Abstract
This paper describes the development of a generic teaching environment that uses agents to support learning. An Intelligent Tutoring System will be described, that guides students during learning. This system is generic in the sense that the core is separated from the exercise modules and user interfaces. Exercises can be provided to the system in the form of XML data-files. A user interface can be text-based or 2D, but it can also be a 3D virtual reality environment. An application of the teaching environment for nurse training is described.
Abstract

This paper describes the development of a generic teaching environment that uses agents to support learning. An Intelligent Tutoring System will be described, that guides students during learning. This system is generic in the sense that the core is separated from the exercise modules and user interfaces. Exercises can be provided to the system in the form of XML data-files. A user interface can be text-based or 2D, but it can also be a 3D virtual reality environment. An application of the teaching environment for nurse training is described.

1. Introduction

This paper describes the development of a teaching environment that uses agents to support learning. A software system called Ines will be described, that guides students during learning. This means that students can use the system to train certain tasks. Before the Ines project started, the idea was to make an Intelligent Tutoring System for nursing students. Hence the name: Intelligent Nursing Education Software.

The main goal of the Ines project is to provide an effective teaching environment. For an environment to be effective as a teaching environment, it has to fulfill two conditions. The first condition that has to be satisfied is that the teaching environment provides a correct and logical guidance system for the student. This means that a teaching system must be able to provide sensible feedback, demonstrations and/or explanations. The second condition is that the user interface of the system is as optimal and accessible as possible. For this purpose, one could for example use a simple 2D graphical interface, but the interface could also consist of a highly accurate 3D virtual environment.

Until now, the Ines project has mainly concentrated on the first condition. Independent of the task trained, the system is capable of providing feedback, explanations and demonstrations. For this purpose, a number of agents, which together form the main part of Ines, are implemented. The agents observe the students actions and check whether they are performed correctly and in a correct order. In the future, the intention is to concentrate more on the second condition.

Ines is a generic teaching system in the sense that the core is separated from the exercise modules and user interfaces. The exercises can be provided to Ines in the form of data-files. The information from these files is used to check if the actions from students are right, and to provide explanations and demonstrations if necessary. An exercise will typically consist of subtasks with some partial ordering defined for them. The tags in the data-files are general, and can be filled in for repairing a punctured tyre as easy as for a nursing task, for instance.

Adding a new teaching task to Ines also means adding a new interface. A new interface may be text-based or 2D, but it can also be a 3D virtual reality environment. Even haptic feedback devices and a head-mounted device can be used, without making changes to the Ines system itself.

The implemented system is applied to a nurse training task. To demonstrate how the system works, an exercise-file and an interface were made for the subcutaneous injection.

The organization of this paper is as follows. First, in section 2 some Intelligent Tutoring Systems that gave the inspiration for the design of Ines are described. Section 3 contains a description of the architecture of Ines. Then, the most important part of the system, the cognition, is described extensively in section 4. The paper concludes with a section about future work.

2. Existing ITS’s

There are a lot of programs that already teach certain tasks to users. The most important systems that gave the inspiration for the design of Ines will be mentioned in this
The Parlevink group, researching language, knowledge and interaction at the University of Twente, combines research on human-computer interaction, dialog systems, autonomous agents and virtual worlds. The group has built the VMC (Virtual Music Centre), which is a replica of an existing music centre and serves as an environment to experiment with multi-modal interaction with agents or other visitors to this virtual world [5]. In this context, Jacob was built, an instruction agent that helps users to solve a mathematical puzzle in virtual reality [2].

Based on the experience with Jacob, the idea arose for creating ADRI (Artificial Didactic Recital Instructor), a system intended to aid people learning to play the piano [1]. A 3D-world with a virtual piano and visualization of notes is connected with a real synthesizer using Midi to interact with the user. Also a multi-agent platform is used within the system. Each agent has a special expertise and knowledge domain and can give information or act when it decides that it is appropriate.

Other instruction agents that inspired the design of Ines are Steve [3], [8], [7] and Adele [3], [9]. Steve is an animated pedagogical agent. It gives instruction in procedural tasks in an immersive virtual environment. To allow Steve to operate in a variety of domains, its architecture has a clear separation between domain-independent capabilities and domain-specific knowledge. Adele (Agent for Distance Learning Environments) is developed by the USC/Information Sciences Institute’s Center for Advanced Research in Technology for Education (CARTE). Adele runs in a student’s web browser, and is designed to be integrated into web-based electronic learning materials. Adele-based courses are currently being developed for continuing medical education in family medicine and graduate level geriatric dentistry.

Another important system that inspired the design of Ines is LAHYSTOTRAIN [4]. This is a training system for two types of minimally invasive surgery techniques. It combines a Virtual Reality Simulator, a Basic Training System that provides web based theoretical training, and an agent-based tutoring system, the Advanced Training System, oriented to supervise the execution of practical exercises. Training in LAHYSTOTRAIN is carried out in two temporal consecutive phases: acquisition of theoretical knowledge with the Basic Training System, and acquisition of practical skills with the Advanced Training System.

### 3. System Architecture

3.1. Input from Devices

The first main part is formed by the input from the real world. This can be any input from any device, such as input from datagloves, mouse/keyboard, or haptic devices. The input is used to perform actions in the concrete user interface. Since Ines is completely separated from the concrete user interface, Ines does not care which kind of input device is used. She only expects to receive the necessary data from the concrete user interface, such as the position of the objects within the user interface. The choice of input devices is thus dependent on the concrete user interface that is used, and not of Ines itself.

3.2. Concrete user interface

The concrete user interface takes care of the visualization of the exercise. This is a very broad definition, which allows a great freedom for the system implementer to alter or create a user interface for an exercise. Any user interface can be implemented, from a simple dialog interface or a 2D graphical interface, to a highly accurate 3D virtual environment. The architecture of Ines allows to have several different user interfaces for the same exercise. Therefore, the contents of the concrete user interface as shown in figure 1 is merely the contents of an example user interface (in this case a virtual 3D environment).

In figure 2, the user interface that was implemented for the nursing application can be found. The left side of the figure contains data from a patient, the right side contains...
descriptions of all subtasks. The bottom contains buttons that can be used to ask for explanations.

Figure 2. User interface of the nurse training application.

All user interfaces that are to be used by Ines should be able to communicate important information to Ines. Without this information, it is impossible for Ines to perform her task. For example, Ines needs to know when the student has picked up an object before she can decide whether that is a correct action or not. To establish this communication from the concrete user interface towards Ines, the programmer of the user interface has an API especially designed for this purpose at his/her disposal.

3.3. Abstract user interface

The abstract user interface is a high level model of the concrete user interface. The concrete user interface directly communicates with the abstract user interface, in that the concrete user interface reports important changes to the abstract user interface and vice versa.

The abstract user interface is, contrary to the concrete user interface, a part of Ines herself. This means that the communication between the two interfaces in fact constitutes the communication between the concrete user interface and Ines. The methods from the Ines Interaction API, mentioned in the previous section, are used to establish this communication.

By separating the concrete user interface from the abstract user interface, Ines does not have to contain the exact representation of each user interface, but only the general aspects that are applicable to all user interfaces, for example the position of important objects.

3.4. Cognition of Ines

The fourth and final part in the Ines architecture is, from a teaching perspective, the most interesting part. This part represents the knowledge and skills of Ines and is therefore the core of the Intelligent Tutoring System. The next section will deal with this cognition of Ines.

4. The core of Ines: the cognition

The cognition of Ines consists of four models as can be seen in figure 1: the domain model, the student model, the instruction model and the patient model. This last model was added to Ines for the purpose of exercises for nursing students. The patient model is an agent that at the moment only describes all knowledge that is known about a patient, like age, sex, religion and so on. In the future, this agent can be extended so that the patient can react to actions from the students, for example pull back his or her arm in anxiety when a virtual reality representation is used. When exercises for non-nursing tasks are added to the system, the patient model can be ignored or replaced by a similar or other model. This is also the reason why the patient model is not a part of the domain model.

4.1. Domain Model

The domain model in Ines encapsulates knowledge about the exercises that should be practiced. Because Ines must be capable of training several different tasks, and because it is unknown which tasks Ines should train in future, it would be very awkward to program all data about all exercises in the system itself. The solution developed for this problem is to only define the elements of the domain model. These elements do not contain any exercise data yet. The real data can be read from an external information source: an XML-file. Thus, instead of programming each exercise into the system itself, a general domain model is used that is filled in by data presented to Ines.

The XML-file contains all the information that Ines needs to know about subtasks, their order, their constraints and explanations and demonstrations. All subtasks have preconditions and postconditions, and these conditions are used by an algorithm to check the order of the subtasks and also to check if errors are made within the subtask. The preconditions describe the conditions that have to be present before the subtask is eligible for execution, and postconditions describe the condition changes that should have happened after the subtask has been completed.

Explanations associated with conditions will usually mention why a certain condition has to be met before the subtask can be performed. Explanations associated with a subtask itself will describe the necessity of the subtask.
within the exercise, but can also explain what the student has to do during the subtask. The XML-file also contains references to files that can be used to demonstrate how a subtask has to be performed, and references to the necessary user interface. It is possible to have several different user interfaces for the same exercise, for example for beginners, medium and advanced level students. These references are preventing Ines from containing all user interfaces and demonstrations herself.

When adding a new exercise to the system, the data should be added to an XML-file with exercise data. This can be done by hand, but since an average exercise consists of a lot of data, this task can be very error prone. To prevent mistakes from being made, an editing program has been created that can be used to alter or to create an exercise file. This program is called “Ines Exercise Editor” and is a stand-alone application that can be executed without the presence of any Ines software.

After reading the necessary data from the XML-file, Ines is able to perform her teaching task. The contents of the domain model are used to observe the abstract user interface. All actions within the interface are interpreted by Ines, and Ines will respond to certain events. This conduct of Ines is determined by the instruction model.

4.2. Student Model

The student model is a model of the user’s characteristics and performance. For the same reason as with the domain model, the student model is also defined in terms of elements that can be filled with data from an XML-file. This file contains data about all students that work with the program: all the information that is needed in order to identify the students and to interpret the results of the students.

A list of exercises is kept for each student. This list contains all the exercises that the student has done in the past, and the results that are achieved for these exercises: the number of times that the student has tried the exercise and, when the exercise is finished successfully, the grade that the student received for the latest attempt. This grade is calculated by one of the agents in the instruction model (the Examination Agent) and then stored in the student XML-file. Also for each subtask, the number of tries is kept, and for each try if the student has passed the subtask successfully.

4.3. Instruction Model

Ines is an Intelligent Tutoring System based on agent technology. This means that agents are used in the instruction model to provide the instruction to the student. The instruction model observes the world, the task and the user’s actions. It can manipulate the world and the user through utterances and actions. These actions include giving a demonstration of the next step of the task, telling what the next step is, or explaining a certain step. Each agent in the instruction model describes a small instructional part of the teaching strategy used: the practice and drill teaching method [6]. With this method, the student has to perform the exercise over and over again, until he masters the task.

For the implementation of the agents, the agent platform developed by the Parlevink group is used. In this platform, there are two kinds of agents: proactive agents and reactive agents. Reactive agents only take actions if they receive messages from other agents, while proactive agents take actions once in a particular amount of time, for example per second.

![Figure 3. The agents. An arrow between two agents means that the first agent sends messages to the second agent.](image)

The agents within Ines can also be divided into these two groups. Each proactive agent observes the user interface with respect to one single aspect. When such an agent has made an observation, it will send it to a reactive agent. Reactive agents receive the observations and respond to it. In figure 3 can be found which agents send messages to which other agents.

Proactive agents

The Time Agent keeps an eye on the elapsed time during a subtask and checks if this is shorter than a certain time-limit. The Error Agent checks if the student makes errors while carrying out the subtasks. The Task Observer Agent is responsible for keeping an eye on the order in which the several subtasks are performed. This means that the Task Observer agent will check whether the order in which the subtasks are performed is correct and, in addition, whether the student doesn’t mix up a number of subtasks. The Sterile Agent is the only agent within Ines that is implemented for the prototype with the nursing exercise. It checks the sterility of the used objects. For other (non-nursing) tasks, this agent can be replaced by another kind of agent.
Reactive agents

The Feedback Agent is able to generate either positive or negative feedback. It receives information from the Examination Agent, which detects whether a student has performed a subtask (in)correctly. This received information is used to generate appropriate feedback (example: “right” or “wrong”). The Explanation Agent determines when the explanations and demonstrations contained in the exercise XML-file are given. The Interaction Observer agent is responsible for keeping up which feedback and demonstrations are given to the student. The data can be used later to see what the student has done and what he or she should know.

The Examination Agent plays a central role in the complete arsenal of agents. It receives all error messages from the proactive agents and acts upon them. Dependent on the agent that has sent the message and on the contents, a new message is sent by the Examination Agent to one of the other processing agents. The Examination Agent is also responsible for updating the student results during the execution of an exercise. When the student finishes the exercise, it will derive a grade for the student and will write this to the student file. The final task of the Examination Agent is to control the creation of all other agents. When instantiating the Examination Agent, all necessary other agents will be instantiated. When quitting the Examination Agent, all created agents will also quit.

5. Future work

A next step in the Ines project is to make a more advanced user interface, for example a 3D virtual environment using haptic feedback devices. In this new user interface, the student must be able to take objects and to perform actions with them. With such an interface, Ines can be tested by students, and they can give feedback on the working of Ines and tips for improvement and extensions. When Ines is tested in this way and found to be working, new exercises can be added to the system and students can learn them.

Ines is extendable in many ways. A number of extensions we are working on are:

- Change Ines so that students can also be examined. This mode differs from the training mode in that the student is not able to ask any questions and that Ines will not give any feedback or hints while the student is executing the assignment;

- Make Ines able to offer more or less guidance to the student depending on his/her performance;

- Make Ines able to offer feedback about the whole exercise when the student has finished the whole exercise.

Now, feedback is only given at the moment that the student does something right or wrong:

- Make a virtual representation of the tutor, that can be seen on the screen. This representation can be made so that it can give demonstrations, and facial expressions can be used to show the students if they are doing well or not.

Presently, the teacher has to use the student XML-file to look at the results of the students and to add new students to the system. A separate program should be written to make it easier to check the results.

References