applied sciences

MDPI

Article

PACA-ITS: A Multi-Agent System for Intelligent Virtual Laboratory Courses

Saima Munawar [1,2,*], Saba Khalil Toor [3], Muhammad Aslam [4] and Esma Aimeur [5]

1 School of Computer Sciences, National College of Business Administration & Economics, Lahore 54660, Pakistan
2 Department of Computer Science, Virtual University of Pakistan, Lahore 54000, Pakistan
3 Department of Computer Science, Forman Christian College University, Lahore 54600, Pakistan; sabakhaliltoor@gmail.com
4 Department of Computer Science, University of Engineering and Technology, Lahore 54890, Pakistan; maslam@uet.edu.pk
5 Department of Computer Science and Operations Research, University de Montréal, Montréal, QC H3T 1J4, Canada; aimeur@iro.umontreal.ca
* Correspondence: saima.munawar@vu.edu.pk

Received: 19 October 2019; Accepted: 21 November 2019; Published: 25 November 2019

Abstract: This paper describes an intensive design leading to the implementation of an intelligent lab companion (ILC) agent for an intelligent virtual laboratory (IVL) platform. An IVL enables virtual labs (VL) to be used as online research laboratories, thereby facilitating and improving the analytical skills of students using agent technology. A multi-agent system enhances the capability of the learning system and solves students' problems automatically. To ensure an exhaustive Agent Unified Modeling Language (AUML) design, identification of the agents' types and responsibilities on well-organized AUML strategies is carried out. This work also traces the design challenge of IVL modeling and the ILC agent functionality of six basic agents: the practical coaching agent (PCA), practical dispatcher agent (PDA), practical interaction and coordination agent (PICA), practical expert agent (PEA), practical knowledge management agent (PKMA), and practical inspection agent (PIA). Furthermore, this modeling technique is compatible with ontology mapping based on an enabling technology using the Java Agent Development Framework (JADE), Cognitive Tutor Authoring Tools (CTAT), and Protégé platform integration. The potential Java Expert System Shell (Jess) programming implements the cognitive model algorithm criteria that are applied to measure progress through the CTAT for C++ programming concept task on IVL and successfully deployed on the TutorShop web server for evaluation. The results are estimated through the learning curve to assess the preceding knowledge, error rate, and performance profiler to engage cognitive Jess agent efficiency as well as practicable and active decisions to improve student learning.

Keywords: agent technology; AUML; cognitive architecture; CTAT; education technology; intelligent virtual laboratory; Jess programming; JADE; ontology engineering

1. Introduction

The physical laboratory is used to assist students in performing practical work [1], but it requires expensive equipment and a high level of expertise. Insufficient instruments in science laboratories, limited time, geographical essentials, and financial anxieties, as well as the preservation of laboratory equipment are major issues in the institutions of under-developed countries. It is a challenging task to circulate practical education online on a large scale in terms of distance and the blending of education modes [2,3]. To resolve these issues, we developed the online practical intelligent virtual laboratory

(IVL) platform, a web-based experimental application by means of which a student can efficiently learn to perform practical tasks (like programming course) anywhere, with student behavior being monitored. The design methodology was as follows: Firstly, the research problem was analyzed through exploratory research by conducting a computer laboratory survey. The participants of the survey were distance learning, e-learning, conventional, and blended learning mode students. We aimed to assess whether the current facilities of computer laboratory tasks can improve students' analytical learning in an applied scientific practical course. We analyzed computer laboratory issues from the perspective of student respondents regarding their learning experience. A questionnaire involving quantitative and qualitative data collection was evaluated. The analysis and challenges of that survey were to measure and identify the major problems faced by students when performing practical tasks, such as the understanding of code syntax and implementation, software or tool issues, an insufficient practical curriculum, failure issues, unavailability of expert course teachers, a lack of training, insufficient laboratory time, and syntax and logical errors. This paper presents the pedagogical agent-based cognitive architecture (PACA-ITS), which is aligned with the intelligent tutoring system (ITS) and was constructed based on the unified theory of cognition for developing IVL courses. It was proposed to overcome the challenges regarding physical laboratory enhancement as a provided solution of the IVL with the assistance of a multi-agent system. Furthermore, this PACA-ITS was constructed based on cognitive modules such as declarative and procedural memory, working memory (WM), long term memory (LTM), semantic memory (SM), episodic memory (EM), procedural memory (PM), and sensory and perception memory. Moreover, it was articulated into three layers as the perspective of the cognitive process. The cognitive process is operated in single or multiple cycles as a design like human brain processing, including sensing, perceiving, focusing on a particular thing through attention, and retrieving information from storage memory by recalling, encoding, and recognizing and then processing the action based on knowledge and learning. Furthermore, this research utilizes cognitive processing to progress the ILC agents that perform and act upon a particular input, like brain simulation, in a number of cognitive cycles. The first layer is a virtual environment that was built as a web graphical user interface (GUI) of the IVL, which senses the students' input and responds as an actuator memory. The second layer has a sensory/motor layer which perceives incoming input selection as perception memory and performs the action based on selected rules. The third layer acts as the cognitive agent processing layer and performs tasks such as focusing on the perceived items and then recalling facts from a temporary store in WM by retrieving and recognizing the knowledge management store in the LTM. It utilizes cues from episodic and semantic knowledge and performs actions based on policy and strategies as PM. This research has the objective of achieving the development of these technologies through the integration of the Java Agent Development Framework (JADE), Protégé, and Cognitive Tutor Authoring Tools (CTAT) platforms. A C++ programming course laboratory is developed as a case study in this research. We also develop the IVL system for a C++ programming course concept using that integration. Furthermore, we expand our research to test and evaluate this cognitive model with the help of Jess programming in a real virtual environment.

The remaining paper has the following sections: An overview of the research has been described. Sections 2 and 3 present the work related to our research. Section 4 presents the multi-agent system design of an intelligent virtual laboratory and provides the mapping implementation of the PACA-ITS framework for C++ programming. Sections 5 and 6 present an infrastructure to enable technology integration for testing, deploying, and evaluating the IVL artifact using CTAT, TutorShop, and DataShop. Lastly, Section 7 concludes the research and a future direction of this domain.

2. Background Theory

A multi-agent system is a self-organized system that performs multiple tasks through the cooperation and coordination of multiple agents. It can solve complex problems which are difficult to handle and attempt concurrently [4]. Further, the intelligent agent has been declared as an autonomous entity which perceives input through sensors and responds to the environment through the actuator. Agents

are categorized into four basic types according to perception and intelligent behaviors such as simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents, and learning agents [5]. The term of agents and objects in object-oriented programming (OOP) have a similar nature in an object-oriented sense but differs in terms of being autonomous, proactive, and goal oriented [6]. Regarding the distinction between objects and agents, Wooldridge stated that "Objects do it for free; agents do it because they want to". It has also been mentioned that agents are active objects because they incorporate a thread of control as opposed to passive objects which are operated by other objects [7].

The agent is designed by Agent Unified Modeling Language (AUML), the agent-based extension of UML graphical modeling language which was proposed by the Foundation for Intelligent Physical Agents (FIPA). It was designed for the modeling and development of agent-based systems. The specifications of agent-based systems are expressed by different notations and agent interaction protocols (AIP). These interactions are expressed by recommending extensions to support the concurrent threads of interaction [8]. Extensions are provided for multiple diagrams of Unified Modeling Language (UML) such as sequence and collaboration diagrams and activity and state charts [9]. These AUML-based modeling diagrams are helpful for the implementation of multi-agent systems [10]. They have also been implemented for multi-agents by Multi-Agent System Modeling Language (MAS-ML) diagrams in Java implementation [11]. The agent-oriented platform JADE is used for the multi-agent system. It is a distributed middleware infrastructure with FIPA specifications and a Java-based framework for developing agent-based applications [12]. Moreover, the Gaia methodology was proposed for analyzing and designing the multi-agent system, and further, it is used for JADE implementation. The steps of this methodology are beneficial for developing the multi-agent system using the JADE platform [13].

The agent's behavioral techniques have been discussed in different fields. For example, Silva and colleagues described agent training and the learning tasks performed by reinforcement learning. Experiments were conducted to generate the curriculum. They claimed that the performance of the proposed work regarding the learning of agents is better than other mentioned techniques [14]. Amir and colleagues discussed different techniques for agent strategies such as machine learning models and deep learning, Markov decision processes, and reinforcement learning [15]. Furthermore, a game theory graph for agent behavior that assumes as a rational decision maker was developed. The graph theory tuples define the state, position, source, and target position of the game [16]. Wu and colleagues described the neurally animated dialog agent (NADiA) agents which were built and developed through a neural network model. The agents have the ability to conduct a conversation between users and agents [17]. Lcarte proposed linear temporal logic (LTL) for performing the multiple agent tasks. The reinforcement learning agent has used this language to specify multiple tasks and their learning skills. Reinforcement learning can allow an agent's behavior to be learnt. The Markov decision process was used to perform actions involved in agent movement from one state to another state. Q-learning methods were used for action selection of the agent learning policy by tuples such as actions, current state, new state, and rewards, which are received from the environment [18]. Grover and colleagues presented a model of an agent interaction graph to generalize the multi-agent system. This model graph shows the nodes as agents and the edges as the interaction between the agents participating. They related examples of these graphs to weak, intermediate, and strong generalization to evaluate the generalization in the multi-agent system [19]. Pöppel presented the behavior of agents through Bayesian theory of mind. The artificial social agents and their navigation tasks were shown [20]. Savarimuthu presented the norm learning for the multi-agent system. An overview of norm learning for agent learning norms was given. The results have contributed to the three phases of learning: communication, observing, and experiential learning. A scenario based on these three learning modes by using Q-learning and the payoff matrix technique of game theory was proposed [21]. Similarly, another author proposed game of trust (GoT) methods for agent teamwork tasks. This is a two-player team game which emphasizes the initial trust related to positive and negative experiences [22]. Metcalf proposed a model of agent behavior by finite

state transducers. The author presented the scenario of a tower game when the interaction between the Q-learning agents and rule-based agents has taken place [23].

Nkambou and colleagues described the four-component architecture of intelligent tutoring systems: the domain model, tutoring model, student model, and interface model. They described all these models in detail, but they mentioned that they left the integration problem open for these models. The domain model focuses on expert knowledge which includes facts, rules, frame, and ontology management. The tutor model emphasizes learning and tutoring strategies based on the domain and student models. The student model maintains the students' knowledge by evaluating the behavior of student actions. The interface model has interaction capability and dispatches all of this knowledge to students to allow them to understand the problem set and enhance the cognitive skills of students [24]. The expert module representation acquired from domain knowledge methods and techniques has also been discussed. The epistemological issues have also been mentioned from the perspectives of philosophy and psychology, with artificial intelligence (AI) and education sciences being emphasized. Different general purpose languages in AI for representing domain knowledge have also been mentioned such as production rules, semantic networks, conceptual graphs, a frame-based system, and ontology, which provide the pedagogical languages. We have also analyzed some tools, such as CTAT and Authoring Software Platform for Intelligent Resources in Education (ASPIREs), which have facilitated domain knowledge extraction using production rules and constraints [25].

Mizoguchi discussed the modeling of tutor actions and strategies with the help of the ontology engineering domain. Omnibus and Smarties tools were used to develop the ontology of instructional theories [26]. Woolf suggested new paradigms and methods for improving student modeling that can be used to update the tutor's performance. Individual learner preferences and interest, learning goals and outcomes, motivation, and engagement have also been discussed. It is further suggested that pedagogical strategies should be improved based on learner skill level [27]. Psyché emphasized ontological engineering (OE) development, which can perform a significant role in intelligent tutoring systems (ITS). The construction intelligently assisted ontologies (CIAO) architecture has been represented. The integrated OE methodology MI2O has been proposed, which provides the details of five phases of developing education through design standards and paradigms from the perspectives of the learning model, domain model, and study model. These include environment and feasibility study; ontology modeling, including conceptualization and formalization; and operational ontology implementation for system deployment, evaluation, conceptual ontology, and formal ontology documentation [28]. Forgy invented the rete algorithm for the pattern matching of a large set of patterns to a large set of objects by comparing its representation of the patterns. This algorithm has been used in multiple scenarios. The operation of the rete algorithm has been performed on the rule-based knowledge system and WM contents. There are three basic types of operation: match, conflict resolution, and action. Firstly, the "if" part of production rules is matched to the current factual knowledge of WM. If it is matched by part of the production rules, then the action part of the rules is executed. If it does not satisfy the given unordered rules, then the interpretation is halted. This task is performed through one or multiple iterations [29].

The above discussion shows that the multi-agent system plays a role in solving complex problems which are difficult to handle and attempt concurrently, such as the analysis of the design of AUML modeling languages. Also, although the domain model, student model, tutoring model, and interface model have been described in detail, the integration of these models is still left open for research. In this research, we handle the integration problem with cognitive architecture and proper infrastructure of the enabling technologies.

3. Related Work

Virtual laboratory is considered as the perspective of simulated real experiments without walls and doors, paper and pen [30]. It can also utilize a virtual learning environment that stimulates like a real laboratory. The evolution of virtual labs has intensely renovated the laboratory education

landscape [31]. The virtual labs can be cohesive into the traditional lab to endorse and improve students' learning in the 21st century, particularly in virtual classes. Virtual laboratories can run in analogue to traditional labs to enhancement, support, and boost student learning in education mode [32,33]. Several VL projects are being built from different standpoints such as iLab project [34], Lila project [35]; Laboratory Share [36], Web Laboratory-Deusto project developed [37], Go-Laboratory school project [38], Global Online Laboratory Consortium (GOLC) Laboratory2Go6 portal [39], Open Wonderland project [40], an online lab project [41], and a Deeptutor project [42].

It begins with an intensive literature review leading to the design architecture of an ILC agent for an IVL. To pledge an exhaustive literature review, the identification and selection of the primary works related to VL in the context of programming which was based on well-organized search strategies that were packed away through the "virtual laboratory" search term. The possible searching which was relevant to the publication during 2007 to 2019 years in selective databases of Institute of Electrical and Electronics Engineers (IEEE), Springer, Association for Computing Machinery (ACM), and Elsevier. The search process identified a total of 2906 studies, out of which 57 have been thoroughly analyzed according to the predefined systematic literature review (SLR) protocol. This SLR aims to find the existing model of VL and identify potential gaps for future research. Based on research topics in selected studies, the authors identified seven main categories: model which are cognitive skills, analysis, software tools, infrastructure, automated assessment tools, and multi-agent architecture. The author has presented the overall selected 57 primary studies of VL obtained in the SLR which emphasized the trends of VL research. It shows that the design and platform type of VL in scientific databases according to publication types as shown in Appendix G.

The literature review revealed that mostly all existing studies were modeled for a specific automation solution to perform practical tasks like simulation and recording student activity, but not general resolution, which is available for using intelligent systems in a virtual laboratory task. This is mainly due to the lack of standardization in the integration of different aspects such as cognitive architecture, expert system, and machine learning algorithms, which is a major difficulty that has stalled general automated solutions virtually to do the lab task. A vital challenge in applying cognitive architecture and machine learning methods is to teach and train students online without any human being intervention and provide new pedagogical methods for improving student learning. Hence, a standardized framework is needed to facilitate these events and impulse the students learning while studying virtually and blended practical course implementation.

4. Materials and Methods

An IVL multi-agent system has the main ILC agent as the generalization agent. This includes specialized agents such as the practical coaching agent (PCA), practical dispatcher agent (PDA), practical interaction and coordination agent (PICA), practical expert agent (PEA), practical knowledge management agent (PKMA), and practical inspection agent (PIA). The functionality of these agents with respect to the lab administrator (LA), lab instructor (LI), and student are determined through agent-based use case modeling for IVL. The IVL system functionality signifies agent-based case modeling, which includes external actors and internal actors of system functionalities. LA, LI, and students are represented as external actors and ILC agents expressed as internal actors and active objects. In this section, we present the AUML modeling languages, which are used to demonstrate the interactions between these ILC agents and specialized agents and communication with the eLearning students according to the knowledge level.

The ILC agents firstly interact with the LA who is responsible for managing laboratory tasks and managing the activities of the LI and students. They also view the activity of the ILC agents and students. The ILC agent initiates when students select a practical training and monitoring session.

The ILC agent assists with requests to initiate the selection of the practice session from the student login. The student can select the option of the practical training session and practical monitoring session. The practical coaching agent is initiated to manage the practical training session, and it

initiates all practical coaching agent (PCA), practical dispatcher agent (PDA), practical interaction and coordination agent (PICA), practical expert agent (PEA), practical knowledge management agent (PKMA), and practical inspection agent (PIA) are to manage the practical monitoring session. It retrieves the student registration data and responses to the lab administrator using the performance of agents in terms of their activity. The identification of multi-agent types and responsibilities is given below.

4.1. Identification of Multi-Agent Types and Responsibilities

The practical coaching agent (PC Agent) retrieves the students' information. It manages the initial assessment test to measure the knowledge levels of students. It creates a training plan and manages the knowledge related to the course curriculum. It creates a pedagogical learning methodology according to the program. It responds to the requests from the ILC. The practical dispatcher agent (PD Agent) manages the protocol to send goal and planning information from one agent to another agent. It retrieves information and communicates with other agents. The practical interaction and coordination agent (PIC Agent) retrieves and stores the students' session information in the student experience memory. It creates the student activity path by interface mapping and manages the keyboard and mouse points by kinesthetic operations. The practical expert agent (PE Agent) retrieves the information about a particular course and generates a simulation path. It retrieves the delay in given tasks and the occurrence of syntax errors then provides multiple directions to students like hints, tutorials, etc. If it gets the wrong output and a logical error occurs, then an improvement plan is provided to a student. It retrieves the sequence of objects and creates the semantic plan. The practical knowledge management agent (PKM Agent) retrieves information cues from the LTM. It retrieves information and stores the student experience path by tracing the activity. It stores and retrieves the episode-wise session store from the episodic memory. It manages the attention memory of students. The practical inspection agent (PI Agent) measures the students' strength and generates a student performance graph. It retrieves the output and generates the scrutiny path of the students. It manages the assessment tests according to Bloom's taxonomy wise and tracks the students' activity.

Challenges to Implementing the System with the JADE Platform

The IVL infrastructure used the JADE platform to develop the multi-agent system. JADE software is based on Java language with a swing and an Abstract Window Tool Kit, and it is integrated with runtime Java platforms such as Eclipse and Netbeans IDE. We can use multiple libraries in Java language to manipulate the JADE platform, such as import jade.core.Agent; import jade.core.behaviors, import jade.gui.GuiEvent, and import jade.lang.acl.ACLMessage [43]. The initial infrastructure of IVL was built through the JADE platform, as shown in Figure 1. This IVL multi-agent based on the JADE platform includes four hosts. The containers are built in these hosts. The main container of JADE has an ILC agent and default agents such as the agent management system (AMS), directory facilitator (DF), and remote management agent (RMA). Agents are built in Host1:132.204.26.1. Container 1 is associated with agents PC Agent 1, PD Agent 1, PIC Agent 1, PE Agent 1, PKM Agent 1, and PI Agent 1, which are built in Host2:132.204.26.2. Container 2 is associated with agents PC Agent 2, PD Agent 2, PIC Agent 2, PE Agent 2, PKM Agent 2, and PI Agent 2, which are built in Host3:132.204.26.3. Container 3 is associated with agents PC Agent 3, PD Agent 3, PIC Agent 3, PE Agent 3, PKM Agent 3, and PI Agent 3, which are built in Host4:132.204.26.4. These containers are registered with the main JADE container. This system responds and stores the sessions of the web interface. The Java classes are extended to build agent libraries in eclipse IDE and run the platform of remote agent management (GUI RMA) of JADE, as shown in Appendix A, for the ILC agents' containers in the JADE remote agent management platform.

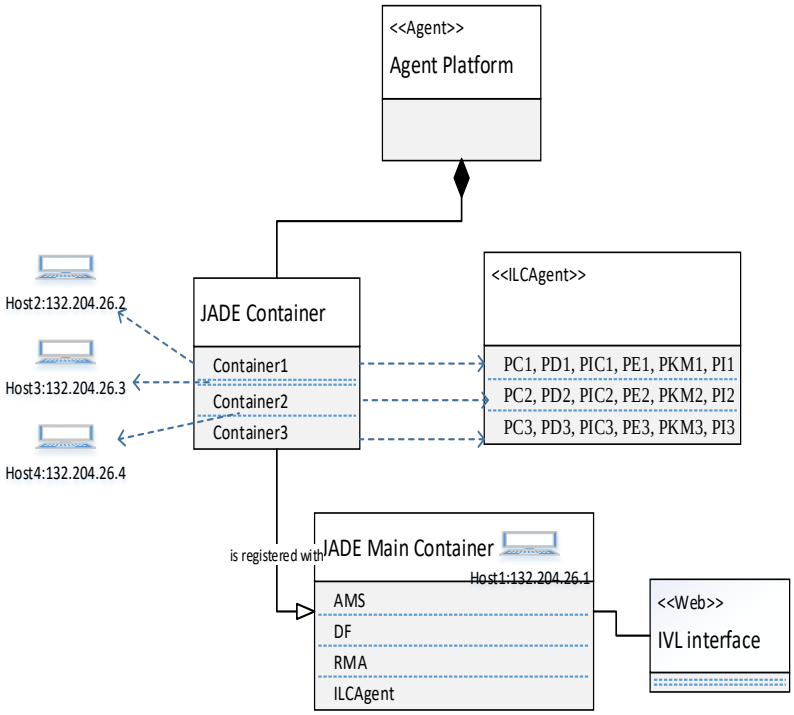

Figure 1. Infrastructure of the intelligent virtual laboratory (IVL) multi-agent system based on JADE structure in host computers network.

4.2. Mapping the Implementation of PACA-ITS For C++ Programming

PACA-ITS has been designed for IVL by constructing cognitive modules using the perspective of intelligent tutoring system (ITS) components, such as the domain/expert model, pedagogical model, learner/Student model, and interface model. The cognitive cycle of a PACA-ITS is composed of these tutoring modules and acts according to the cognitive processes, as shown in Figure 2. The cognitive modules within each tutoring component are worked by processing knowledge that is conceived by the sensing, perceiving, focusing, recognizing, temporary and permanent restoration, recalling, encoding, and decision making based on these affirmations. The emphasis of this research is to implement a tutoring system that senses and learns to correspond to a cognitive cycle in human mind simulations. We mapped components into the PACA-ITS architecture. There are four basic knowledge units of PACA-ITS. The domain knowledge unit to be learned is composed of WM, LTM, SM, EM, the associative memory, and a subset of the learner model. The learner model evaluates and maintains the students' solutions using the attention memory which is traced to model knowledge. The model tracing is performed using expert knowledge traced to student observations and actions through a pattern matching algorithm—the rete algorithm. It represents the student knowledge and skill levels traced by Bayesian network knowledge tracing, and it updates the pedagogical model. The pedagogical model acts as the procedural memory task, which has a strategic plan based on a set of actions, learning objectives, and pedagogical ontology from the domain model. Pedagogical action responses are based on a learner activity and domain model that links to an interface model through Q-learning strategies. The interface model reflects an interaction between the student/learner and tutor agents by feedback, hints, decisions, and skills judgment from expert domain knowledge. The interface model perceives the student action pattern by event classification and responds to complete cognitive cycle modules using knowledge units and the reinforcement learning of PACA-ITS. The details of each of these knowledge units and their cognitive modules are discussed below under the perspective of the C++ programming tasks domain. We think that the practical performance of any task is comparatively more powerful and effective than just listening and memorizing the lecture. We developed two problem set scenarios based on the array concept by using Jess programming code. The students can understand and perform these

problem tasks with the help of ILC agents. These scenarios are mapped on PACA-ITS. The first array problem is "write a program to enter five integer numbers into an array named 'a'". This involves finding the sum of array elements used for the loop which includes header files, array declaration and initialization, and the sum of array integers. The expert agent responds to the student at each step by model tracing and matches the pattern through the rete algorithm. The skill performance tag also shows the student's performance when performing the practical task. The second scenario is "write a program to declare and initialize three floating numbers into an array named 'a'" This includes the declaration and initialization of array values. When students perform the wrong step, the expert agent responds with a message to correct the action using hints and instructions.

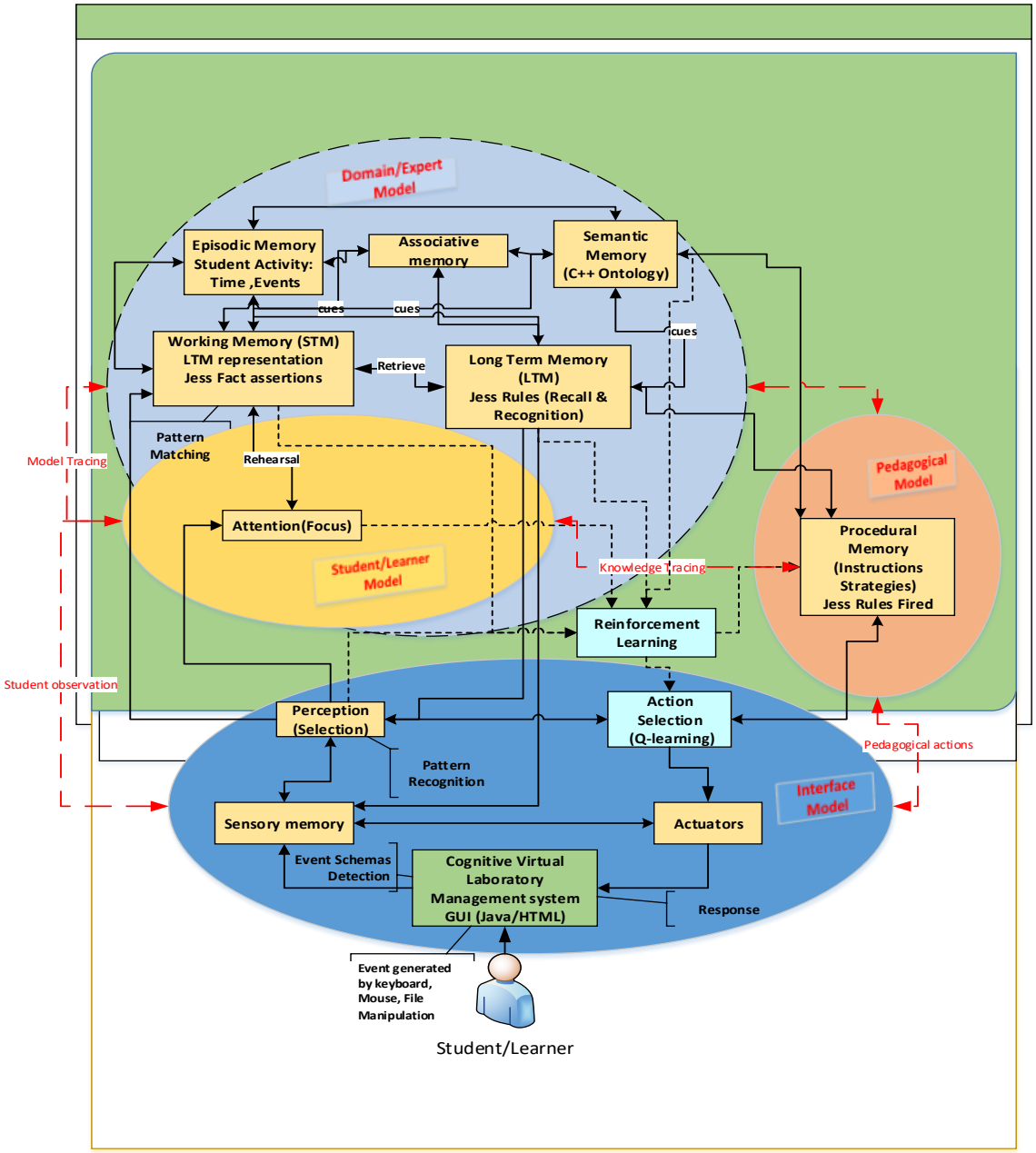

Figure 2. Framework of the pedagogical agent-based cognitive architecture (PACA-ITS) for a virtual laboratory aligned with intelligent tutoring system (ITS) components.

Integration Algorithm for Student–Agent Interaction

The integration algorithm for student–agent interaction through PACA-ITS involves intelligent tutoring system components such as the domain model, pedagogical model, student model, and interface model using the inner and outer loops in the form of pseudo-code, as given below.

```
Input: Text input from student actions through the Java/Html Cognitive GUI interface
Output: Student performance while practicing the C++ concept of the practical task problem.
Begin
    if (student login = true) then
    State = active
    for (Student_State = active, (Student_State =! null), Student_State ++)
    for (timespan = start, timespan = required time, timespan++)
    #Start the student behavior recording by State machine graph until the task is completed
if (student text input by keyboard or mouse = detected by the sensor) Then
#(Interface Model)
    While Loop (event sequence = finite event)
    Activate the Expert agent by the plugin JadeJessProtege with Jade
    Record the skill object map for each state
    Start the Bayesian network for the knowledge tracing of student skills
if (button pressed = hint)
    Display instructions
    else
    Continue working
    else
    if (button pressed = done), then
if (done = Null) then
    Continue working until the task is not completed
if (Perceive the input by pattern recognize = true) then
    if (student selection = focus) then
    Store the selection focus in the attention memory
#(Student Model)
    Start the model tracing
    if (Working memory state = active) then
(Jess fact assertion and deftemplate in Domain Model) then
Start pattern matching by the rete method
    Retrieve Jess Rules from the LTM
(Build Jess defrules maintained in the LTM by production rules.pr)
inside Loop
    Recall Jess rules activated by the Conflict tree form
    Retrieve cues from the semantic memory = active
    Start Ontology mapping using the JessTab plugin with protégé
    Get the C++ ontology concept by Jess Console query
    if (Jess Engine = cleared) in pedagogical model by CTAT compiler, then
# (Pedagogical Model)
    if (Fact assertion match = concerned Jess rule by recognize) then
    Fire the Jess rule = Match
    Predict rule selection and input and action = Student selection and input and action match
    end if
    end if
    end inside Loop
```

end if
Record the student's behavior by the state graph = true then
Draw state graph
Move next state
else
Buggy Rule fired
Display error message
if (hint button pressed = true) then
Expert agent response = true
Jess Message display
if (pressed button = done)
if (done =! Null), then
Update the Student behavior and knowledge state in a student model by clustering
#(Student Model)
Update the Pedagogical model agent strategies and policy = student current_state
Store the student's state by time and event management event session as episode wise in the episodic memory
Agent performs the actions through reinforcement learning
if (student_state = current_state)
Update On-policy in the pedagogical Model by the SARSA method
Agent Action performed
else
if state = previous + current state
Update off-policy by the Q-learning method
Agent performs action
end if
end if
end if
end if
Deactivate Agent
End while Loop
Measure the Student's Performance
Return the student state knowledge Q-table
Return the student knowledge graph
end if
end for Loop
end for Loop
end if
End

4.3. Domain/Expert Model

The domain model of PACA-ITS emphasizes the representation of expert knowledge based on C++ programming taxonomy. The domain model represents declarative and procedural knowledge such as the working memory (WM), which includes the assertions related to problem sets which retrieve the knowledge from the LTM, such as production rules and current/older facts, by recall and recognition. LTM is further divided into explicit and implicit memory. The explicit memory includes semantic memory which stores the knowledge of C++ programming concepts and maintains the ontology. It is associated with episodic memory, which stores the students' session times and current events. The learner model is also a subset of the domain model. This cognitive model was developed

through Jess programming language. The mapping of these memories is in accordance with C++ programming, as follows.

4.3.1. Long Term Memory (LTM)

The LTM is the knowledge-based system (KBS) which was developed through Jess production rules for specific C++ programming tasks and which decides the student's task level and launches Rules. The Jess production rule of the array problem set was developed with the Eclipse Java platform, as shown in Figure 3, under the production name "Rules" with the .pr extension file. The defrule function is used to declare and develop rules for each step of the problem using the WM template elements by the name of (wmeTypes.clp) code to execute the task. Almost 86 defrule functions have been developed for this array problem. The LTM performs encoding, storage, and retrieval tasks by recalling all rules for activation and recognizing the activated rule. The WM initiates the recall process by student attention, and the LTM transfers the encoding back to the WM.

```
(require* wmeTypes "wmeTypes.clp")

(defrule FirstHeadColumn1
    ?problem <- (problem
        (interface-elements $? ?table $?))
    ?table <- (HTable (hcolumns $? ?first-column ?second-column ?third-column ?fourth-column ?fiftth-column ?six-colu
    ?first-column <- (HColumn
        (hcells $? ?first-h ?second-h))
    ?second-column <- (HColumn
        (hcells $? ?third-h ?fourth-h))
    ?third-column <- (HColumn
        (hcells $? ?fitth-h ?six-h))
    ?fourth-column <- (HColumn
        (hcells $? ?seven-h ?eight-h))
    ?fiftth-column <- (HColumn
        (hcells $? ?ninth-h ?tenth-h))
    ?six-column <- (HColumn
        (hcells $? ?eleven-h ?twelve-h))

    ?first-h <- (Hcell (name ?cell-name1)(value ?num1))
    ?second-h <- (Hcell (name ?cell-name2)(value ?num2))
    ?third-h <- (Hcell (name ?cell-name3)(value ?num3))
    ?fourth-h <- (Hcell (name ?cell-name4)(value ?num4))
    ?fitth-h <- (Hcell (name ?cell-name5)(value ?num5))
    ?six-h <- (Hcell (name ?cell-name6)(value ?num6))
    ?seven-h <- (Hcell (name ?cell-name7)(value ?num7))
    ?eight-h <- (Hcell (name ?cell-name8)(value ?num8))
    ?ninth-h <- (Hcell (name ?cell-name9)(value ?num9))
    ?tenth-h <- (Hcell (name ?cell-name10)(value ?num10))
    ?eleven-h <- (Hcell (name ?cell-name11)(value ?num11))
    ?twelve-h <- (Hcell (name ?cell-name12)(value ?num12))
```

Figure 3. Jess programming production rules for the array problem in the Eclipse workspace.

4.3.2. Working Memory (WM)

The WM receives student events with the attention memory using the classification of perception memory objects. The working memory has two main parts: the factual knowledge about attributes of things represented in a template file and stored in slots, or multi-slots like the array problem (deftemplate MAIN::DColumn (slot name) (multislot Dcells) (slot dposition) (slot description)), which is stored in wmeTypes with the clp extension file, as shown in Figure 4. These slots are used in Jess defrules and defunction. The cognitive model's working memory has a collection of facts in a fact assertion file which stores the initial facts of variables which are used in the working memory under the array problem name with the .wme extension, as shown in Figure 5. These collections of facts can be modified and inspected. The percentage of facts changes per unit of time by removing old facts when new facts arrive. The rules are designated according to the relevant case performed by the student during a single cognitive cycle or multiple cognitive cycles in seconds. The pattern can be recalled, matched, and encoded to the factual knowledge of the WM. The production rules of the LTM fetch the fact information from the WM and perform pattern matching using the rete algorithm. These actions are performed in multiple cognitive cycles in seconds. The knowledge base matches the most relevant information to the current contents that reside and control all contents of the currently executed facts of the working processes. The relationship between matches is that the number of

production rules and the number of facts in the knowledge base of the executed pattern is the average number of patterns per rule on the LHS, and the RHS rule actions are applied based on current facts.

```
(deftemplate MAIN::DCell
   (slot name)
   (slot value)
   (slot row_number)
   (slot column_number))
(deftemplate MAIN::DColumn
   (slot name)
   (multislot Dcells)
   (slot dposition)
   (slot description))
(deftemplate MAIN::DTable
   (slot name)
   (multislot Dcolumns))
(deftemplate MAIN::HColumn
   (slot name)
   (multislot hcells)
   (slot position)
   (slot description))
(deftemplate MAIN::Hcell
   (slot name)
   (slot value)
   (slot row_number)
   (slot column_number))
(deftemplate MAIN::HTable
   (slot name)
   (multislot hcolumns))
(deftemplate MAIN::ArrayFieldSize
   (slot name)
   (slot value))
(deftemplate MAIN::hint
   (slot now))
(deftemplate MAIN::problem
   (slot name)
```

Figure 4. Jess programming working memory elements code for an array problem in the Eclipse workspace.

```
(require* wmeTypes "wmeTypes.clp")

;;;; Fact assertions: slot assignments are below.

(bind ?var1 (assert(MAIN::hint)))
(bind ?var2 (assert(MAIN::problem (name Start))))
(bind ?var3 (assert(MAIN::HColumn (name Table1_column1))))
(bind ?var4 (assert(MAIN::HColumn (name Table1_column2))))
(bind ?var5 (assert(MAIN::HColumn (name Table1_column3))))
(bind ?var6 (assert(MAIN::HColumn (name Table1_column4))))
(bind ?var7 (assert(MAIN::HColumn (name Table1_column5))))
(bind ?var159 (assert(MAIN::HColumn (name Table1_column6))))
(bind ?var8 (assert(MAIN::HTable (name HeaderFile))))
(bind ?var9 (assert(MAIN::Hcell (name HeaderFile.R0C0))))
(bind ?var10 (assert(MAIN::Hcell (name HeaderFile.R1C0))))
(bind ?var11 (assert(MAIN::Hcell (name HeaderFile.R0C1))))
(bind ?var12 (assert(MAIN::Hcell (name HeaderFile.R1C1))))
(bind ?var13 (assert(MAIN::Hcell (name HeaderFile.R0C2))))
(bind ?var14 (assert(MAIN::Hcell (name HeaderFile.R1C2))))
(bind ?var15 (assert(MAIN::Hcell (name HeaderFile.R0C3))))
(bind ?var16 (assert(MAIN::Hcell (name HeaderFile.R1C3))))
(bind ?var17 (assert(MAIN::Hcell (name HeaderFile.R0C4))))
(bind ?var18 (assert(MAIN::Hcell (name HeaderFile.R1C4))))
(bind ?var160 (assert(MAIN::Hcell (name HeaderFile.R0C5))))
(bind ?var161 (assert(MAIN::Hcell (name HeaderFile.R1C5))))
(bind ?var19 (assert(MAIN::ArrayFieldSize (name ArraySize))))
(bind ?var20 (assert(MAIN::DTable (name DeclarationArray))))
(bind ?var21 (assert(MAIN::DColumn (name DTable_column1))))
(bind ?var22 (assert(MAIN::DColumn (name DTable_column2))))
(bind ?var23 (assert(MAIN::DColumn (name DTable_column3))))
(bind ?var24 (assert(MAIN::DColumn (name DTable_column4))))
(bind ?var25 (assert(MAIN::DColumn (name DTable_column5))))
```

Figure 5. Jess programming fact assertion code for an array problem in the Eclipse workspace.

4.3.3. Semantic Memory (SM)

The SM has schemas which declaratively represent the ontology of the C++ programming task in the perspective of domain knowledge. The ontology is maintained by categorizing the hierarchy of C++ concepts and their relationships. This declarative knowledge is also related to the procedural knowledge of the pedagogical model and student learning model. It provides the structure and systematic knowledge of C++ taxonomy and IVL system hierarchy. This declarative knowledge can be reused for all concepts of practical C++ programming tasks. In this paper, we represent the complete ontology development for programming agents' knowledge by fulfilling the expertise of the domain model, pedagogical models, student model, and interface mapping model in a grid view of ontology, as shown in Figure 6, but we demonstrate two scenarios which relate to the array concepts. This ontology was developed by ontology engineering languages such as Web Ontology Language (OWL), Resource Description Framework (RDF), and XML for a cognitive Jess tutor in the programming domain with the Protégé tool.

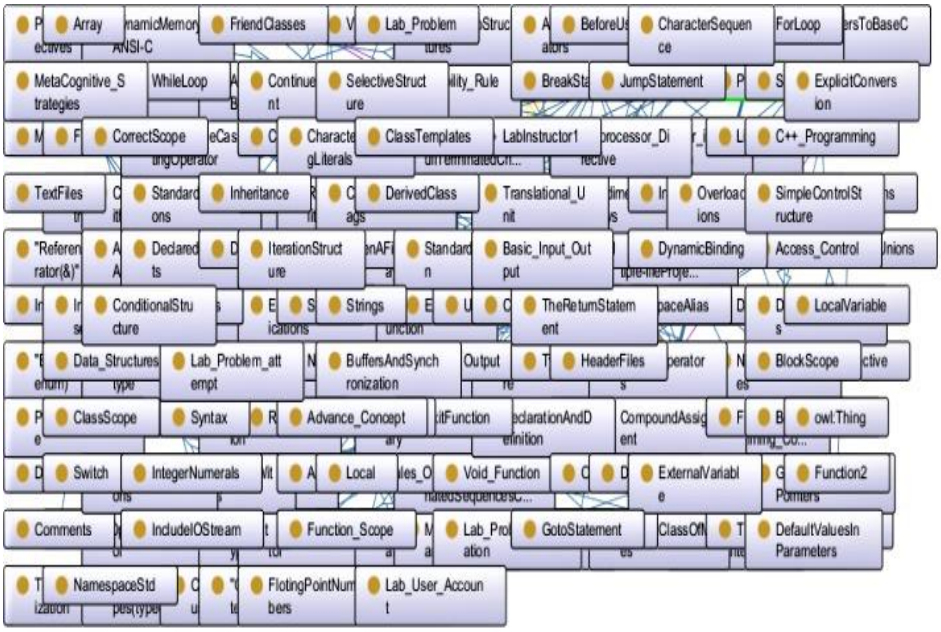

Figure 6. Grid view of the C++ programming ontology for the IVL.

We represented the complete OWL graph in protégé, which uses node types such as classes, individuals, and relationships expressed in arc types such as individuals, subclasses, the management of the practical course, the inheritance of managing users and studies, and the teaching of a course of C++ by students and teachers, as shown in Figure 7.

The Simple Protocol and RDF Query Language (SPARQL) query plugin is used to access the ontology hierarchy in protégé, as shown in Figure 8, to access classes and subclass relationships and to select, update, and insert the existing ontology according to mapping functions.

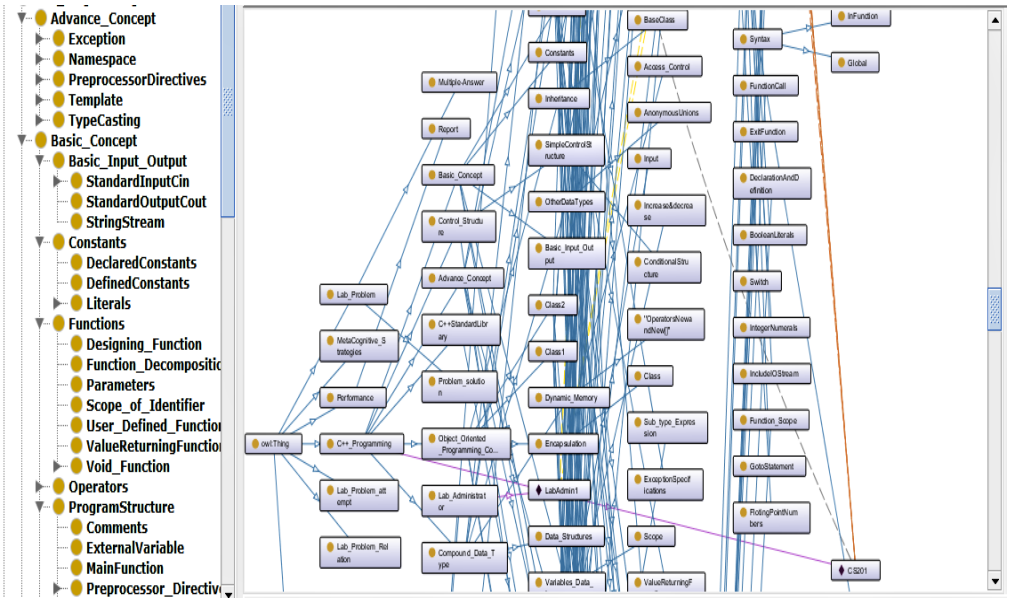

Figure 7. IVL ontology for C++ programming concepts with node and arc types in Protégé.

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?x (STR(?lab) AS ?label) WHERE {
        ?x rdf:type owl:Class .
        OPTIONAL {?x rdfs:label ?lab}
}
ORDER BY ?label

Execute
```

Figure 8. SPARQL Query Plugin for C++ programming.

4.4. Pedagogical Model

The pedagogical model emphasizes procedural memory tasks which involve learning decisions and strategies and predicts student performance based on single or multiple cognitive cycles of all cognitive modules such as WM facts, LTM production rules, and the behavior state graph of current actions performed by the student. It makes decisions based on the domain model developed from expert knowledge and the student model to determine student actions and behaviors. These student observations, selections, and action behavior tasks are recorded by model tracing. The basic purpose of model tracing is to match the pattern to the current facts and rules from declarative memories. The rete algorithm is used to match the pattern using the node network structure. There are two basic parts: compilation rules and runtime execution. These compile the rules in production nodes in the network and execute the actions by terminating nodes. This model has the Jess console interpreter engine which fires the rules according to observations and predicts the students' behaviors. Cognitive model files are cleared by the engine, such as WM elements template files, fact assertions, and defrule functions, according to the problem set, and then the behaviors are observed and the cognitive model functionality is accepted. The student observation behaviors are recorded by the state diagram which is matched by cognitive modules, as shown in Figure 9, which records correct answers, incorrect answers, hints, instructions, and buggy behaviors of the array program. It also updates the pedagogical model according to knowledge tracing from the student/learner model.

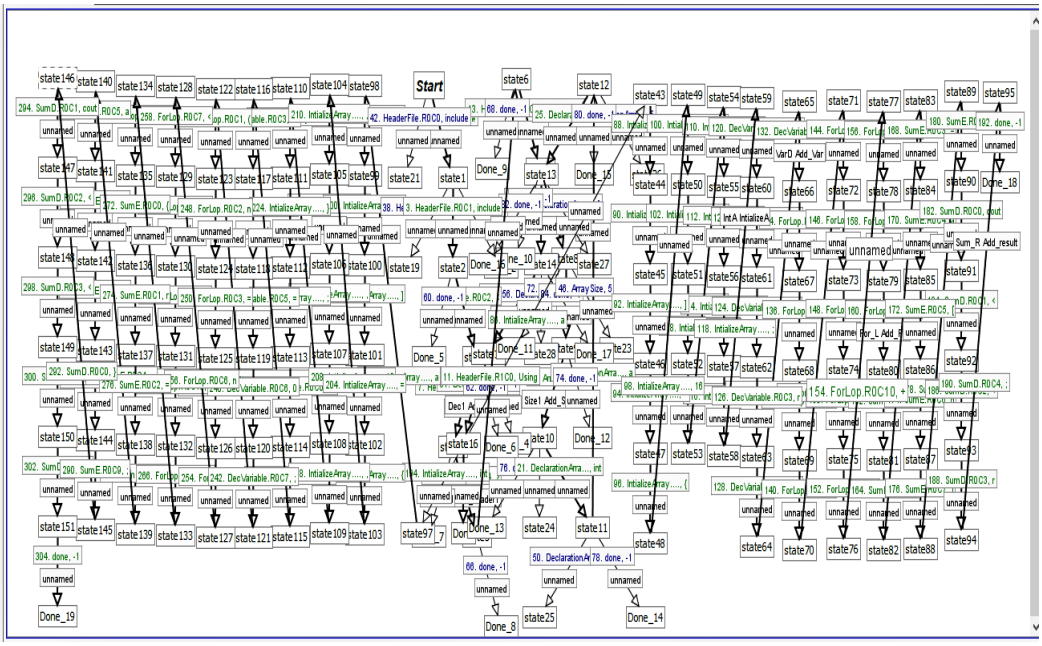

Figure 9. State diagram of the student behavior recorder by model tracing.

4.5. Learner/Student Model

The student model emphasizes student knowledge by evaluating student actions and performance. The attention memory focuses on perceiving the students' steps and selecting the actions from a single or multiple cognitive cycles while the WM is proceeding. It perceives the skill set of student performance by knowledge tracing while recording student observations, as shown in Figure 10. The knowledge tracing can be traced by different techniques, such as the Bayesian network, data mining, and open learner modeling [24].The student input is observed by the interface model which is traced by a model tracing from recalling, pattern matching, and the encoding process of cognitive modules of the domain model. The student model is the subset of the domain model which updates the domain expert knowledge according to the current knowledge component received through skill set items. The pedagogical strategies also update the Jess fired rules according to the expert knowledge of the domain model and the skill knowledge component matrix of a student model.

Step Name	Attempt At Step	Is Last Attempt	Outcome	Selection	Selection	Action	Action	Input	Feedback Text	Feedback Classification	Help Level	Total Num Hint
SumD.R0C1 UpdateTextArea	1	0	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
HeaderFile.R0C0 UpdateTextArea	1	0	INCORRECT	HeaderFile.R0C0		UpdateTextArea		1				
HeaderFile.R0C0 UpdateTextArea	2	1	CORRECT	HeaderFile.R0C0		UpdateTextArea		#				
SumD.R0C1 UpdateTextArea	2	1	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
HeaderFile.R0C0 UpdateTextArea	1	1	CORRECT	HeaderFile.R0C0		UpdateTextArea		#				
SumD.R0C1 UpdateTextArea	1	0	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
				done		ButtonPressed		-1				
				done		ButtonPressed		-1				
SumD.R0C1 UpdateTextArea	1	1	INCORRECT	done		ButtonPressed		-1				
HeaderFile.R0C0 UpdateTextArea	1	1	CORRECT	HeaderFile.R0C0		UpdateTextArea		#				
SumD.R0C1 UpdateTextArea	1	1	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
SumD.R0C1 UpdateTextArea	1	1	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
HeaderFile.R0C0 UpdateTextArea	1	0	CORRECT	HeaderFile.R0C0		UpdateTextArea		#				
SumD.R0C1 UpdateTextArea	1	0	HINT	hint		ButtonPressed		hint request	Start with the column on the right. This is the ones column.		1	1
HeaderFile.R0C1 UpdateTextArea	1	0	CORRECT	HeaderFile.R0C1		UpdateTextArea		include				
SumD.R0C1 UpdateTextArea	2	0	INCORRECT	done		ButtonPressed		-1				
SumD.R0C1 UpdateTextArea	3	0	HINT	done	hint	ButtonPressed	PreviousFocus	hint request	Start with the column on the right. This is the ones column.		1	1
dec1 UpdateTextField	1	0	INCORRECT	dec1		UpdateTextField		double []				
dec1 UpdateTextField	2	0	HINT	hint		ButtonPressed		hint request	Please declare the array in the highlighted field.		1	3
dec1 UpdateTextField	3	0	INCORRECT	dec1		UpdateTextField		double a[3]	No, this is not correct.missing terminator.			
dec1 UpdateTextField	4	0	CORRECT	dec1		UpdateTextField		double a[3];				
int1 UpdateTextField	1	0	INCORRECT	done		ButtonPressed		-1	I'm sorry, but you are not done yet. Please continue working.			
dec1 UpdateTextField	1	1	CORRECT	dec1		UpdateTextField		double a[3];				
int1 UpdateTextField	1	1	CORRECT	int1		UpdateTextField		a[0]=5.6;				
int2 UpdateTextField	1	1	CORRECT	int2		UpdateTextField		a[1]=4.5;				
int3 UpdateTextField	1	1	CORRECT	int3		UpdateTextField		a[2]=3.3;				
done ButtonPressed	1	1	CORRECT	done		ButtonPressed		-1				

Figure 10. Step-wise student observations.

4.6. Interface Model

The interface model is where the students first interact to begin learning the concepts. It also includes the interaction mirror of the domain and the pedagogical model. The student/learner interacts with the GUI of the IVL by manipulating and navigating the interface using the mouse and keyboard. The student performs the practical course task according to a given problem set, for example, the array problem set concept has the interface of the visual input table sense text fields of the header, declaration, initialization, and sum array fields available, as shown in Figure 11. These fields are perceived and traced by model tracing according to expert knowledge, and instructions, hints, and buggy messages are sent back through pedagogical model strategies to the student to allow better monitoring and inspection of conceptual understanding. This interface model also has the skill matrix tab which traces the students' knowledge in each task provided in the problem. This skill matrix operates and detects through Bayesian network probability to trace the knowledge according to event manipulation.

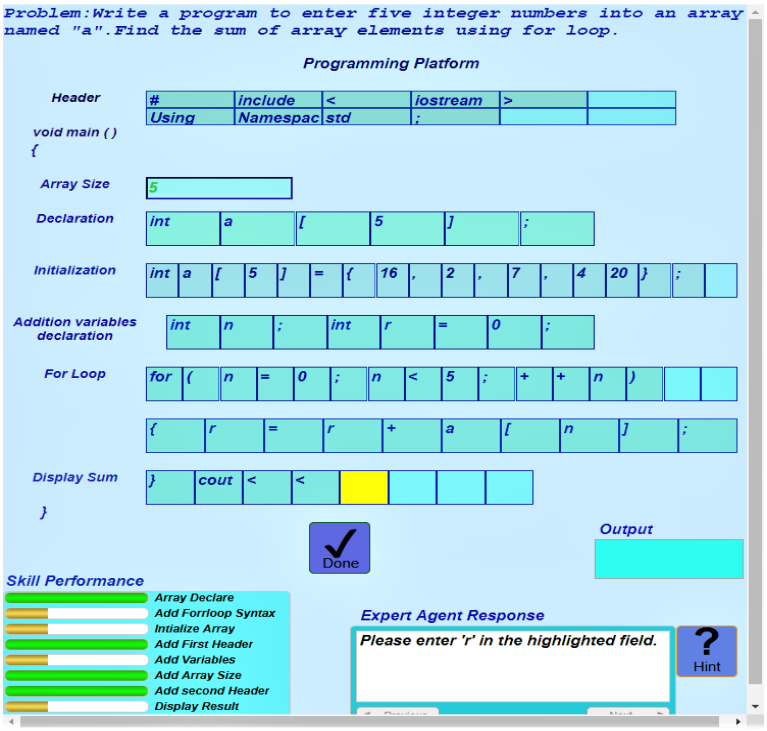

Figure 11. Learner interface for the array problem.

4.7. Reinforcement Learning

Reinforcement Learning (RL) has some basic components, such as agent actions, current states, and the new state which is received from the interface environment. The reward is based on the current action and is used to determine its state position. The strategies and agent policies are based on the current state that the agent decides to use for the next action. The value and Q-value are based on the reward policy provided to the agent [44]. We applied RL in our laboratory task scenario performed by students based on single or multiple stages. Reinforcement learning can learn student and agent behavior. The student model perceives and observes the current state of student actions by event manipulation in the interface model. The clustering of student knowledge is carried out according to the state and actions on the practical task platform. The reward is received according to the students' performance by a skills evaluation when the hints, instructions, and feedback received by pedagogical model are used. The pedagogical model agent strategies perform the actions based on two policies—the off-policy and the on-policy. The on-policy pedagogical agents learn the current policy based on the current action. SARSA (state-action-reward-state-action) [45] is used as the on-policy, whereas Q-learning [46] is used as the off-policy. Off-policy agents learn the greedy policy based on general actions and update the current state. Q-learning methods for the action selection of the agent's learning policy involve tuples such as student actions, the current state and new state of student performance, and the reward received by student evaluation through the payoff matrix, which is assigned to determine the students' skill performance. These knowledge skills are stored in clustering form, and the pedagogical model is updated to determine future actions based on the current situation of student skills.

4.8. Infrastructure of Enabling Technologies Integration

We have presented the infrastructure to integrate the enabling technology for developing the IVL, as shown in Figure 12. The IVL infrastructure uses the JADE platform to develop the multi-agent system and distributes middleware infrastructure with the FIPA specifications and Java-based framework to develop the ILC agent. It is integrated with a runtime Java platform such as Eclipse. The C++

programming concept ontology is developed through the Protégé platform. Cognitive models such as Jess rules, fact assertion, deftemplate, and student interfaces are developed through Jess programming in Eclipse Java and executed with the Jess console interpreter.

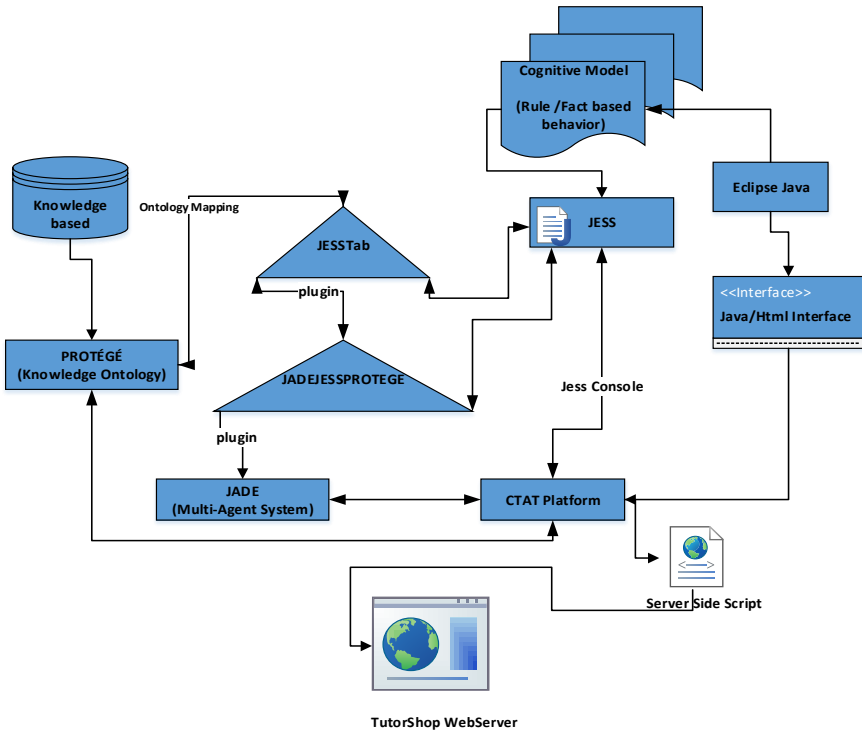

Figure 12. Infrastructure of Protégé, Jade, and Cognitive Tutor Authoring Tools (CTAT) technologies with Jess programming.

The integration of the JessTab plugin with Protégé is used to automatically create the Jess deftemplates and assertions based on OWL C++ ontology classes. The Jess language console is integrated with the JadeJessProtege plugin to connect with Jade and Protégé. The CTAT is used to compile the cognitive Jess model through collaboration of the Jess console. The cognitive Jess tutor model was tested on the TutorShop Web Server. This web server launched the student interface by assigning the enrolled student to the programming class. The teacher created the programming lab and uploaded the problem task based on multiple programming concepts. The students solve and practice the programming problem with login authorization. The teacher can view students' performance and reports.

5. Results

We developed a cognitive agent tutor based on PACA-ITS, which involved building a cognitive model for demonstration and testing the IVL artifact using CTAT. CTAT is an authoring tool that is used to demonstrate, debug, and test the cognitive based tutors [47]. The development and debugging of the cognitive based agent tutor is difficult to carry out and requires the use of AI-based language such as jess programming to provide the logic of the cognitive model [48]. We examined the array problem scenario through CTAT, which is authorized to provide the Jess tools suite for planning, demonstrating, debugging, and testing the cognitive model using the following tools: the working memory (WME) editor, behavior recorder, conflict tree, Why Not, and Jess console.

5.1. Working Memory Editor

The WME editor is used to inspect and modify the contents of the WM in a cognitive model. The array problem has the deftemplates and facts shown in Appendix B. We can also check the contents of WM before and after the model-tracing task. There are two panels in the WME editor. The array

problem in the Jess deftemplates and facts is shown in the top panel tree structure, and the bottom panel shows the selected details of these deftemplates and facts.

5.2. Conflict Tree and Why Not Window

The conflict tree is used as a debugger to predict the student selection (S), input (I), and action (A) by activating the rules fired. The conflict tree shows the rules in chain form through model tracing execution. The green icons show the correctly matching predictions between rules and student actions and the red icons show the mismatching criteria. The Why Not? Window is used to debug the Jess rules fired with the matching facts of the WM. The highlighted green and red variables indicate variables that are bound or not bound, as shown in (Figure 13).

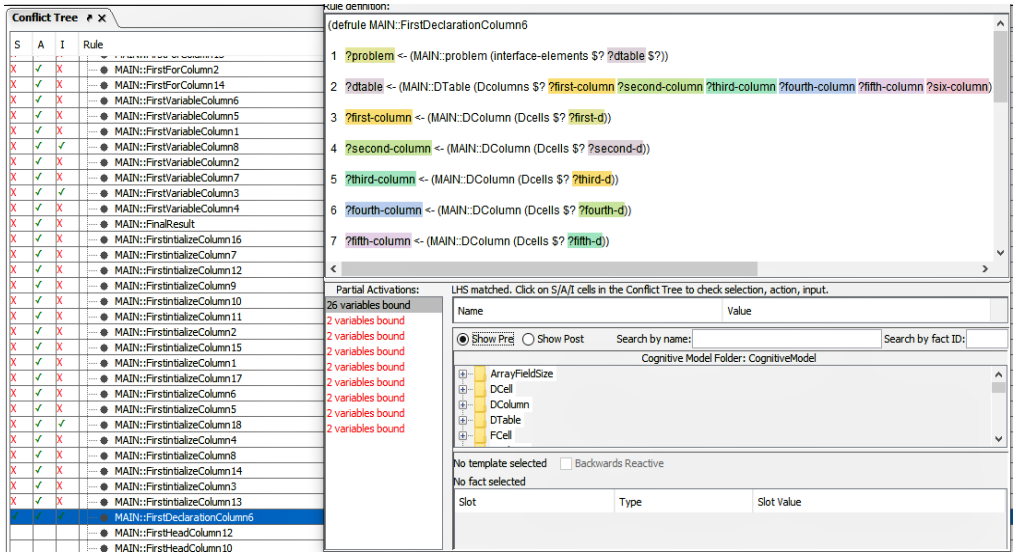

Figure 13. Conflict tree with the Jess programming compiler.

5.3. Jess Console and Behavior Recorder

The basic purpose of the behavior recorder is to record the students' states in the state machine graph, whereas rules activation and the support suite are used for planning and testing the rules and WM facts. The Jess console is a Jess interpreter command line that is used for debugging and executing the Jess queries and defrules.

5.4. Deployment through the TutorShop Webserver

After the demonstration and testing of the cognitive Jess tutor by CTAT, it is deployed on TutorShop Webserver, which was developed by Carnegie Mellon University, USA, a research center that deploys intelligent system tutors. TutorShop has collaborated with CTAT to provide course and learning content management services as a learning management system (LMS). There is a client view and a server view. These two TutorShop sites run on the tutor engine based on a script such as Java Server, JavaScript, or Java Applet. On the server side, the teacher can give and assign practical tasks to a student according to their course. They can also check student reports such as student performance, student progress reports, student skills, problem set performance, and skills. On the client side, the student can run the assigned problem set and perform practical tasks on their machine. The inner loop model tracing is performed while the student is carrying out a task, and it runs on the client side. The outer loop functionality is tested when the student completes the task, and this knowledge is stored and the student model is updated on the web machine server [49].

We deployed the programming lab on the TutorShop web server, which runs on the Java server tutor engine as shown in Appendix C. We developed multiple practical tasks based on C++

programming concepts by Jess programming and uploaded them on the assignment interface, as shown in Appendix D. Students can check and view the practical tasks on their machines by running the problem set tab shown in Appendix E. The student interface appears as shown in Figure 14. The student performs their task with the help of the cognitive Jess agent tutor which uses metacognitive variables such as hints, instructions, and correct and incorrect errors. Student performance is measured using the following metrics: when a student completes the task (date-wise); progress, measured by the fraction of the problems completed from the total problem set; and the correct step percentage, determined by the number of correct responses on the first attempt. The error percentage is counted according to the step and the number of hint requests. Hints appear according to student requests and allow corrections while performing the task. If an error still occurs, the average time spent on each problem and for the total problem set are determined, as shown in Appendix F.

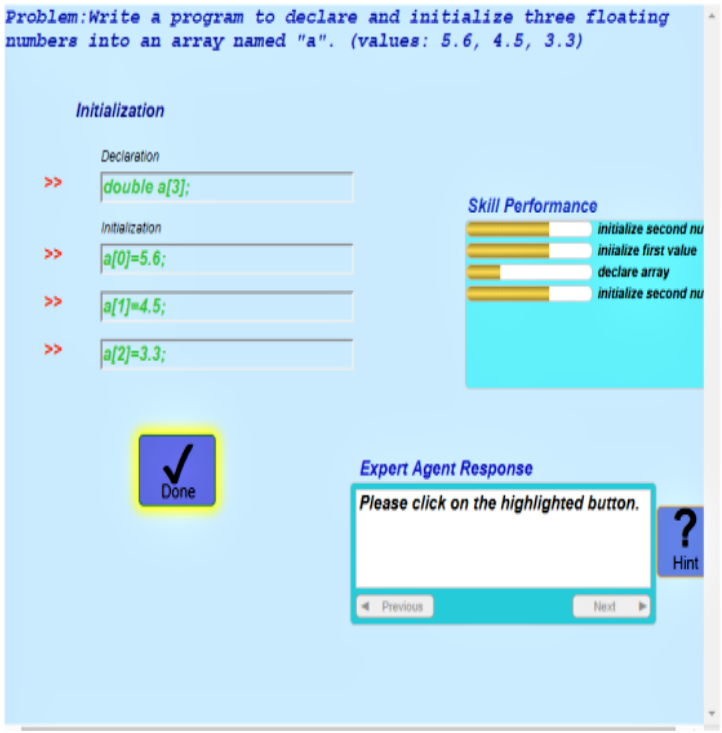

Figure 14. Array declaration and problem initialization.

6. Discussion

The results were evaluated through DataShop. DataShop is a data analysis web service that is used as a data repository for education data mining and learning science researchers. It comprises data analysis and web logging services to extract knowledge based on projects [50,51]. The programming lab project was analyzed in DataShop to produce an error report, performance profiler, and learning curve reports. The basic purpose of the quantitative analysis of the programming lab is to ensure that the student and teacher engage through timely feedback and instructions from the cognitive Jess agent tutor to determine whether the student is capable of gaining knowledge and improving their skills in practical programming concepts. The error report measures the student's first attempts at data analysis. It provides information about the student's correct and incorrect step-wise problems and the knowledge component of their skills-wise performance. It provides details of a student's actions using particular programming concepts, and feedback is received, as shown in Figure 15.

The outcome was estimated through the learning curve to assess preceding knowledge, error rate, and performance profiler to engage cognitive Jess agent efficiency, practicable, and active decisions to improve student learning by providing well-timed feedback. The student's performance was measured

by skill performance its task with the help of cognitive Jess agent tutor and metacognitive variables such as hint, instructions, correct and incorrect error. The analysis of student performance was measured with respect to the student progress by the metrics that were fraction of worked problems over problem count or worked steps over required steps, ratio of steps answered correctly on the first attempt, ratio of steps with hint requests, but no incorrect answer, ratio of steps with incorrect answers, but no hint requests, ratio of steps with incorrect answers and hint requests, average/actual time spent per problem and total time spent on problem set.

Step	Attempts			
dec1 UpdateTextField Number of Students: 13 Number of Observations: 14 Knowledge Component(s): KC161 Sample: intializeArray	Evaluation	Number of Observations	Answer	Feedback/Classification
	correct	10 (71.43%)	double a[3];	
	hint	1 (7.14%)	hint request	Please declare the array in the highlighted field.
	incorrect	1 (7.14%)	do	
	incorrect	1 (7.14%)	double []	
	incorrect	1 (7.14%)	double a[2];	No, this is not correct size.
done ButtonPressed Number of Students: 13 Number of Observations: 13 Knowledge Component(s): KC113 Sample: intializeArray	Evaluation	Number of Observations	Answer	Feedback/Classification
	correct	12 (92.31%)	-1	
	hint	1 (7.69%)	hint request	Please click on the highlighted button.
int1 UpdateTextField Number of Students: 13 Number of Observations: 14 Knowledge Component(s): KC128 Sample: intializeArray	Evaluation	Number of Observations	Answer	Feedback/Classification
	correct	9 (64.29%)	a[0]=5.6;	
	hint	2 (14.29%)	hint request	Please initialize first array in the highlighted field.
	incorrect	1 (7.14%)	-1	I'm sorry, but you are not done yet. Please continue working.
	incorrect	1 (7.14%)	a[0] =5.6;	
	incorrect	1 (7.14%)	a[0]=	
int2 UpdateTextField Number of Students: 13 Number of Observations: 13 Knowledge Component(s): KC39 Sample: intializeArray	Evaluation	Number of Observations	Answer	Feedback/Classification
	correct	12 (92.31%)	a[1]=4.5;	
	hint	1 (7.69%)	hint request	Please initialize in the highlighted field.
int3 UpdateTextField Number of Students: 13 Number of Observations: 13 Knowledge Component(s): KC151 Sample: intializeArray	Evaluation	Number of Observations	Answer	Feedback/Classification
	correct	11 (84.62%)	a[2]=3.3;	
	hint	1 (7.69%)	hint request	Please enter 'a[2]=3.3;' in the highlighted field.
	incorrect	1 (7.69%)	a	

Figure 15. Error analysis of the performance on the initial array problem.

The student learning curve measure [52] is analyzed and monitored by a cognitive Jess tutor. This quantifies how accurately a student performs a particular task and whether they are able to learn quickly. A student is shown to produce fewer errors during the evaluation of their actions in the array program; the error rate is mapped along the y-axis, and the opportunity, which is provided by a hint or instructions by an expert agent over time, is shown on the x-axis in the line graph shown in Figure 16. The prediction of student performance was done using the performance profiler which assessed the error rate versus the knowledge components (KC) provided to students in a particular problem. The initial array problem set includes four types of KC, which are used to evaluate the skills of the student, as shown on the y-axis, and the error rate over the correct and incorrect answers and hints provided is also shown in Figure 17. The outcome showed that IVL has an encompassing model for improving the pupil's learning and recommendations for auxiliary research application.

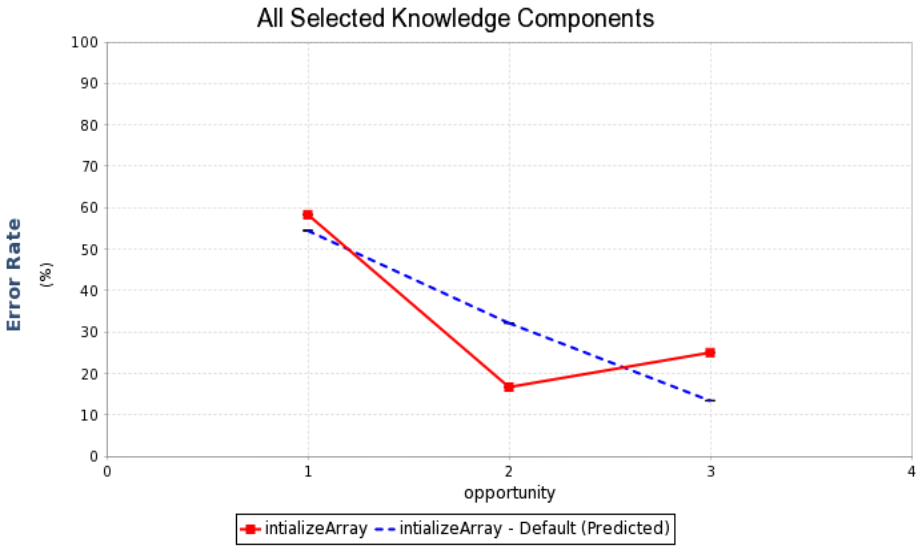

Figure 16. Deviations of the student performance learning curve over time.

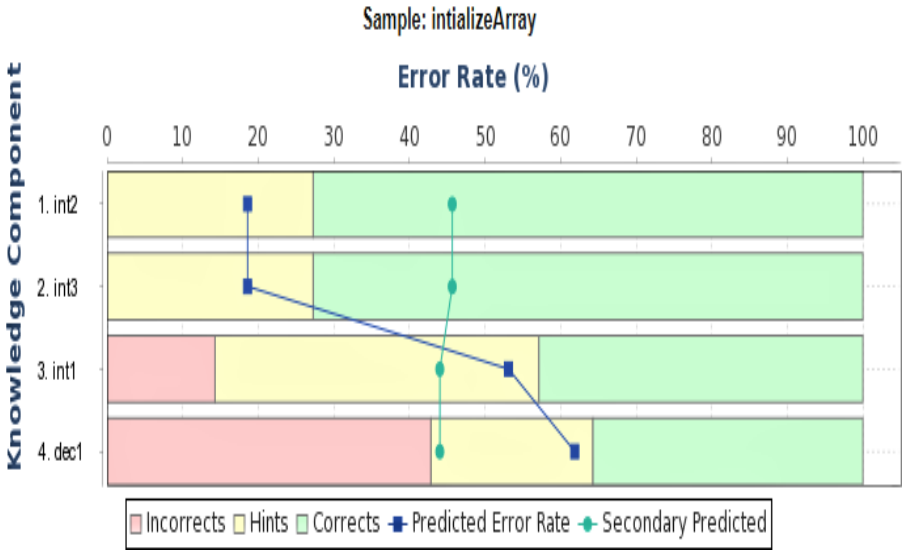

Figure 17. Deviations of the performance profiler showing the error rate over the knowledge Components.

7. Conclusions

In this research, we designed and developed a multi-agent system in the context of IVL through AUML modelling language and the JADE platform. We presented the ILC Java-based agent operations, which interact with each other, and we also showed these agents' behaviors. The prototype implementation infrastructure for working on the proposed PACA-ITS framework according to the ITS components was also presented. The integration of enabling technology enhanced the capability and flexibility of the system through collaboration with JADE for agent development, Protégé for designing the C++ ontology, and CTAT to successfully evaluate and test the system with the Jess cognitive tutor. After the quantitative analysis of the programming lab, student and teacher engagement can occur through timely feedback and instructions from the cognitive Jess agent tutor to show whether the student is capable of gaining knowledge and improving skills in programming concepts. The cognitive model algorithm was designed with Jess programming to measure the student performance through learning curve metrics that assess preceding knowledge, error rate metrics, and performance profiler metrics, which show the engagement of the student and cognitive Jess agent tutor. The cognitive Jess agent tutor performs efficiently in a practical way and makes active decisions to

improve and enhance student learning regarding the C++ programming concept. In the future, we will enhance the computational costs and time efficiency of the cognitive Jess tutor. We plan to create more problems beyond these given array problems and to put the tutor in front of more students. We will also implement the knowledge tracing by recording student observations through the deep learning neural network functionality in the student model and enhance the pedagogical model strategies by updating and learning.

Author Contributions: S.M. contributed to the idea of this paper, designed the algorithms and written the original draft. S.M., was responsible for performing the experiments and the analysis. S.K.T., M.A. and E.A. was in charge of supervision.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

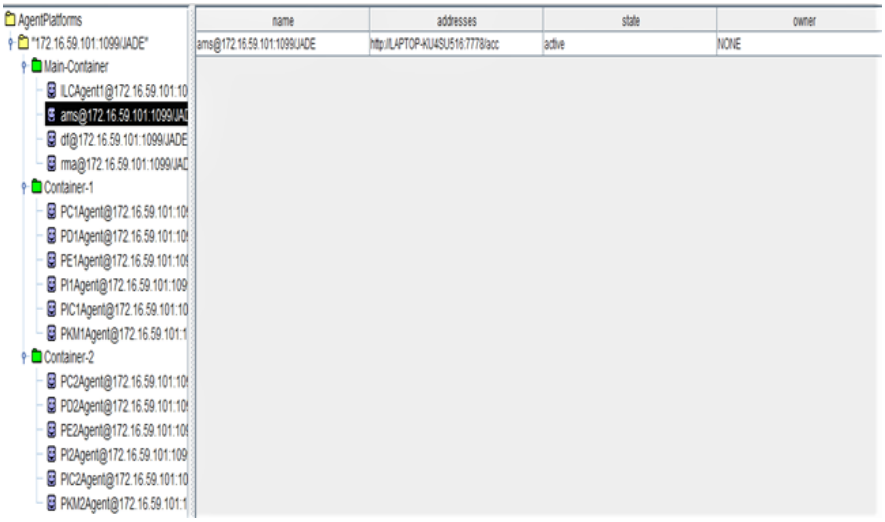

Figure A1. ILC agent's containers on the JADE remote agent management platform.

Appendix B

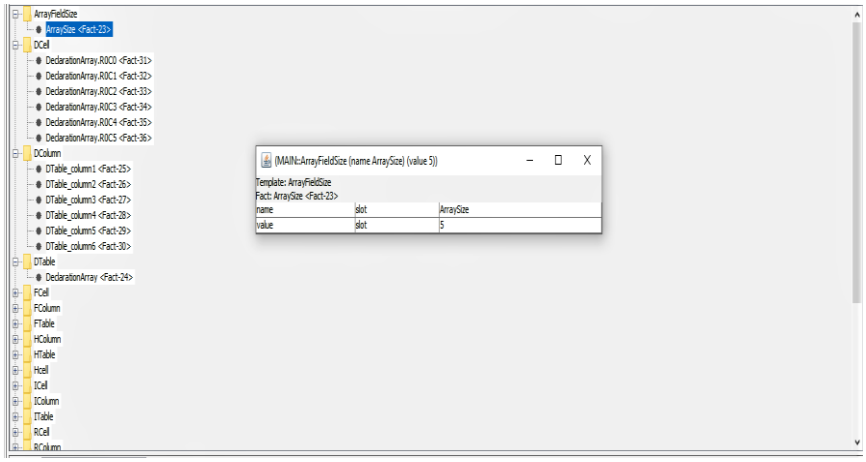

Figure A2. Array problem facts compiled in the working memory editor (WME).

Appendix C

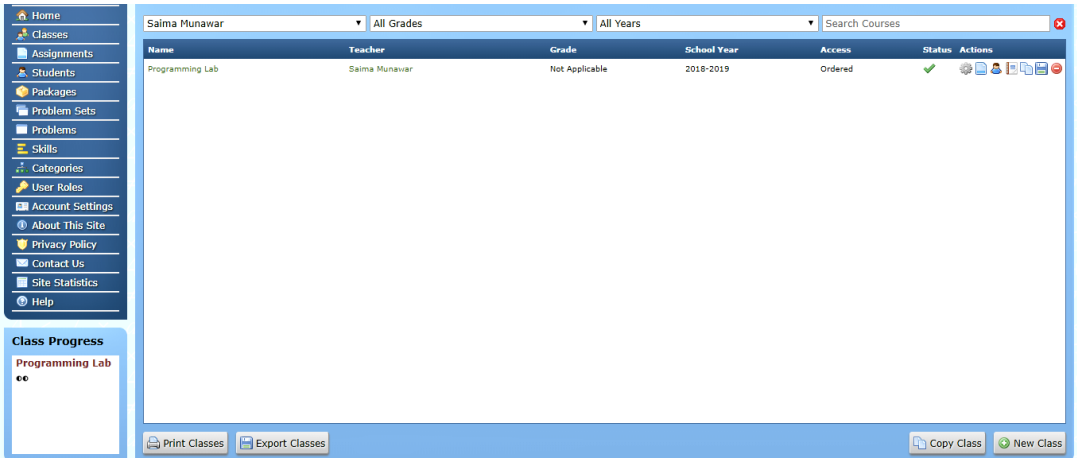

Figure A3. Teacher's view deployed on the TutorShop Web Server.

Appendix D

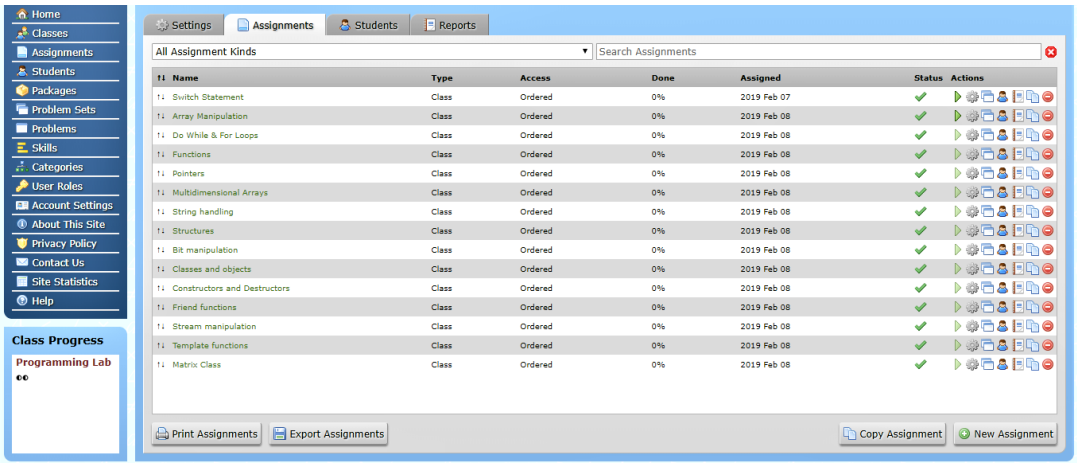

Figure A4. C++ concepts for the programming lab deployed on the TutorShop Web Server.

Appendix E

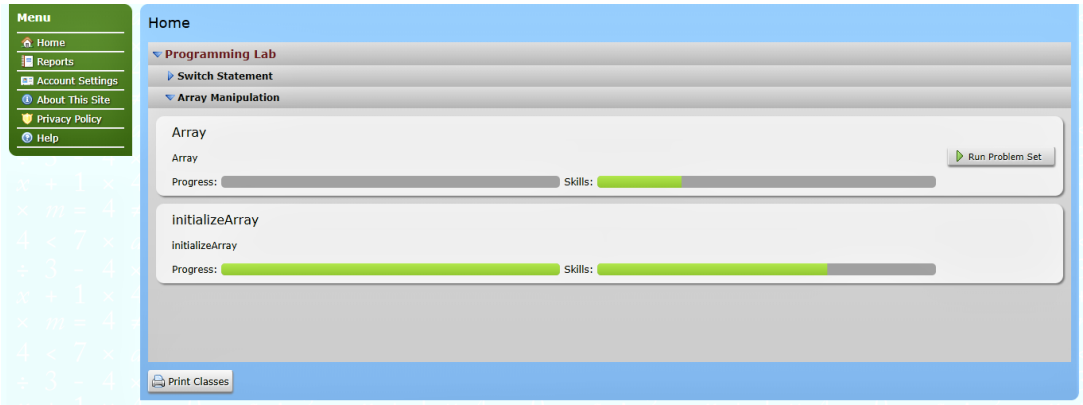

Figure A5. C++ programming lab in student view.

Appendix F

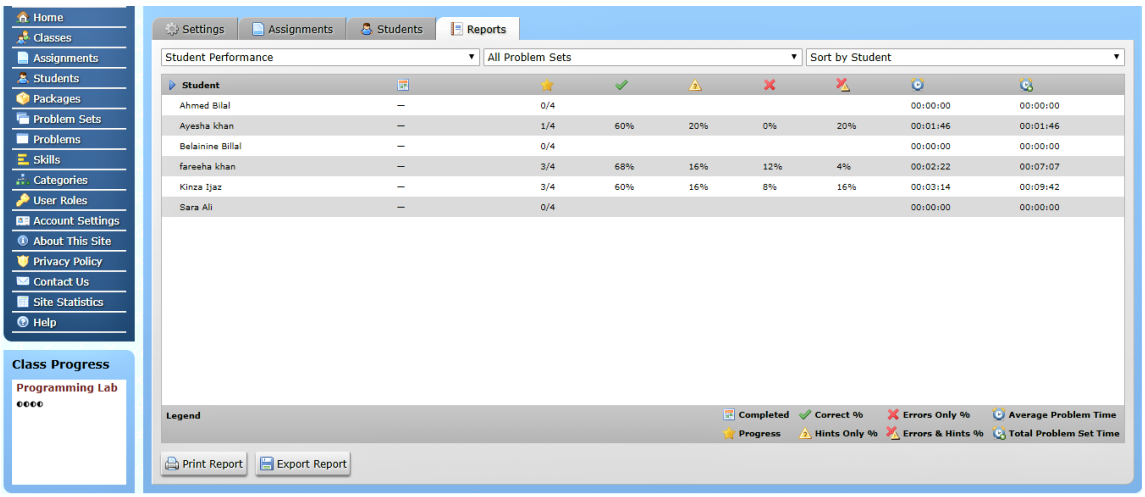

Figure A6. Analysis of a student's performance.

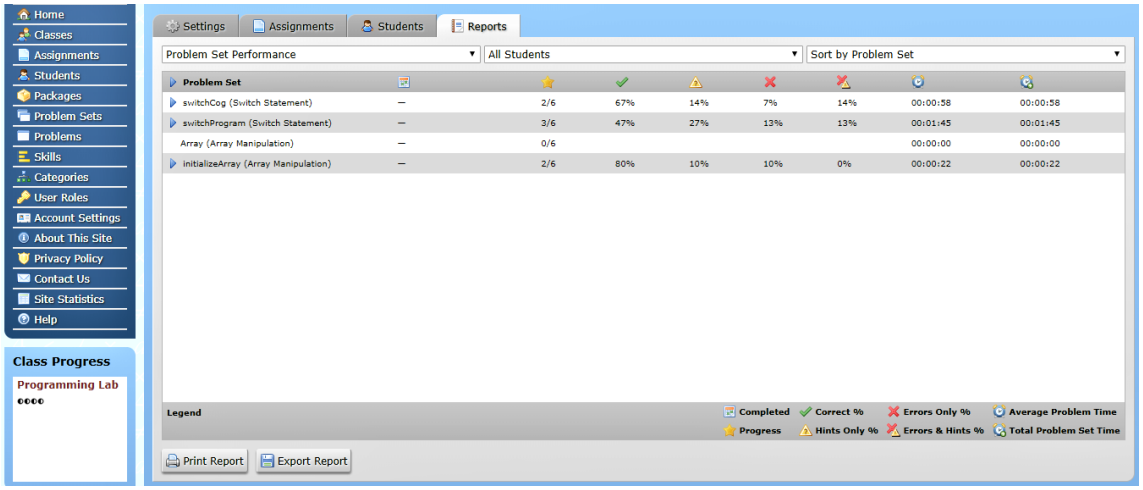

Figure A7. Analysis of a problem set's performance.

Appendix G

Table A1. Overview of selected primary studies of VL.

Sr.No	Architecture Design	Platform Type	Year	Reference
1	Virtual Laboratory	Multi-agent System	2007	[53]
2	Virtual Laboratory Platform	Virtual Simulation Experiment	2008	[54]
3	Online Virtual Laboratory (CyberLab)	Toolkit	2008	[55]
4	Virtual Laboratory Platform (VL-JM)	VL platform	2008	[56]
5	Web-based Interactive Learning	Web Technologies	2008	[57]
6	Online Internet Laboratory (ONLINE I-LAB)	Internet Assisted Laboratory	2008	[58]
7	Distributed Virtual Laboratory(DVL)	Architecture	2009	[59]
8	VLs and the Synchronous Collaborative Learning Practice	Web-Learning System Tool	2009	[60]

Table A1. *Cont.*

Sr.No	Architecture Design	Platform Type	Year	Reference
9	Virtual Laboratory Environment (VLE)	Software Tool	2009	[61]
10	Multi-agent Architecture	Experiment	2010	[62]
11	Architecture of the Remote Laboratory(LRA-ULE)	Educational Tool	2010	[63]
12	Virtual Laboratory for Robotics (VLR)	Virtual Tool	2010	[64]
13	Virtual Experiment	Multi-agent	2010	[65]
14	LAB Teaching Assistant (LABTA)	Agent-based	2010	[66]
15	Cyberinfrastructure(VLab)	Experiment and GUI Tool	2011	[67]
16	Artificial and Real Laboratory	Virtual Tool	2011	[68]
17	Virtual Laboratory Architecture	Architecture	2011	[68]
18	Web-Based Virtual Machine Laboratory	Semantic Web	2011	[69]
19	Virtual Computational Laboratory	Architecture	2011	[70]
20	Virtualized Infrastructure	Experiment	2012	[71]
21	Learning Environment (NoobLab)	Automated Assessment Tool	2012	[72]
22	Virtualized Infrastructure(V-Lab)	Experiment and GUI Tool	2012	[73]
23	Automatic Grading Platform.	Automated Assessment Tool	2012	[74]
24	Real Time Virtual Laboratory	Multi-agent System	2012	[75]
25	Virtual Embedded System Laboratory	Assessment Tool	2012	[76]
26	Virtual and Remote Laboratories	Automation Tool	2012	[77]
27	Educational Robotics Laboratories	Experiential Laboratories Tool	2013	[78]
28	VCL Virtual Laboratory	Virtual Tool	2013	[79]
29	Simulation Environment for a virtual Laboratory	Simulation Tool	2014	[80]
30	Electronic Circuit Virtual Laboratory	System Architecture	2014	[81]
31	Collaborative Virtual Laboratory (VLAB-C)	Cloud based Architecture	2014	[82]
32	Cloud-Based Virtual Laboratory (V-Lab)	Experiment	2014	[83]
33	Software Virtual and Robotics tool	Experiment and GUI Tool	2015	[84]
34	IUVIRLAB	Virtual Tool	2015	[85]
35	Virtual Laboratory	Experiment	2015	[86]
36	Virtual Laboratory Platform	System Structure	2015	[87]
37	Virtual Laboratory Platform	Experiment and GUI Tool	2015	[88]
38	Virtual Laboratory for Computer Organization and Logic Design (COLDVL)	Experiment and GUI tool	2015	[89]
39	Virtual Laboratories	Automata Model	2015	[90]
40	Cloud-Based Integrated Virtual Laboratories (CIVIL Model)	Model	2016	[91]
41	Virtual and Remote Labs	Analysis	2016	[92]
42	Virtual Laboratories for Education in Science, Technology, and Engineering	Analysis	2016	[93]
43	Virtual Laboratory Technology	Comparative Study Analysis	2016	[94]
44	Virtual Laboratory	Multi-agent System	2016	[95]
45	Virtual Laboratory of a Spider Crane	Simulink Model	2016	[96]
46	Virtual Laboratory Platform	Experiment and GUI Tool	2016	[97]

Table A1. *Cont.*

Sr.No	Architecture Design	Platform Type	Year	Reference
47	Virtual Laboratory HTML5	Experiment and GUI Tool	2016	[98]
48	Collaborative Virtual Laboratory (VLAB-C)	Cloud Infrastructure	2016	[99]
49	Game Technologies VL	Platform	2016	[100]
50	Virtual Laboratory (VLs)	Tool	2017	[101]
51	CodeLab	Conversation-Based Educational Tool	2018	[102]
52	Web-based Remote FPGA Laboratory	web-based platform	2019	[103,104]
53	Network virtual lab	Framework of Virtual Laboratory System	2019	[104]
54	RoboSim	Robotics simulation for programming virtual Linkbot	2019	[105]
55	ARCat	programming tool	2019	[106]
56	Virtual/Remote Labs	Virtual Reality Machines	2019	[107]
57	REMOTE LAB	fully open integrated system (FOIS)	2019	[108]

References

1. Ozana, S.; Docekal, T. The concept of virtual laboratory and PIL modeling with REX control system. In Proceedings of the 2017 21st International Conference on Process Control (PC), Štrbské Pleso, Slovakia, 6–9 June 2017; pp. 98–103.
2. Broisin, J.; Venant, R.; Vidal, P. Lab4CE: A remote laboratory for computer education. *Int. J. Artif. Intell. Educ.* **2017**, *27*, 154–180. [CrossRef]
3. Diwakar, S.; Kumar, D.; Radhamani, R.; Sasidharakurup, H.; Nizar, N.; Achuthan, K.; Nedungadi, P.; Raman, R.; Nair, B.G. Complementing education via virtual labs: Implementation and deployment of remote laboratories and usage analysis in south indian villages. *Int. J. Online Eng.* **2016**, *12*, 8–15. [CrossRef]
4. Ahmed, S.; Karsiti, M.N. *Loh, R.N. Control Analysis and Feedback Techniques for Multi Agent Robots*; IntechOpen: London, UK, 2009; p. 426. ISBN 978-3-902613-51-6.
5. Russell, S.J.; Norvig, P.; Canny, J.F.; Malik, J.M.; Edwards, D.D. *Artificial Intelligence: A Modern Approach vol. 2: Prentice Hall Upper Saddle River*; Pearson Education, Inc.: New Jersey, NJ, USA, 2003.
6. Franklin, S.; Graesser, A. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *International Workshop on Agent Theories, Architectures, and Languages*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 21–35.
7. Wooldridge, M. *An Introduction to Multiagent Systems*; John Wiley & Sons: New York, NY, USA, 2009.
8. Odell, J.; Parunak, H.V.D.; Bauer, B. Extending UML for agents. *Ann Arbor* **2000**, *1001*, 48103.
9. Abushark, Y.; Thangarajah, J. AUML protocols: From specification to detailed design. In Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems, Saint Paul, MN, USA, 6–10 May 2013; pp. 1173–1174.
10. Bauer, B.; Müller, J.P.; Odell, J. Agent UML: A formalism for specifying multiagent software systems. *Int. J. Softw. Eng. Knowl. Eng.* **2001**, *11*, 207–230. [CrossRef]
11. Da Silva, V.T.; Choren, R.; de Lucena, C.J. A UML based approach for modeling and implementing multi-agent systems. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, 19–23 July 2004; Volume 2, pp. 914–921.
12. Bellifemine, F.L.; Caire, G.; Greenwood, D. *Developing Multi-Agent Systems with JADE*; John Wiley & Sons: Hoboken, NJ, USA, 2007; Volume 7.
13. Magid, N.; Giovanni, C.; Parisa, A.B. A Methodology for the Analysis and Design of Multi-Agent Systems using JADE. *Comput. Syst. Sci. Eng.* **2006**, *21*.

14. Da Silva, F.L.; Costa, A.H.R. Automatic Object-Oriented Curriculum Generation for Reinforcement Learning. In Proceedings of the 1st Workshop on Scaling-Up Reinforcement Learning (SURL), Skopje, North Macedonia, 18 September 2017; Available online: http://surl.tirl.info/proceedings/SURL-2017_paper_1.pdf (accessed on 1 October 2018).
15. Amir, O.; Doshi-Velez, F.; Sarne, D. Agent Strategy Summarization. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 1203–1207.
16. Meir, R.; Parkes, D. Playing the Wrong Game: Bounding Externalities in Diverse Populations of Agents. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 86–94.
17. Wu, J.; Ghosh, S.; Chollet, M.; Ly, S.; Mozgai, S.; Scherer, S. NADiA-Towards Neural Network Driven Virtual Human Conversation Agents. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 2262–2264.
18. Icarte, R.T.; Klassen, T.Q.; Valenzano, R.; McIlraith, S.A. Teaching multiple tasks to an RL agent using LTL. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 452–461.
19. Grover, A.; Al-Shedivat, M.; Gupta, J.K.; Burda, Y.; Edwards, H. Evaluating Generalization in Multiagent Systems using Agent-Interaction Graphs. In Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018.
20. Track, S.I.A.; Pöppel, J.; Kopp, S. Satisficing Models of Bayesian Theory of Mind for Explaining Behavior of Differently Uncertain Agents. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 10–15 July 2018.
21. Savarimuthu, B.T.R. *Norm Learning in Multi-Agent Societies*; University of Otago: Dunedin, New Zealand, 2011.
22. Sen, S. The Effects of Past Experience on Trust in Repeated Human-Agent Teamwork. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 514–522.
23. Metcalf, K.; Theobald, B.-J.; Apostoloff, N. Learning Sharing Behaviors with Arbitrary Numbers of Agents. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, 10–15 July 2018; pp. 1232–1240.
24. Nkambou, R.; Bourdeau, J.; Mizoguchi, R. Introduction: What are intelligent tutoring systems, and why this book? In *Advances in Intelligent Tutoring Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 308, pp. 1–12.
25. Nkambou, R. Modeling the domain: An introduction to the expert module. In *Advances in Intelligent Tutoring Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 308, pp. 15–32.
26. Mizoguchi, R.; Hayashi, Y.; Bourdeau, J. Ontology-based formal modeling of the pedagogical world: Tutor modeling. In *Advances in Intelligent Tutoring Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 308, pp. 229–247.
27. Woolf, B.P. Student modeling. In *Advances in Intelligent Tutoring Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 308, pp. 267–279.
28. Psyché, R.N.V. *The Role of Ontologies in the Authoring of ITS: An Instructional Design Support Agent*. 2008. Available online: https://www.researchgate.net/publication/327253898_The_Role_of_Ontologies_in_the_Authoring_of_ITS_an_Instructional_Design_Support_Agent (accessed on 1 October 2018).
29. Forgy, C.L. Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Readings in Artificial Intelligence and Databases*; Elsevier: San Mateo, CA, USA, 1989; pp. 547–559.
30. Harry, E.; Edward, B. Making real virtual lab. *Sci. Educ. Rev.* **2005**, *4*, 2005.
31. Scanlon, E.; Morris, E.; di Paolo, T.; Cooper, M. Contemporary Approaches to Learning Science: Technologically-Mediated Practical Work. *Stud. Sci. Educ.* **2002**, *38*, 73–114. [CrossRef]
32. Lewis, D.I. The pedagogical benefits and pitfalls of virtual tools for teaching and learning laboratory practices in the biological sciences. *High. Educ. Acad. STEM York UK* **2014**.
33. Miyamoto, M.; Milkowski, D.M.; Young, C.D.; Lebowicz, L.A. Developing a Virtual Lab to Teach Essential Biology Laboratory Techniques. *J. Biocommun.* **2019**, *43*, 23–31.
34. Heradio, R.; de la Torre, L.; Dormido, S. Virtual and remote labs in control education: A survey. *Annu. Rev. Control* **2016**, *42*, 1–10. [CrossRef]

35. Richter, T.; Tetour, Y.; Boehringer, D. Library of labs-a european project on the dissemination of remote experiments and virtual laboratories. In Proceedings of the 2011 IEEE International Symposium on Multimedia (ISM), Dana Point, CA, USA, 5–11 December 2011; pp. 543–548.
36. Lowe, D.; Conlon, S.; Murray, S.; Weber, L.; de la Villefromoy, M.; Lindsay, E. Labshare: Towards cross-institutional. In *Internet Accessible Remote Laboratories: Scalable E-Learning Tools for Engineering and Science Disciplines: Scalable E-Learning Tools for Engineering and Science Disciplines*; IGI Global: Hershey, PA, USA, 2011; Volume 453.
37. García-Zubia, J.; Angulo, I.; Hernández, U.; Orduña, P. Plug&Play remote lab for microcontrollers: WebLab-DEUSTO-PIC. In Proceedings of the 7th European Workshop on Microelectronics Education, San Jose, CA, USA, 28–30 May 2008; pp. 28–30.
38. De Jong, T.; Sotiriou, S.; Gillet, D. Innovations in STEM education: The Go-Lab federation of online labs. *Smart Learn. Environ.* **2014**, *1*, 3. [CrossRef]
39. Lindsay, E.; Murray, S.; Stumpers, B.D. A toolkit for remote laboratory design & development. In Proceedings of the 2011 First Global Online Laboratory Consortium Remote Laboratories Workshop (GOLC), Cyberjaya, Malaysia, 19–20 December 2011; pp. 1–7.
40. Kaplan, J.; Yankelovich, N. Open wonderland: An extensible virtual world architecture. *IEEE Internet Comput.* **2011**, *15*, 38–45. [CrossRef]
41. Achuthan, K.; Sreelatha, K.; Surendran, S.; Diwakar, S.; Nedungadi, P.; Humphreys, S.; Sreekala, S.C.O.; Pillai, Z.; Raman, R.; Deepthi, A.; et al. The VALUE@ Amrita Virtual Labs Project: Using web technology to provide virtual laboratory access to students. In Proceedings of the 2011 IEEE Global Humanitarian Technology Conference (GHTC), Seattle, WA, USA, 30 October–1 November 2011; pp. 117–121.
42. Rus, V.; Ştefănescu, D. Non-intrusive assessment of learners' prior knowledge in dialogue-based intelligent tutoring systems. *Smart Learn. Environ.* **2016**, *3*, 2. [CrossRef]
43. JAVA Agent DEvelopment Framework. 2018. Available online: http://jade.tilab.com/ (accessed on 1 October 2018).
44. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, UK, 1998; Volume 135.
45. Dorça, F.A.; Lima, L.V.; Fernandes, M.A.; Lopes, C.R. Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis. *Expert Syst. Appl.* **2013**, *40*, 2092–2101. [CrossRef]
46. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
47. Aleven, V.; McLaren, B.M.; Sewall, J.; Koedinger, K.R. The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains. In Proceedings of the International Conference on Intelligent Tutoring Systems, Jhongli, Taiwan, 26–30 June 2006; pp. 61–70.
48. Koedinger, K.R.; Anderson, J.R.; Hadley, W.H.; Mark, M.A. Intelligent tutoring goes to school in the big city. *Int. J. Artif. Intell. Educ.* **1997**, *8*, 30–43.
49. Aleven, V.; Sewall, J.; Popescu, O.; Xhakaj, F.; Chand, D.; Baker, R.; Wang, Y.; Siemens, G.; Rosé, C.; Gasevic, D. The beginning of a beautiful friendship? Intelligent tutoring systems and MOOCs. In Proceedings of the International Conference on Artificial Intelligence in Education, Madrid, Spain, 22–26 June 2015; pp. 525–528.
50. Aravind, V.R.; Refugio, C. Efficient learning with intelligent tutoring across cultures. *World J. Educ. Technol. Curr. Issues* **2019**.
51. Koedinger, K.R.; Baker, R.S.; Cunningham, K.; Skogsholm, A.; Leber, B.; Stamper, J. A data repository for the EDM community: The PSLC DataShop. In *Handbook of Educational Data Mining*; CRC Press: Boca Raton, FL, USA, 2010; Volume 43, pp. 43–56.
52. Jaber, M.Y.; Glock, C.H. A learning curve for tasks with cognitive and motor elements. *Comput. Ind. Eng.* **2013**, *64*, 866–871. [CrossRef]
53. Mechta, D.; Harous, S.; Djoudi, M.; Douar, A. An Agent-based approach for designing and implementing a virtual laboratory. In Proceedings of the IIT'07 4th International Conference on Innovations in Information Technology, Dubai, United Arab Emirates, 18–20 November 2007; pp. 496–500.
54. Xie, W.; Yang, X.; Li, F. A virtual laboratory platform and simulation software based on web. In Proceedings of the ICARCV 2008 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 2–5 December 2008; pp. 1650–1654.
55. Zhao, K.; Evett, M.P. CyberLab: An Online Virtual Laboratory Toolkit for Non-programmers. In Proceedings of the ICALT'08. Eighth IEEE International Conference on Advanced Learning Technologies, Cantabria, Spain, 1–5 July 2008; pp. 316–318.

56. Sheng, Y.; Wang, W.; Wang, J.; Chen, J. A virtual laboratory platform based on integration of java and matlab. In Proceedings of the International Conference on Web-Based Learning, Magdeburg, Germany, 23–25 September 2008; pp. 285–295.
57. Moon, I.; Han, S.; Choi, K.; Kim, D.; Jeon, C.; Lee, S. Virtual education system for the c programming language. In Proceedings of the International Conference on Web-Based Learning, Jinhua, China, 20–22 August 2008; pp. 196–207.
58. Salihbegovic, A.; Ribic, S. Development of online internet laboratory (online I-lab). In *Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education*; Springer: Dordrecht, The Netherlands, 2008; pp. 1–6.
59. Grimaldi, D.; Rapuano, S. Hardware and software to design virtual laboratory for education in instrumentation and measurement. *Measurement* **2009**, *42*, 485–493. [CrossRef]
60. Jara, C.A.; Candelas, F.A.; Torres, F.; Dormido, S.; Esquembre, F.; Reinoso, O. Real-time collaboration of virtual laboratories through the Internet. *Comput. Educ.* **2009**, *52*, 126–140. [CrossRef]
61. Quesnel, G.; Duboz, R.; Ramat, É. The Virtual Laboratory Environment–An operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simul. Model. Pract. Theory* **2009**, *17*, 641–653. [CrossRef]
62. Combes, M.; Buin, B.; Parenthoën, M.; Tisseau, J. Multiscale multiagent architecture validation by virtual instruments in molecular dynamics experiments. *Procedia Comput. Sci.* **2010**, *1*, 761–770. [CrossRef]
63. Domínguez, M.; Reguera, P.; Fuertes, J.; Prada, M.; Alonso, S.; Morán, A.; Fernández, D. The virtual laboratory on cybernetics illustrates main results of the paper. Development of an educational tool in LabVIEW and its integration in remote laboratory of automatic control. *IFAC Proc. Vol.* **2010**, *42*, 101–106. [CrossRef]
64. Potkonjak, V.; Vukobratović, M.; Jovanović, K.; Medenica, M. Virtual Mechatronic/Robotic laboratory—A step further in distance learning. *Comput. Educ.* **2010**, *55*, 465–475. [CrossRef]
65. Li, X.; Ma, F.; Zhong, S.; Tang, L.; Han, Z. Research on virtual experiment intelligent tutoring system based on multi-agent. In Proceedings of the International Conference on Technologies for E-Learning and Digital Entertainment, Changchun, China, 16–18 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6249, pp. 100–110. [CrossRef]
66. Yang, C. LABTA: An Agent-Based Intelligent Teaching Assistant for Experiment Courses. In Proceedings of the International Conference on Web-Based Learning, Shanghai, China, 8–10 December 2010; pp. 309–317.
67. Da Silveira, P.R.; Valdez, M.N.; Wenzcovitch, R.M.; Pierce, M.; da Silva, C.R.; Yuen, D.A. Virtual laboratory for planetary materials (VLab): An updated overview of system service architecture. In Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, Salt Lake City, UT, USA, 18–21 July 2011; p. 33.
68. Logar, V.; Karba, R.; Papič, M.; Atanasijević-Kunc, M. Artificial and real laboratory environment in an e-learning competition. *Math. Comput. Simul.* **2011**, *82*, 517–524. [CrossRef]
69. Salonen, J.; Nykänen, O.; Ranta, P.; Nurmi, J.; Helminen, M.; Rokala, M. An implementation of a semantic, web-based virtual machine laboratory prototyping environment. In Proceedings of the International Semantic Web Conference, Bonn, Germany, 23–27 October 2011; pp. 221–236.
70. Juszczyszyn, K.; Paprocki, M.; Prusiewicz, A.; Sieniawski, L. Personalization and content awareness in online lab–virtual computational laboratory. In Proceedings of the Asian Conference on Intelligent Information and Database Systems, Daegu, Korea, 20–22 April 2011; pp. 367–376.
71. Uludag, S.; Guler, E.; Karakus, M.; Turner, S.W. An affordable virtual laboratory infrastructure to complement a variety of computing classes. *J. Comput. Sci. Coll.* **2012**, *27*, 158–166.
72. Neve, P.; Hunter, G.; Livingston, D.; Orwell, J. NoobLab: An intelligent learning environment for teaching programming. In Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, Washington, DC, USA, 4–7 December 2012; Volume 3, pp. 357–361.
73. Xu, L.; Huang, D.; Tsai, W.-T. V-lab: A cloud-based virtual laboratory platform for hands-on networking courses. In Proceedings of the 17th ACM Annual Conference on INNOVATION and Technology in Computer Science Education, Larnaca, Cyprus, 2–4 July 2012; pp. 256–261.
74. Sánchez, C.; Gómez-Estern, F.; de la Peña, D.M. A virtual lab with automatic assessment for nonlinear controller design exercises. *IFAC Proc. Vol.* **2012**, *45*, 172–176. [CrossRef]

75. Al-hamdani, A.Y.H.; Altaie, A.M. Designing and implementation of a real time virtual laboratory based on multi-agents system. In Proceedings of the 2012 IEEE Conference on Open Systems (ICOS), Kuala Lumpur, Malaysia, 21–24 October 2012; pp. 1–6.
76. Mohanty, R.; Routray, A. Advanced virtual embedded system laboratory. In Proceedings of the 2012 2nd Interdisciplinary Engineering Design Education Conference (IEDEC), Santa Clara, CA, USA, 19–20 March 2012; pp. 92–95.
77. Leão, C.P.; Soares, F.; Rodrigues, H.; Seabra, E.; Machado, J.; Farinha, P.; Costa, S. Web-assisted laboratory for control education: Remote and virtual environments. In *The Impact of Virtual, Remote, and Real Logistics Labs*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 62–72.
78. Caci, B.; Chiazzese, G.; D'Amico, A. Robotic and virtual world programming labs to stimulate reasoning and visual-spatial abilities. *Procedia-Soc. Behav. Sci.* **2013**, *93*, 1493–1497. [CrossRef]
79. Jordá, J.M.M. Virtual Tools: Virtual Laboratories for Experimental science—An Experience with VCL Tool. *Procedia-Soc. Behav. Sci.* **2013**, *106*, 3355–3365. [CrossRef]
80. Dias, F.; Matutino, P.M.; Barata, M. Virtual laboratory for educational environments. In Proceedings of the 2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV), Porto, Portugal, 26–28 February 2014; pp. 191–194.
81. Yang-Mei, L.; Bo, C. Electronic Circuit Virtual Laboratory Based on LabVIEW and Multisim. In Proceedings of the 2014 7th International Conference on Intelligent Computation Technology and Automation (ICICTA), Changsha, China, 25–26 October 2014; pp. 222–225.
82. Yu, J.; Dong, K. VLAB-C: A cloud service platform for collaborative virtual laboratory. In Proceedings of the 2014 IEEE International Conference on Services Computing (SCC), Anchorage, AK, USA, 27 June–2 July 2014; pp. 829–835.
83. Xu, L.; Huang, D.; Tsai, W.-T. Cloud-based virtual laboratory for network security education. *IEEE Trans. Educ.* **2014**, *57*, 145–150. [CrossRef]
84. Esquembre, F. Facilitating the creation of virtual and remote laboratories for science and engineering education. *IFAC-PapersOnLine* **2015**, *48*, 49–58. [CrossRef]
85. İnce, E.; Kırbaşlar, F.G.; Güneş, Z.Ö.; Yaman, Y.; Yolcu, Ö.; Yolcu, E. An innovative approach in virtual laboratory education: The case of "IUVIRLAB" and relationships between communication skills with the usage of IUVIRLAB. *Procedia-Soc. Behav. Sci.* **2015**, *195*, 1768–1777. [CrossRef]
86. Achuthan, K.; Bose, L.S. Concept mapping and assessment of virtual laboratory experimental knowledge. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015; pp. 887–893.
87. Chen, X. Research on User Identity Authentication Technology for Virtual Laboratory System. In Proceedings of the 2015 Sixth International Conference on Intelligent Systems Design and Engineering Applications (ISDEA), Guiyang, China, 18–19 August 2015; pp. 688–691.
88. Li, Y.; Xiao, L.; Sheng, Y. Virtual laboratory platform for computer science curricula. In Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, USA, 21–24 October 2015; pp. 1–7.
89. Roy, G.; Ghosh, D.; Mandal, C. A virtual laboratory for computer organisation and logic design (COLDVL) and its utilisation for MOOCs. In Proceedings of the 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE), Amritsar, India, 1–2 October 2015; pp. 284–289.
90. Chezhin, M.S.; Efimchik, E.A.; Lyamin, A.V. Automation of variant preparation and solving estimation of algorithmic tasks for virtual laboratories based on automata model. In Proceedings of the International Conference on E-Learning, E-Education, and Online Training, Novedrate, Italy, 16–18 September 2015; pp. 35–43.
91. Erdem, M.B.; Kiraz, A.; Eski, H.; Çiftçi, Ö.; Kubat, C. A conceptual framework for cloud-based integration of Virtual laboratories as a multi-agent system approach. *Comput. Ind. Eng.* **2016**, *102*, 452–457. [CrossRef]
92. Heradio, R.; de la Torre, L.; Galan, D.; Cabrerizo, F.J.; Herrera-Viedma, E.; Dormido, S. Virtual and remote labs in education: A bibliometric analysis. *Comput. Educ.* **2016**, *98*, 14–38. [CrossRef]
93. Potkonjak, V.; Gardner, M.; Callaghan, V.; Mattila, P.; Guetl, C.; Petrović, V.M.; Jovanović, K. Virtual laboratories for education in science, technology, and engineering: A review. *Comput. Educ.* **2016**, *95*, 309–327. [CrossRef]
94. Trnka, P.; Vrána, S.; Šulc, B. Comparison of Various Technologies Used in a Virtual Laboratory. *IFAC-PapersOnLine* **2016**, *49*, 144–149. [CrossRef]

95. Castillo, L. A virtual laboratory for multiagent systems: Joining efficacy, learning analytics and student satisfaction. In Proceedings of the 2016 International Symposium on Computers in Education (SIIE), Salamanca, Spain, 13–15 September 2016; pp. 1–6.
96. Chacón, J.; Farias, G.; Vargas, H.; Dormido, S. Virtual laboratory of a Spider Crane: An implementation based on an interoperability protocol. In Proceedings of the 2016 IEEE Conference on Control Applications (CCA), Buenos Aires, Argentina, 19–22 September 2016; pp. 827–832.
97. Francis, S.P.; Kanikkolil, V.; Achuthan, K. Learning curve analysis for virtual laboratory experimentation. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 1073–1078.
98. Sheng, Y.; Huang, J.; Zhang, F.; An, Y.; Zhong, P. A virtual laboratory based on HTML5. In Proceedings of the 2016 11th International Conference on Computer Science & Education (ICCSE), Nagoya, Japan, 23–25 August 2016; pp. 299–302.
99. Yu, J.; Dong, K.; Zheng, Y. VLAB-C: Collaborative Virtual Laboratory in Cloud Computing and Its Applications. In *Big Data Applications and Use Cases*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 145–174.
100. Daineko, Y.; Ipalakova, M.; Muhamedyev, R.; Brodyagina, M.; Yunnikova, M.; Omarov, B. Use of Game Technologies for the Development of Virtual Laboratories for Physics Study. In Proceedings of the International Conference on Digital Transformation and Global Society, St. Petersburg, Russia, 22–24 June 2016; pp. 422–428.
101. Achuthan, K.; Francis, S.P.; Diwakar, S. Augmented reflective learning and knowledge retention perceived among students in classrooms involving virtual laboratories. *Educ. Inf. Technol.* **2017**, *22*, 2825–2855. [CrossRef]
102. Mor, E.; Santanach, F.; Tesconi, S.; Casado, C. CodeLab: Designing a Conversation-Based Educational Tool for Learning to Code. In Proceedings of the International Conference on Human-Computer Interaction, Las Vegas, NV, USA, 15–20 July 2018; pp. 94–101.
103. Wan, H.; Liu, K.; Lin, J.; Gao, X. A Web-based Remote FPGA Laboratory for Computer Organization Course. In Proceedings of the 2019 on Great Lakes Symposium on VLSI, Tysons Corner, VA, USA, 9–11 May 2019; pp. 243–248.
104. Si, H.; Sun, C.; Chen, B.; Shi, L.; Qiao, H. Analysis of Socket Communication Technology Based on Machine Learning Algorithms Under TCP/IP Protocol in Network Virtual Laboratory System. *IEEE Access* **2019**, *7*, 80453–80464. [CrossRef]
105. Gucwa, K.J.; Cheng, H.H. RoboSim: A simulation environment for programming virtual robots. *Eng. Comput.* **2018**, *34*, 475–485. [CrossRef]
106. Deng, X.; Jin, Q.; Wang, D.; Sun, F. ARCat: A Tangible Programming Tool for DFS Algorithm Teaching. In Proceedings of the 18th ACM International Conference on Interaction Design and Children, Boise, ID, USA, 12–15 June 2019; pp. 533–537.
107. Morales-Menendez, R.; Ramírez-Mendoza, R.A. Virtual/Remote Labs for Automation Teaching: A Cost Effective Approach. *IFAC-PapersOnLine* **2019**, *52*, 266–271. [CrossRef]
108. Sanchez-Herrera, R.; Mejías, A.; Márquez, M.; Andújar, J. The Remote Access to Laboratories: A Fully Open Integrated System. *IFAC-PapersOnLine* **2019**, *52*, 121–126. [CrossRef]