OLYMPUS: AN INTELLIGENT INTERACTIVE LEARNING PLATFORM FOR PROCEDURAL TASKS

Aitor Aguirre[1], Alberto Lozano[1], Mikel Villamañe[2], Begoña Ferrero[2] and Luis Matey[3]
[1]*CEIT, Manuel de Lardizábal 15, 20018 San Sebastián, Spain*
[2]*Department of Computer Languages and Systems, University of the Basque Country UPV/EHU, San Sebastián, Spain*
[3]*CEIT and Tecnun, University of Navarra, 20018 San Sebastián, Spain*

Keywords: Simulation, Interactive System, Intelligent Tutoring System, Procedural Tasks, Authoring Tool.

Abstract: Providing Interactive Systems with educational capabilities is essential in order to achieve real effectiveness in simulation based training. However, the development cost of adding intelligence to those systems is huge. In this paper we present OLYMPUS, a generic learning platform that allows integrating Interactive Systems with an Intelligent Learning System. The platform is able to diagnose students' activity while they solve procedural tasks, it assists them with feedback and helps instructors follow students' progress with a monitoring tool. Additionally, OLYMPUS provides the instructional designers with authoring tools to help them during the knowledge acquisition process.

1 INTRODUCTION

Human beings spend a considerable part of their lives learning how to carry out every type of activity. During our lives we acquire different types of skills, and a lot of them share a common feature: they are procedural tasks. The learning process of this type of task is not easy. Sometimes the tasks to be learned can be dangerous, or they must be acquired in real environments. Unfortunately, most of the time the cost of trainings is unaffordable, and for this reason, the use of Virtual Reality (VR) or Mixed Reality (MR) based training systems is the best solution.

In order to achieve real effectiveness, VR systems need to be combined with Intelligent Learning Systems (Mellet d'Huart, 2002). This means Interactive Intelligent Learning Systems (IILS) are a step ahead of common VR training systems, because they assist the students in the learning process of a task or a set of tasks. An IILS is composed of an Interactive System (IS, an interface based on VR or MR), and of an Intelligent Learning System (ILS). The former aims at achieving faithful reproduction of reality, while the latter is centered in the instruction and learning processes. Combining those two, IILSs that are capable of supporting students in their learning process are obtained.

Our objective in this work has been to mimic a real student-expert learning process, that is, transferring expert knowledge to students. In order to achieve this, we propose OLYMPUS, a generic learning platform that has the ability of diagnosing students' activity while they solve procedural tasks, it assists them with feedback when needed and provides the instructors with a monitoring tool. In order to give students this intelligent assistance, our platform's kernel contains a set of generic modules that execute a methodological process to give intelligent feedback in any Interactive System for training procedural tasks. The process has three main steps: firstly the events happening in the virtual environment are observed; then, captured observations are interpreted, so high level representation of students' activity is obtained; and lastly, the activity is diagnosed in order to detect students' errors and possible gaps in their knowledge. The diagnostic results are used by other modules of the platform to generate feedback for the students or by any other educational component that might be integrated. Still, the amount of information required for this process is huge and it needs to be modelled, which usually is a tedious and time consuming task.

In this paper, we present a general overview of OLYMPUS, which is composed of a framework that facilitates the development of IILSs for specific domains, its runtime kernel, a set of tools that help to complete the knowledge models required by the runtime kernel (also known as knowledge acquisition process), and a monitoring tool that provides the instructors with visual information about the students activity. The paper begins briefing some other relevant related works in the field of Intelligent Tutoring Systems which have inspired our work. After that, the learning platform is described and, next, an example of the knowledge acquisition process provided by OLYMPUS' expert tool for a top-of-the-range truck driving IILS is presented. Finally the main conclusions are stated and some of our current lines of reasearch are indicated.

2 RELATED WORK

The area of Artificial Intelligence (AI) in education has followed different paradigms of development throughout history. Among them, during the 80s and 90s, Intelligent Tutoring Systems (ITS) started to emerge. ITSs are computer systems for intelligent tutoring which provide many of the benefits of one-on-one instruction without requiring a tutor for every student (Bloom, 1984). These systems have also been integrated with ISs, allowing the students to "learn by doing" in real world contexts. Since the first ITSs, three important approaches have been established: model-tracing tutors (Anderson and Pelletier, 1991), constraint based tutors (Mitrovic et al., 2009) and example based tutors (Aleven, 2005). Constraint based tutors and example based tutors were developed to reduce the cost of the process of building an ITS, although it still requires substantial expertise in AI and programming, and that is why ITSs are difficult and expensive to build. In order to avoid this obstacle, authoring systems have been shown as a successful solutions. Some of them can build tutors that integrate simulators, but their simulation capabilities are quite limited. XAIDA (Wenzel et al., 1999), RIDES (Munro et al., 1997), VIVIDS (Munro and Pizzini, 1998) and SIMQUEST (Joolingen and Jong, 1996) can be placed in this group.

Remolina´s flight training simulator (Remolina, 2004) is a more sophisticated ITS authoring tool. It is designed for non-programmers so it offers a GUI to edit task-principles, exercises and student models. Further, tasks are described by finite state machines where the situations that the students are going to face are defined. These state machines are related to students' activity, so they are able to discern if the previously defined learning objectives have been achieved. Depending on the skills acquired by the students, the student model will be modified. Although these features are quite powerful, they require some programming skills. In addition, the relation between the authoring tool and the simulator is high, which involves representing low level information, and hence, it increases the probability of generating an incomplete domain model. A similar approach is followed by The Operator Machine Interface Assistant (OMIA) (Richards, 2002), which includes a scenario generator tool and an ITS. The scenarios are edited using a visual authoring tool, where elements of the simulation are defined so they can be detected later in the simulation. It also allows for configuring some parameters for the ITS. OMIA is capable of providing automatic diagnosis and, depending on the exercise definition, it provides students with enhancements and simulates different discussions with other crew members.

In recent years, more ITS authoring tools have been proposed, which address more efficiently the problems involved on ITS development. CTAT (Aleven et al., 2006) is composed of a set of authoring tools that allow creating Example-Tracing tutors. For this kind of tutor, the author carries out demonstrations of how the students should solve a particular problem. CTAT offers tools to implement student interfaces and to add correct and incorrect examples of solutions. Using these authoring tools implies defining each solution separately, which can be time consuming as the number of possible solutions increases (eg. environments with a high grade of unpredictability). With the aim of reducing the tutor development time, SimStudent (Matsuda et al., 2007) generalizes authors' demonstrations, so not all the possible solutions need to be defined. ASPIRE (Mitrovic et al., 2009) is another authoring system for both procedural and non procedural tasks. The system allows creating domain independent constraint based ITSs. To achieve this, it provides a workspace to generate the domain ontology, which is the base for the author to define the solutions of the tasks that are going to be learned. These solutions are defined using constraints, and one of the advantages of ASPIRE is that it can generate the constraints automatically. Another domain independent ITS framework is ASTUS (Paquette et al., 2010), which focuses its efforts on knowledge representation. Among its strengths are the capacity of recognizing the composition of errors made by the students, the generation of feedback for specific errors, and the generation of hints and demonstrations for specific steps. However, it is only suitable for well-defined domains, so it can be difficult to use for ill-defined domains.

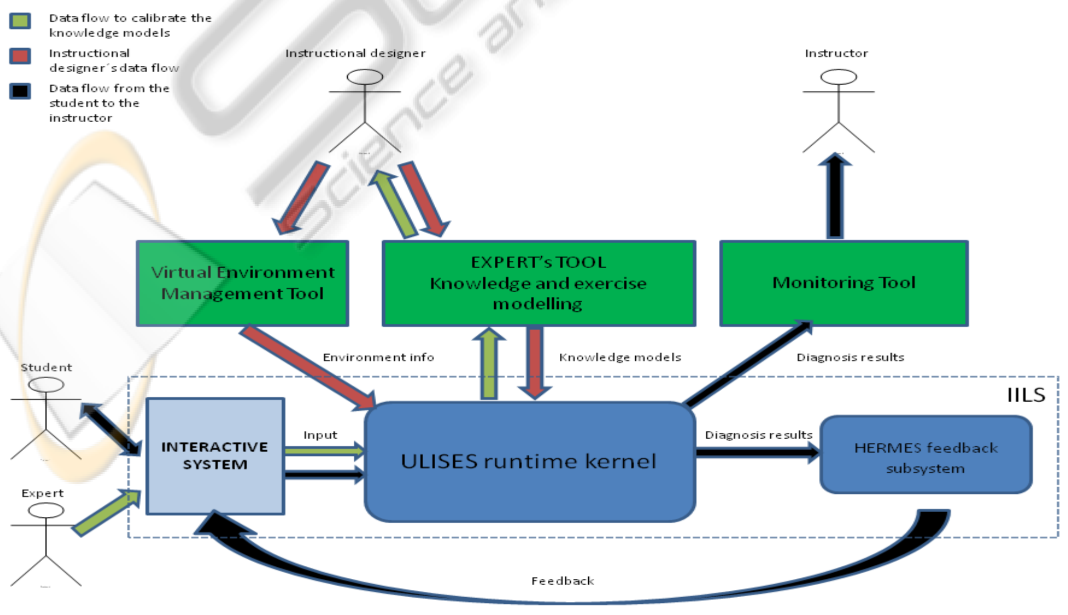
OLYMPUS was designed with the objective of adding educational capabilities to a wide range of Interactive Systems. In addition, our efforts are centred on achieving sophisticated knowledge representation independently from the learning domain. OLYMPUS allows the instructional designers to create personalised learning courses. The platform is composed of various tools: The Virtual Environment Management Tool allows the instructional designers to identify objects of the virtual environment that are interesting for the task solutions and to define knowledge about them. The Expert's Tool lets them create different knowledge models in order to design tasks for the students, as well as capture and analyze experts' activity as an aid to define the models. In this manner, the IILSs created with OLYMPUS are able to automatically diagnose, assist and monitor students while they perform tasks. In comparison with the above mentioned ITS authoring tools, OLYMPUS offers a multi-technique diagnosis module. Thanks to this feature, different diagnosis techniques can be integrated depending on the domain where the IILS is going to be used.

3 OLYMPUS' ARCHITECTURE

In order to obtain the benefits of a real one-on-one tutoring, an IILS has to be able to diagnose, give suitable feedback and monitor students' activity. In order to attain this, we defined the architecture of OLYMPUS that is shown in *Figure 1*. This architecture involves the complete process from knowledge acquisition to assisting students during the tasks, and four different roles can be distinguished:

- Student: the user to be trained.
- Instructional Designer: the teacher that designs the tasks.
- Expert: the professional that the instructional designers capture solving the tasks so they can define the knowledge models needed by OLYMPUS.
- Instructor: the one that supervises the students in the training sessions.

The whole process starts by designing the tasks that the instructors are going to propose to the students. This design process is guided by the methodology defined by ULISES: a generic framework for the development of IILSs that establishes the communication process between an Interactive System and other educational components in three steps: observation, interpretation and diagnosis. For the first step, the ULISES framework needs to gather information from the virtual scene where the tasks are executed, which will be defined by the instructional designer with the help of the *Virtual Environment Management Tool*. For example, in a driving simulation environment, various elements of interest can be defined with this tool: lanes, vehicle properties (eg. speed, acceleration), road signs, etc.



🟩 Data flow to calibrate the knowledge models
🟥 Instructional designer's data flow
■ Data flow from the student to the instructor
Instructional designer
Instructor
Virtual Environment Management Tool
EXPERT's TOOL Knowledge and exercise modelling
Monitoring Tool
Environment info
Knowledge models
Diagnosis results
Student
IILS
INTERACTIVE SYSTEM
Input
ULISES runtime kernel
Diagnosis results
HERMES feedback subsystem
Expert
Feedback

Figure 1: OLYMPUS architecture.

Once the observable information is defined, instructional designers can model the rest of the knowledge needed by the system. To make this possible, they have to make use of the *Expert's Tool.* With this tool, instructional designers can capture the experts' activity in the IS and thereby, design the "ideal" knowledge models. Following this process, the ULISES runtime kernel will be able to generate diagnosis information that will be delivered to *HERMES*, an adaptive and configurable feedback module that selects feedback messages in real time in response to the students' activity.

After the tasks have been modelled, instructors can launch them and supervise the students through the *Monitoring Tool.* As is shown in the student's data flow in *Figure 1*, a student's activity is sent from the IS to the ULISES runtime KERNEL, which generates a diagnosis and transmits the results to the *Monitoring Tool.* These results are displayed graphically in real time to help the instructors follow and control the training sessions.

In the following sections, the components that compose OLYMPUS are explained more thoroughly.

3.1 ULISES Framework

ULISES (Lozano-Rodero, 2009) is a framework for the development of IILSs. It is the core system of OLYMPUS, as it defines how the educational functionalities must be integrated with the IS. ULISES is based on the natural process that the teachers follow when they are supervising students. Firstly they perceive what is happening in their environment, then they interpret what is happening out there and lastly they make a diagnosis about students' activity. In order to follow this behaviour, ULISES is based on a metamodel that is divided into three main abstraction levels: observation, interpretation and diagnosis levels. These three levels describe generically the elements that have to be particularised to describe an IILS. In other words, the methodology specifies how to observe the actions being carried out in the interactive system, how to interpret the steps made by the students and the context where they are executed, and lastly, how to diagnose them.

ULISES represents each level of the metamodel with a corresponding model, thus, the framework needs to have the observation, interpretation and diagnosis models. For this reason, ULISES's architecture is composed with the corresponding three subsystems. ULISES runtime kernel subsystems communicate with the others via multi-agent architecture, using the FIPA standard of interoperability. This architecture allows the agents to exchange information by subscription, request and query communication protocols.

3.2 HERMES Feedback Subsystem

HERMES is a domain independent feedback system whose behaviour is customisable to suit the student's characteristics and the task's context (Lopez, 2011). What is more, it takes advantage of the multimodal capabilities of ISs. HERMES selects its feedback based on the diagnosis results generated by ULISES. This feedback selection process discerns the most important action among all the actions that are being executed by a student in a particular moment. Further, it also determines what the most appropriate message is, taking into account students' characteristics. The selection algorithm takes into account both assimilation of the messages and educational factors. Besides, the algorithm makes it possible for HERMES to customise its behaviour to the characteristics of the domain and to the experts' preferences.

3.3 Virtual Environment Management Tool

The objective of the tool is to interactively generate knowledge associated with the elements in the training scenarios. To achieve this, three steps need to be followed according to the following: (1) define the scene ontology, (2) create the knowledge-mesh and (3) validate knowledge observation.

1. Firstly, the objects that are going to appear in the simulation need to be specified. Those objects are contained in an ontology that is called scene ontology. This ontology will be composed of classes and their properties. For example, if a lane needs to be defined in the IS, a class will be created for it with its corresponding properties: maximum speed of the lane, type of lane, etc.
2. Once the scene ontology is created, the next step is to define the physical representation of the objects. This definition is done by assigning customisable geometries called knowledge-mesh. Additionally, the simulation environment can be provided by observers. Their function is to observe the environment during a simulation session. To achieve this, they are equipped with sensors, so when they collide with an instance of a knowledge-mesh, the system is able to recognise the object and to

identify its properties.

3. Lastly, the tool offers the possibility of validating whether or not all the generated elements are consistent and if the generated knowledge satisfies the requirements.

3.4 Expert's Tool

The main objective of this tool is to give expression to expert's knowledge, without requiring any expertise in AI. As we have seen, ULISES and HERMES subsystems need some knowledge, which is provided by the expert's authoring tool. Regarding the ULISES framework, interpretation and task models are two models that need to be defined.

Before explaining these two models, it is indispensable to refer to the lower layer; the observation model. The basic unit of this level is the observation, which describes a fact taking place in an interval of time. Without defining observations it would be impossible to define further levels. For example, if an overtaking in the right lane needs to be described at the interpretation level, first the lane change needs to be observed. The definition of the observation model is done manually taking into account the ontologies defined in the Virtual Environment Management Tool, that is, observations are the link to join the scenario information with the ULISES framework. However, making this connection requires some programming and a simple process needs to be followed, which is out of the scope of our tools.

One step ahead in the completion of the knowledge models is the definition of the interpretation model. Its aim is to represent the necessary information to guess what actions (steps) the students are executing in the virtual environment. To do so, detecting the context where the actions are being carried out is decisive. For example, accelerating before a traffic light is not the same as in an overtaking situation. Therefore, the authoring tool provides modelling steps and situations using a constraint based approach. In domains that are not well structured, events do not

occur in a predictable manner, so temporal knowledge is not relative. For this reason, the use of constraints to define relations is an appropriate solution (Allen, 1983).

Once the steps and situations are modelled in the interpretation model, that is, once the system is able to discern what is happening in the IS, the next step is to state the correction of the actions made by the students in a given situation. This information is defined in the *Task Model*, which is composed in the following manner: a task is composed of situations, and each situation has one or many possible solutions. As has been noted, this framework is independent of how to define solutions, in other words, it is independent of the diagnosis technique. Either way, the solutions define how each step of a situation is solved correctly or incorrectly. At the same time, the Task Model will also gather the necessary information for HERMES.

Defining the information for those two models is not an easy task. The grade of complexity of the maneuvers that can occur in the IS is unpredictable. Thus, the variety of signals that come from the virtual environment can be high and the relations and patterns between these signals needs to be identified in order to generate accurate interpretation and task models. To give a solution to this problem, the Expert's Tool offers a *Capturing Tool*. Its objective is to capture experts while they interact in the IS to monitor their activity in the form of signals and to show it in the Expert's tool afterwards. Further, the tool allows analysing rigorously the information and establishing patterns between them, which is a great asset for the definition of the required models.

3.5 Monitoring Tool

Visualising the diagnosis results is as important as generating an accurate diagnosis and that is why OLYMPUS offers another visual resource: the Monitoring Tool. This tool allows the instructors to monitor students' activity in real time. The activity shows which steps are being executed and in which situations they are happening. And, what is more, the monitoring tool indicates if each situation and step has been carried out correctly or incorrectly.

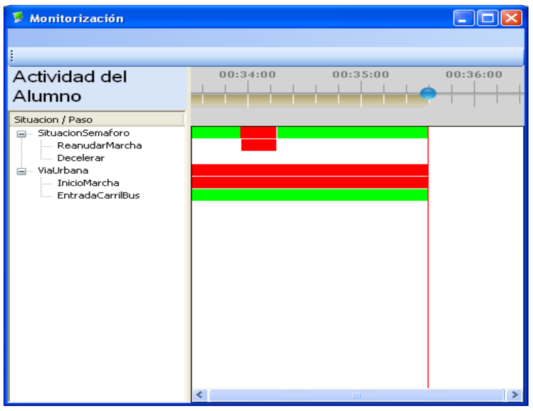


Figure 2: View of the expert capturing tool.

Although the diagnosis results are showed in real time (*Figure2*), when the exercise is completed a thorough report of the exercise can be displayed. In this report, the marks obtained for each situation and steps can be consulted, as well as information about the acquisition grade of the learning objectives. Besides, students' results from each exercise are saved for the instructor to follow the progress that the students are having over time.

4 THE MODELLING PROCESS WITH THE EXPERT'S TOOL

After having done a general overview of the different tools that OLYMPUS offers, in this chapter, examples of how to model a task with the Expert's Tool will be described. This tool is the core tool of OLYMPUS, as it makes the rest of the tools run. Firstly, it uses the information defined in the scenario, and then it generates the interpretation and task models, which feed the ULISES runtime kernel and HERMES feedback subsystem. In this manner, HERMES can generate personal feedback for the students and the Monitoring Tool can show diagnosis results successfully, which would be impossible without the Expert's Tool.

4.1 Defining the Task Model

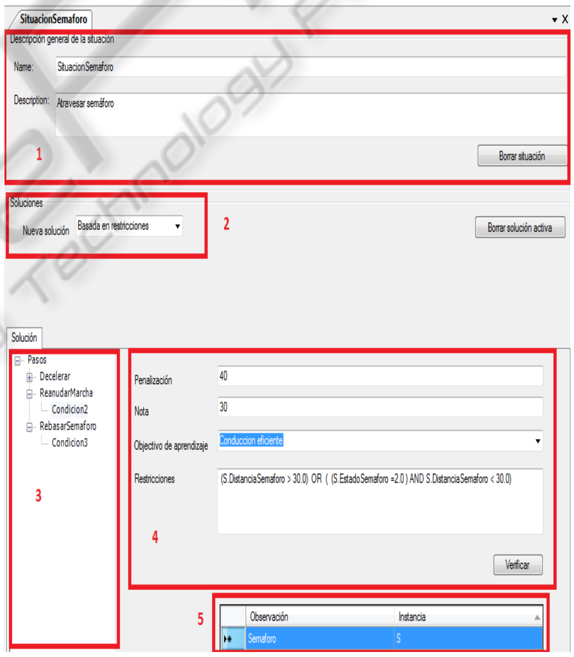
Defining the Task Model implies that experts think about the situations that the students are going to face during their training. In each task the students will need to acquire certain learning objectives, which will help the experts to know if the students are acquiring the desired skills or not. In an economical driving context, an expert may want to measure if the students act with sufficient anticipation when they face a traffic light. In this case, for example, a *TrafficLight* situation could be defined. The next step is more difficult, the solution or solutions for the TrafficLight situation need to be defined. In such a context, some possible steps of interest can be accelerating, breaking and passing the traffic light. The conditions for this situation are:

- Accelerating when the traffic light is yellow or red is incorrect if the car is less than 30 metres from the traffic light.
- Passing the traffic light in red is incorrect.

Once the situation and the steps are identified, the solution for the situation needs to be specified. As noted before, these solutions will be diagnosed with a specific diagnosis technique. As our diagnosis module allows integrating different diagnosis techniques, in this case a constraint based approach has been developed due to the high unpredictability that a driving context implies. So this time, the correct steps performance will be modelled with constraints. The *Accelerate* step would be defined with this constraint:

(S.TrafficLightDistance>30.0)
OR
(S.LightState=2.0
AND
S.TrafficLightDistance < 30.0)

Figure 3 shows the solution definition for the *Accelerate* step within the *TrafficLight* situation. In order to define a constraint, instances of the observation will be used. In the constraint definition for *Acceleration* just one instance (S) of the TrafficLight is used. *TrafficLightDistance* and *LightState* are properties of the observation *TrafficLight (S)*. On the other hand, each constraint is related to a learning objective, so when an action is executed by the student, it will affect the corresponding learning objective.


SituacionSemaforo

Descripción general de la situación

Name: SituacionSemaforo

Description: Atravesar semáforo

Soluciones

Nueva solución: Basada en restricciones

Solución

```
Pasos
  Decelerar
  ReanudarMarcha
    Condicion2
  RebasarSemaforo
    Condicion3
```

Penalización: 40

Nota: 30

Objectivo de aprendizaje: Conduccion eficiente

Restricciones: (S.DistanciaSemaforo > 30.0) OR ((S.EstadoSemaforo =2.0) AND S.DistanciaSemaforo < 30.0)

Observación	Instancia
Semaforo	S

Figure 3: 1.Name and description of the situation 2.Selection of the diagnosis technique 3.Steps that are inside a situation (decelerate, accelerate and passTrafficLight 4.Definition of the constraints and parameters to calculate the marks related to a learning objective. 5.Definition of instances.

4.2 Interpreting Students' Activity

In order to diagnose the correctness of the situations and the steps that have been defined in the Task

Model, firstly they have to be interpreted, which is achieved thanks to the modelling of the steps and situations in the interpretation model. Following with the previous example, the situation *TrafficLight* and the steps *Accelerate, Break* and *PassTrafficLight* need to be modelled. Unlike the diagnosis module, the interpretation module always need that its elements be modelled using constraints. The interpretation *interface* allows entering different types of constraints, as constraints to detect the starting point of an action as well as its end point. As the cited steps are easy to model (we just need to detect if we are breaking, accelerating etc.), a more complex step is explained to illustrate the definition of a step: *StartCar*. For the case of *StartCar*, the constraints below will indicate the beginning and end of the step:

Start constraint: P2 [Meets] P3 AND P3 [Meets] P2
End constraint: P3 [Meets] P2

P2 and P3 correspond to the contact and start position of the key when the car is going to start. The *meets* relation keyword indicates that an observation (P3) starts at the same time that another observation (P2) ends.

4.3 Capturing Experts Activity

The design process of interpretation and task models is a complex task. In order to define the constraints, the relation between observations and the threshold values of the observations is unpredictable just by observing a student while interacting in the virtual environment. As the experts are the ones that have expertise in a particular domain, the objective of the tool is to capture their activity and to model the necessary models according to their activity. For example, if the objective is to define correct overtaking, the expert would have to define parameters as distance to the car ahead or how to make a lane change. For this purpose, the Expert's tool offers a tool to capture experts and to monitor their activity. The capturing tool offers interfaces that can be distributed as graphs and tabs, where the signals of interest will be placed as best suits. Furthermore, the tool offers methods to record sessions, zoom and other features that provide visual benefits that will let the instructional designers analyze the observations that interest them.

Once the models have been completed, one last step remains: their validation. In order to check whether the results of the generated models are the desired ones, the Expert's tool uses the same visual resource of the monitoring tool. In this manner, the experts will be able to see if the diagnosis results meet their criteria.

5 GENERAL DISCUSSION AND CURRENT WORK

The OLYMPUS platform has been successfully applied in an IILS for training professional truck drivers and in another IILS for the training of mentally challenged people in gardening tasks. Our IILS paradigm has been shown effective for the driving and gardening simulation domains where the unpredictability of the actions carried out by the students is high. Compared to an ITS authoring tool, our platform has proven itself capable of saving the majority of the software development effort. However, some domain specific software, as observations, need to be programmed for any domain. In order to justify this "problem", some factors need to be kept in mind. Our platform is not designed for a specific domain, so we prioritised preserving the high flexibility of the platform to avoid eliminating all programming effort. The most domain specific software in this kind of simulation based scenarios forces us to program all the cases that can occur in the simulation context and to write the code for action/decision correctness evaluation. In our case, thanks to the Expert's tool and ULISES, all this effort is saved.

The constraint based approach used in this work has provided a good base for the definition of task models. Still, there have been some situations where we have missed other diagnosis techniques. For example, regarding to the truck simulation, in the case of crossroads the constraint based diagnosis technique was not sufficient. This is because when a crossroad is going to be exited, some previous steps need to be taken into account in order to diagnose correctly such a situation. Nevertheless, this is not an unsolvable case; our platform allows integrating multiple diagnose techniques at the same time. This hybrid diagnose technique can be really useful in various domains. In some domains, it is advantageous to use knowledge discovery techniques for automatically learning a partial problem space, but in other cases, as in the driving domain, it is not suitable.

Although the scenario definition and the knowledge models definition take a relatively short time, the Virtual Environment Management Tool and the Expert's tool need a learning process. Both tools have proven themselves easy to use, but due to the number of features they provide, it takes time learning to use the tools. In our opinion, it is worth spending the time on the familiarisation process with the tools, due to the huge amount of programming time that is saved.

At the moment, OLYMPUS has been tested in the truck driving domain and in the development of an IILS of gardening tasks for disabled people, so further evaluation of the platform is needed in order to test the limits of its generality. While for the driving domain just the constraint based diagnosis technique has been implemented, a research avenue would be to integrate other diagnosis techniques and in order to take advantage of the benefits of each technique. For beginners, our intention is to apply our platform in an ill-defined domain. Completing solutions with a constraint based solution in ill-defined domains can be a tedious task, so our intention is to implement techniques similar to the ones used in model-tracing tutors in order to expedite even more the knowledge definition process for OLYMPUS. Following the same principle, we are working on automating the generation of interpretation models by using data mining methods.

REFERENCES

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2005). Example-Tracing Tutors : A New Paradigm for Intelligent Tutoring Systems. *Human-Computer Interaction*.

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2006). The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains.

Anderson, J. R., Pelletier, R., (1991). “A Development System for Model-Tracing Tutors”. In *Proceedings of the International Conference of the Learning Sciences.* Charlottesville.

Bloom, B. S., 1984. “The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring”, Educational Researcher, 13(6): 4-16).

Joolingen, W. R., Jong, T. d., (1996). “Supporting the authoring process for simulation-based discovery learning”, In *Proceedings of European Conference on Artificial Intelligence in Education,* pp. 66-73, Brna.

Lopez-Garate, M., (2011). “Sistema de Selección de Feedback Adaptativo y Configurable para Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje”. Thesis, University of Navarra, Donostia (Spain).

Lozano-Rodero, A., (2009). “Metodología de Desarrollo de Sistemas Interactivos Inteligentes de Ayuda al Aprendizaje de Tareas Procedimentales basados en Realidad Virtual y Mixta”. Thesis, University of Navarra, Donostia (Spain).

Pizzini, Q., Munro, A., Wogulis, J., Towne, D., (1996). “The Cost-Effective Authoring of Procedural Training”. In *Proceedings of Intelligent Tutoring Systems.*

Matsuda, N., Cohen, W. W., Sewall, J., Lacerda, G., & Koedinger, K. R. (2007). Predicting Students ’ Performance with SimStudent : Learning Cognitive Skills from Observation 1. *Learning*.

Mellet d’Huart, D., 2002. “Virtual Environment for Training: An Art of Enhancing Reality”. In *Workshops of the Intelligent Tutoring Systems conference.* Donostia (Spain).

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Mcguigan, N., & Zealand, N. (2009). ASPIRE : An Authoring System and Deployment Environment for Constraint-Based Tutors. *International Journal of Artificial Intelligence in Education*, *19*, 155-188.

Munro, A., Jonhson, M., Pizzini, Q., Surmon, D., Towne, A., Wogulis, J., (1997). “Authoring simulation-centered tutors with RIDES”, *International Journal of Artificial Intelligence in Education,* vol. 8 (3-4), Pp. 284-316.

Paquette, L., Lebeau, J.-françois, & Mayers, A. (2010). Authoring Problem-Solving Tutors : A Comparison between ASTUS and CTAT. *Advances*, (Murray 2003), 377-405.

Remolina, E., Ramachandran, S., & Stottler, R. (2004). Intelligent Simulation-Based Tutor for Flight Training. *Security*, (1743).

Richards, R. A., 2002. Common Cockpit Helicopter Training Simulator. In *AVSIM*.

Wenzel, B. M., Dirnberger, M. T., Hsieh, P. Y., Chudanov, T. J., Halff, H. M., “Evaluating Subject Matter Experts’ Learning and Use of an ITS Authoring Tool”. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems.*