence of steps taken in the development of a system (usually the initial step that starts with the breakdown of the "statement of problem" or description of the problem to solve)
Goal		the realizable target or achievement set for an agent

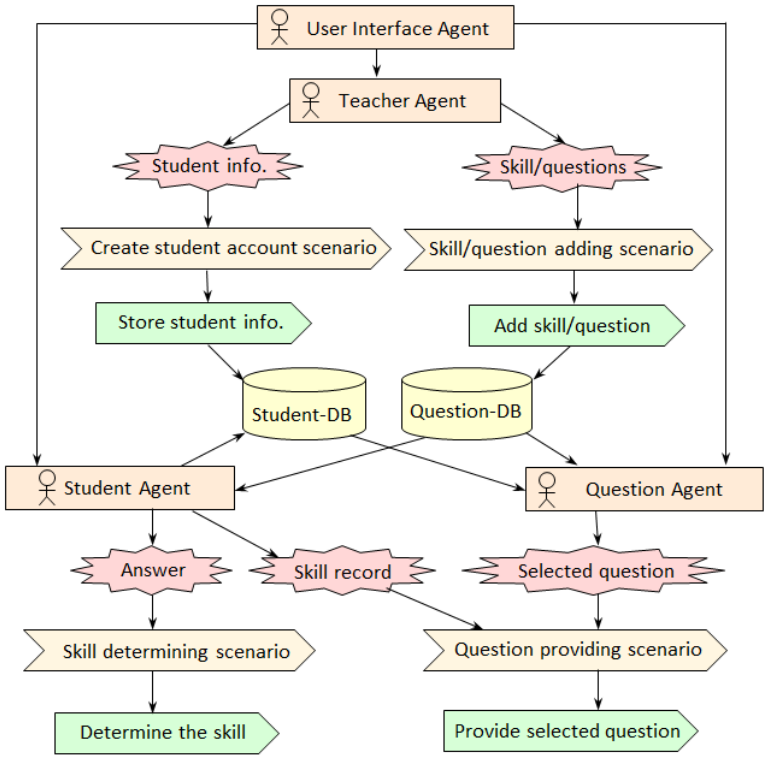

Fig. 7. Analysis overview diagram

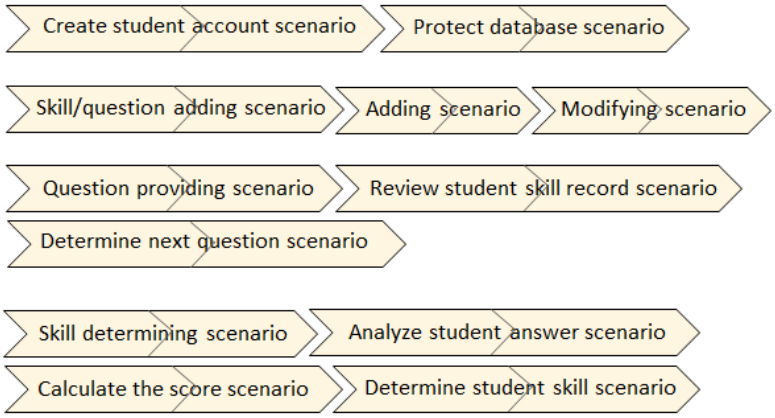

Fig. 8. Scenarios overview diagram

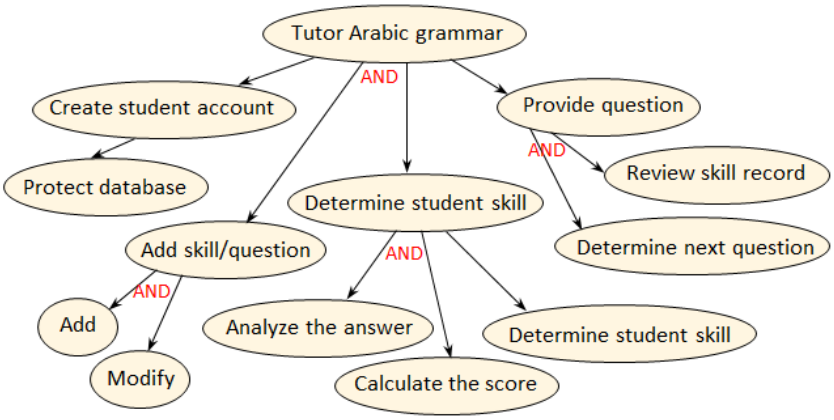

Fig. 9. Goals overview diagra

- *Architectural Design*

This phase is responsible for determining the internal structure of the system. For example, the *agent role grouping diagram* captures decisions about role grouping into agents. Fig. 11 shows the 'create account' role and 'add skill/question' role as part of a teacher agent. Also, 'provide interactive communication' role as being part of user interface agent which is responsible for providing an interactive interface available for the user to interact easily with the system over the network.

Once decisions have been made regarding how roles should be assigned, information can be transmitted from the role specifications to indicate for each agent the percepts and actions associated with it. This data is automatically populated into the *system overview diagram*, which, once completed, offers a visual representation of the internal system architecture. To complete this overview, protocols representing interactions between agents, as well as any data they share should be identified. For example, in our proposed system, the *system overview diagram* (Fig. 12) shows that the question agent receives 'skill record' as a percept from student agent (through skill module). Then, the question agent provides a question based on student skill. It interacts with the student agent via 'providing question' protocol. Similarly, the question agent communicates with the teacher agent via 'adding question' protocol.

- *Detailed design*

This phase focuses on the description of the internal structure of the individual agent's tasks and capabilities, as well as how each agent would do its roles inside the system. The agent's capabilities will match the roles that have been allocated to it, albeit roles may be divided into numerous finer-grained capabilities or combined into a bigger capacity. As illustrated in Fig. 13, the question agent has two capabilities: question determination and question providing both matching the corresponding 'provide question' role.

The teacher agent has two roles: create account and add skill/question which are divided into three capabilities: account creating, skill adding/modifying, and question adding/modifying as shown in Fig. 14. Similarly, the roles of the student agent and the user interface agent are divided into capabilities as shown in Fig. 15 and Fig. 16 respectively.

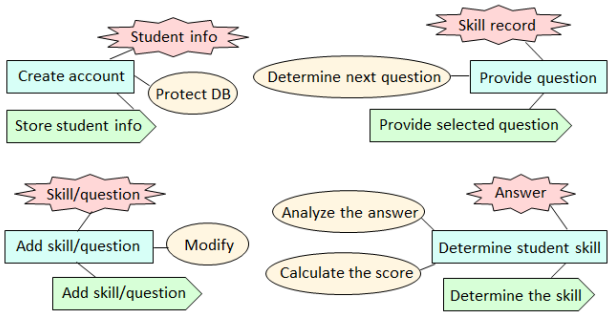

Fig. 10. System roles diagram

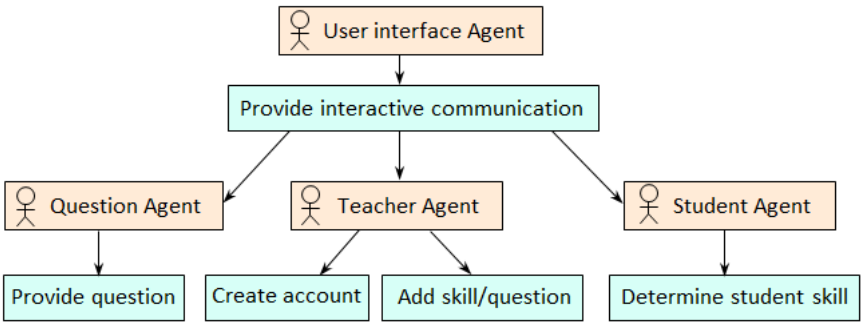

Fig. 11. Agent role grouping diagram

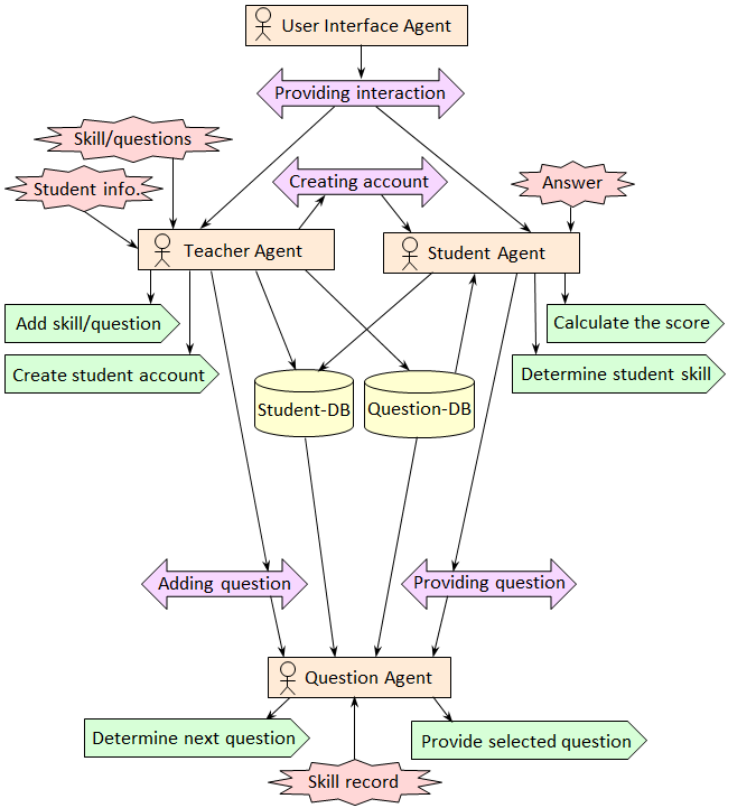

Fig. 12. System overview diagram

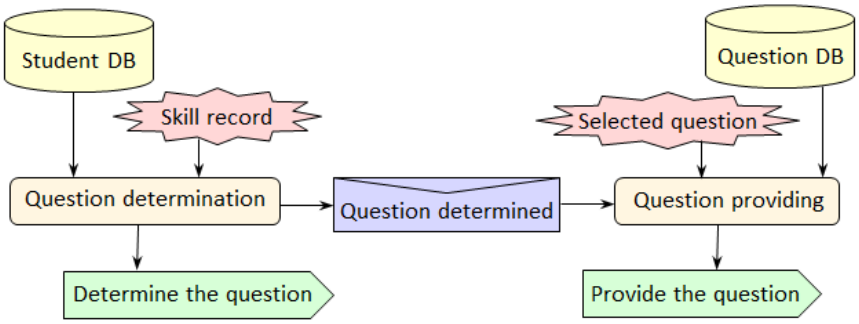

Fig. 13. Question agent overview diagram

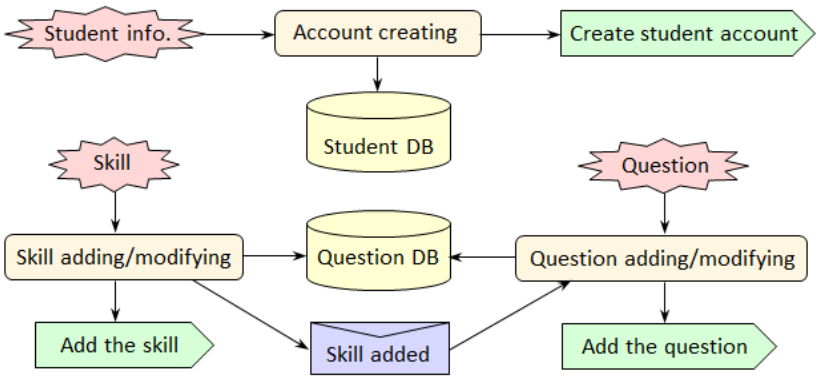

Fig. 14. Teacher agent overview diagram

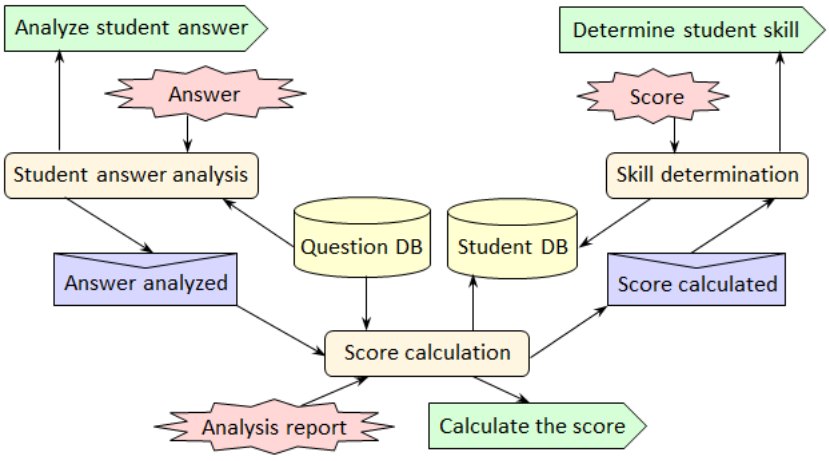

Fig. 15. Student agent overview diagram

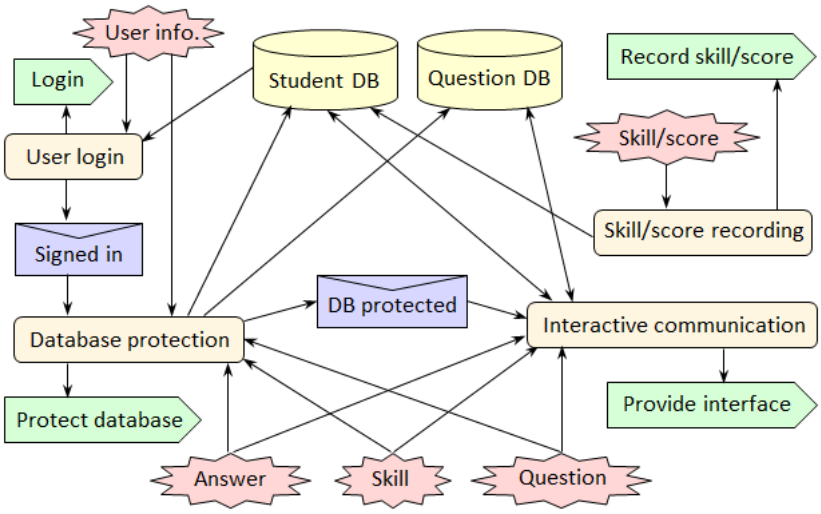

Fig. 16. User interface agent overview diagram

C. *Implementation*

Our proposed system is developed using Java and PHP languages, in addition to MySQL database management system. The Foundation for Intelligent Physical Agents (FIPA) aims to encourage the development of agents including intelligent agents. Java Agent DEvelopment Framework (JADE) version 2, which is a middleware under standard FIPA, is used to implement the agent module. This section examines the implementation of the agent module and the system interfaces.

- *Implementation Using JADE*

FIPA provides several containers for developing agents. The container is an instance of JADE run time comprising Java processes. A platform is a collection of containers [29]. As previously noted, an agent in AOP has a level of autonomy. An agent knows when to do a given behaviour and how and when to interact with other agents without external interaction. This is unlike OOP, in which the object is completely controlled by the programmer. The programmer decides when an object is created, what the object can do, when the object can do a given behaviour and when to interact with other objects and how this is achieved. All agents in the proposed system were implemented using similar steps. This section presents some important methods used in the implementation of *question agent* as an example. Firstly, *Question_Agent* was created and added to the JADE environment by implementing the setup() method as indicated by the code below.

```
import jade.core.Agent;

public class Question_Agent extends Agent{
        protected void setup() {
```

The *setup()* method is responsible for *Question_Agent* initialization and for adding behaviours to the agent. It is much like an object constructor. The constructor is called immediately after the object is created. Similarly, the *setup()* method is called after the agent is created.

Following that, we added behaviour to the question agent, which is a task that the agent may execute, and implemented it as an object of a class that extends *jade.core.behaviours*. Each class that extends *behaviours* must implement the *action()* method, which specifies the operations to be performed while the behaviour is in execution. This is as well as the *done()* method, which returns a Boolean value, to show whether a specific behaviour has completed and should be deleted from an agent's collection of behaviours.

As shown in the code below, for example, we added *OneShotBehaviour()* to the code to make sure that the *action()* method only runs once. The *action()* method of *Question_Agent* performs the task of sending *INFORM* message to the *Sudent_Agent*. This implies that the sender *Question_Agent* only wants to inform the receiver *Student_Agent* about a fact without making an action. A message can generally request doing an action or can be merely a query. Filling in the parameters of an *ACLMessage* object and then calling the Agent class's *send()* method is all it takes to send a message to another agent. This communication among agents is specified by Agent Communication Language (ACL) defined by FIPA.

```
addBehaviour(new OneShotBehaviour() {

public void action() {
        ACLMessage msg=new
ACLMessage(ACLMessage.INFORM);

        msg.setContent("Skill record was
received");

        msg.addReceiver(new
AID("Student_Agent",AID.ISLOCALNAME));
send(msg);
```

- *System Interfaces*

The interface of *teacher home page* is shown in Fig. 17. The left-hand side of the figure shows the active number of students and the number of available training exams. The icons of other related pages of the system appear in the right-hand side. The teacher adds the skills followed by their sub-skills via the *skill adding page* depicted in Fig. 18. She also adds the questions and relates them with skills and sub-skills via the *question adding page* shown in Fig. 19. Students' results appear in the *student result page* shown in Fig. 20.

The *student home page* interface is illustrated in Fig. 21, which shows all available tests and passed tests on the left-hand side. The student can start a test via the *student test page* shown in Fig. 22. Examples of correct and wrong answers of students are shown in Fig. 23 and Fig. 24. All tests' results appear in the *result page* shown in Fig. 25.

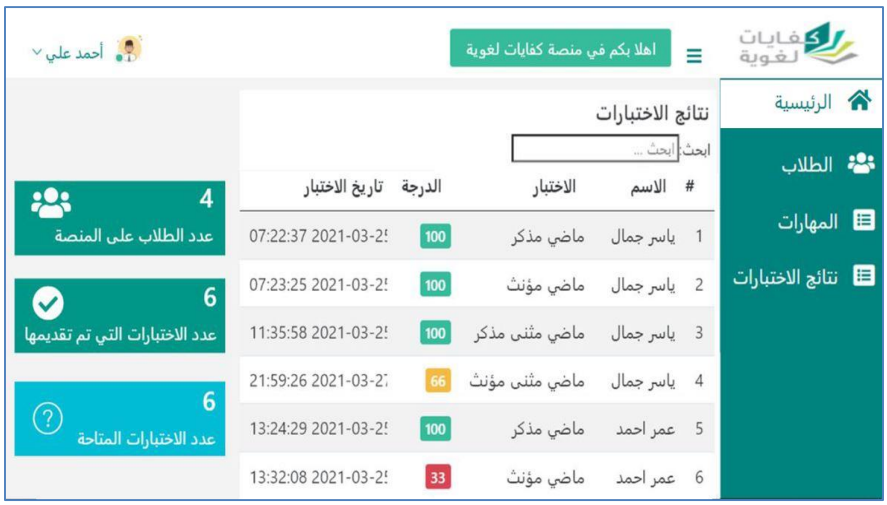

Fig. 17. Teacher home page

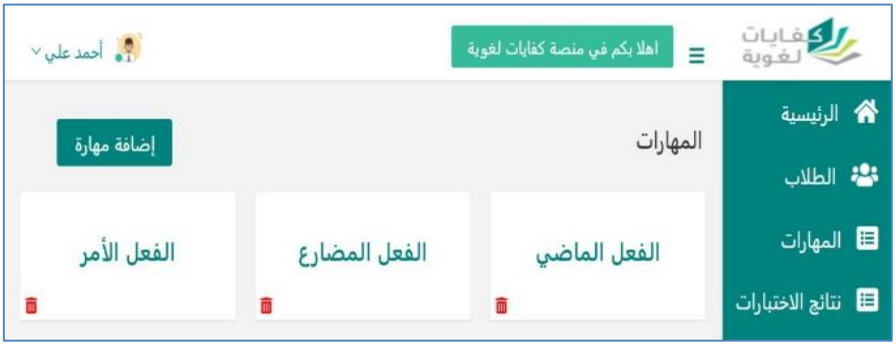

Fig. 18. Skill adding page

Fig. 19. Question adding page

Fig. 20. Student result page

Fig. 21. Student home page

Fig. 22. Student test page

Fig. 23. Correct answer

Fig. 24. Wrong answer

Fig. 25. Test result page

V. DISCUSSION

This paper provided an agent-based ITS for university-level Arabic Grammar. The goal of the system is to seize the student's knowledge and skill level while the student is answering questions, to provide suitable questions according to weak areas. Towards this goal, we inspected the problematic issue of analyzing Arabic language grammar courses and defined related Arabic linguistic skills and sub-skills to attach them to suitable questions. This was followed by proposing a suitable system architecture consisting of different components and modules such as agent architecture, skills and sub-skills, grammar questions, databases, data processing tools, and SQL.

The system integrated intelligent agents in the design of the ITS to facilitate development and future modifications. Meanwhile, the paper provided a framework for developing such systems in a systematic way using PDT and JADE. Recommendations for success of such systems can be summarized as follows:

1. Students have to be aware of the system implementation since they will not be able to move to the next skill or sub-skill unless the current is fulfilled. This will be modified in future versions.
2. Teachers have to be aware of using the system by adding skills and sub-skills then attaching them to suitable questions.
3. In addition to tutoring students, the proposed system can provide teachers with a report of each student, such as the scores of skills and

sub-skills, date and time of solving each question, and tutoring status.

VI. Conclusion and Future Work

In the future, the question bank will contain a huge number of questions covering all groups of questions such as Multiple Choices Questions (MCQ), true/false, match the correct sentence form, press on something, fill in the space with the correct answer from a bracket, get out a verb, a noun, or a particle, parse a sentence, reorder a nominal sentence to be a verbal sentence and vice versa, generate the plural, in addition to providing plural or single form of a noun.

The system will also be improved to generate questions using intelligent agents and to parse free form sentences. CBM will be used to check violated constraints and generate suitable feedback. We would also like to make the system globally accessible. This is of course in addition to extending the system to cover the whole Arabic grammar curriculum.

References

[1] M. Demeuse, and A. Baye, "Efficiency and equity in European education and training systems," 2007, IP/-B/CULT/FWC/2006_169.

[2] M. Wenger, "The restratification of higher education in contemporary America: Dimensions and consequences," *Humanity & Society*, vol. 11, no. 2, pp. 137-151, 1987.

[3] J. Dede et al., "Intelligent computer-assisted instruction: A review and assessment of ICAI research and its potential for education," *Educational Technology Center*, Cambridge, MA, USA, 1985.

[4] I. Padayachee, "Intelligent tutoring systems: Architecture and characteristics," in *Proc. of Sacla Conference*, Port Alfred, 2002.

[5] W. Clancey, "Tutoring rules for guiding a case method dialogue," International Journal of Man-Machine Studies, vol. 11, pp. 25-49, 1979.

[6] K. Butcher and V. Aleven, "Using student interactions to foster rule–diagram mapping during problem solving in an intelligent tutoring system," *Journal of Educational Psychology*, vo. 105, no. 4, p. 988, 2013.

[7] P. Sedlmeier, "BasicBayes: A tutor system for simple Bayesian inference," *Behavior Research Methods, Instruments, & Computers*, vol. 29, pp. 328-336, 1997.

[8] K. Chrysafiadi and M. Virvou, "Student modeling approaches: A literature review for the last decade," *Expert Systems with Applications*", vol. 40, no. 11, pp. 4715-4729, 2013.

[9] C. Carmona and R. Conejo, "A learner model in a distributed environment," in *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, 2004, pp. 353-359.

[10] E. Rich, "Stereotypes and user modeling," in *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer Berlin Heidelberg, 1989, pp. 35-51.

[11] J. Kay, "Stereotypes, student models and scrutability," in *Intelligent Tutoring Systems*, G. Gauthier et al., Eds. Springer Berlin Heidelberg, 2000, pp. 19-30.

[12] A. Grubisic , S. Stankov and B. Zitko," Stereotype student model for an adaptive e-learning system," in *Proc. of World Academy of Science, Engineering and Technology*, vol. 76. 2013, p. 20.

[13] A. Mitrovic, "Fifteen years of constraint-based tutors: What we have achieved and where we are going," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 39–72, 2012.

[14] A. Mitrovic, M. Mayo, P. Suraweera, and B. Martin, "Constraint-based tutors: A success story," in *Engineering of Intelligent Systems*, L. Monostori et al., Eds. Springer Berlin Heidelberg, 2001, no. 2070, pp. 931-940.

[15] H. Mouratidis, P. Giorgini, and G. Manson, "Modelling secure multiagent systems," in *Proc. of 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne, Australia, 2003, pp. 859-866.

[16] Y. Shoham, "Agent-oriented programming," *Artificial Intelligence*, vol. 60, pp. 51-92, 1993.

[17] *Agents and Artificial Intelligence*, *Proc. of 11th International Conference of ICAART*, J. Herik and A. Rocha, Eds., Prague, Czech Republic, 2019.

[18] Y. Zhenzhen et al. "Remote intelligent tutoring system based on multi-agent," in *Proc. of 2nd International Conference on Information Engineering and Computer Science*, 2010.

[19] S. Yu, L. Zhiping, and X. Youming, "A model of intelligent tutoring systems with emotional pedagogical agents," in *Proc. of International Conference on Computer Science & Education*, 2012.

[20] S. Ramachandran, E. Remolina, and D. Fu, "FlexiTrainer: A visual authoring framework for case-based intelligent tutoring systems," in *Proc. of International Conference on Intelligent Tutoring Systems*, 2004.

[21] J. Cardoso et al., "MathTutor: A multiagent intelligent tutoring suystem," *IFIP Advances in Information and Communication Technology*, vol. 154, pp. 231-242, 2004.

[22] E. Sierra, P. Britos, D. Rodriguez, and R. Garcia-Martinez, "A multi-agent intelligent tutoring system for learning computer programming", *in Proc. Electronics, Robotics and Automotive Mechanics Conference*, 2007, pp. 382-385.

[23] T. Heift and D. Nicholson, "Theoretical and practical considerations for web-based intelligent language tutoring systems," in *Proc. of International Conference on Intelligent Tutoring Systems*, 2000.

[24] Y. Tashtoush, "Adaptive e-learning web-based English tutor using data mining techniques and Jackson's learning styles," in *Proc. of 8th International Conference on Information and Communication Systems*, 2017.

[25] N. Khodeir, M. Hafez, and H. Elazhary, "Multi-level skills-based student modeling in an arabic grammar tutor," in *Proc. of International Conference on Advanced Control Circuits Systems*, 2017.

[26] M. Al Emran and K. Shaalan, "A survey of intelligent language tutoring systems," in *Proc. of International Conference on Advances in Computing, Communications and Informatics*, 2014.

[27] M. Hafez, "A multiagents based intelligent tutoring system for teaching Arabic grammar," *International Journal of Education and Learning Systems*, vol. 3, 2018.

[28] L. Padgham and M. Winikoff, "Prometheus: A methodology for developing intelligent agents," in *Proc. of the International Workshop on Agent-Oriented Software Engineering*, 2002, pp. 174-185

[29] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with a FIPA-compliant agent framework, *Software: Practice and Experience*, vol. 31, no. 2, pp. 103-128, 2001.

نظام ذكي معتمد على الوكيل البرمجي لتعليم قواعد اللغة العربية

سمية نذير [١] ، حنان الأزهري [١،٢] ، حنين الأحمدي [1]

[١] قسم علوم الحاسبات ، كلية العلوم و هندسة الحاسب الآلي، جامعة جدة، جدة ، المملكة العربية السعودية

[٢] معهد بحوث الالكترونيات ، القاهرة ، مصر

sahmed0023.stu@uj.edu.sa, helazhary@uj.edu.sa, hhalahamade@uj.edu.sa

مستخلص. يوفر نظام التدريس الذكي (ITS) تعليمًا مخصصًا للطلاب بناءً على تفضيلاتهم التعليمية و مهاراتهم في معالجة مشكلات التعلم. وتشمل هذه المشكلات صعوبة توفير تعليم جيد للطلاب في المناطق النائية ، والتي قد يصعب الوصول إليها من قبل الطلاب أو المعلمين. بالإضافة إلى ذلك ، فإن تتبع تقدم كل طالب على حدة وتوفير المواد والتدريبات التعليمية المناسبة ليس بالمهمة السهلة. تعتبر قواعد اللغة العربية من أفضل المرشحين لمثل هذه الأنظمة ، وهي مادة معقدة للغاية. ومع ذلك ، بسبب هذا التعقيد ، لم يتم التعامل معها بشكل مناسب من قبل الباحثين. أيضًا ، تم تطوير هذه الأنظمة بشكل عام لمادة القواعد النحوية للمستوى الابتدائي. وبالتالي ، يهدف هذا البحث إلى تطوير نظام تعليمي ذكي لمعالجة مشاكل تعلم قواعد اللغة العربية وجعل التعليم قابلاً للوصول في أي مكان بناءً على التعلم الذاتي. تم تطوير النظام لمادة قواعد اللغة العربية بجامعة جدة. الوكيل البرمجي الذكي عبارة عن حزمة من أدوات البرامج يتم دمجها مع مختلف التطبيقات وقواعد البيانات لتسهيل تشغيلها من خلال جعلها معيارية. كما أنها تسهل تعديل وتطوير أي ITS دون الحاجة إلى إعادة بنائها من الصفر. على الرغم من أن الوكلاء البرمجيين قد تم دمجهم في أنظمة تدريس قواعد اللغة العربية ، على حد علمنا ، لم يتم اقتراح أي إطار عمل أو منهجية لتطوير أنظمة التدريس الذكية الخاصة بقواعد اللغة العربية. نحاول معالجة هذه المشكلة في هذا البحث من خلال اقتراح بنية نظام مناسبة تتكون من مكونات ووحدات مختلفة مثل الوحدة التربوية ووحدة قاعدة البيانات ووحدة واجهة المستخدم والوكلاء البرمجيين. قام النظام بدمج عوامل ذكية في تصميم أنظمة التدريس الذكية لتسهيل التطوير والتعديلات المستقبلية. وفي الوقت نفسه ، قدم البحث إطارًا لتطوير مثل هذه الأنظمة بطريقة منهجية.

الكلمات المفتاحيةـ الوكيل البرمجي، قواعد اللغة العربية ، نظام التعليم الذكي.