An Animated Pedagogical Agent
for SQL-Tutor

Honours Project

Pramuditha Suraweera

Supervisor

Dr. Antonija Mitrovic

Contents

Abstract

Animated pedagogical agents are animated characters that inhabit interactive learning environments. In addition to providing problem-solving advice in response to a student's actions, they are also able to play a powerful motivational role. This project develops an animated pedagogical agent for the computer based teaching system, SQL-Tutor. The introduction of a pedagogical agent to SQL-Tutor enables it to provide higher motivational support to the students and enhances their quality of learning.

An evaluation of the impact of the agent on the student's learning experience was carried out with second year Computer Science students from the University of Canterbury. The study revealed that the presence of an animated character with an interesting personality has a strong positive effect on student's perception of the learning experience. The study also demonstrated that students were more motivated to interact with the system equipped with the agent compared with the SQL-Tutor with no agent.

Chapter 1

Introduction

Intelligent Tutoring Systems (ITS) are an attempt to automate personal tutors. They have understanding of the subject domain to offer individualised problem-solving advice to the students. Animated pedagogical agents deliver such advice with a strong visual appeal. This research project introduces an animated pedagogical agent for an ITS.

Animated pedagogical agents inhabit interactive computer based learning environments and provide customised feedback to the students. Their lifelike and captivating presence motivates students to interact longer with educational software. A significant improvement in the quality of learning results.

SQL-Tutor is a computer based teaching system developed to assist students in learning Structured Query Language (SQL). It offers a problem-solving environment focussing on individualising problem-solving advice to a particular student. The feedback from the system is displayed as text messages that may appear monotonous.

SQL-Tutor was improved to provide more support for the student, increase their learning rate and enhance the quality of learning by the introduction of an animated pedagogical agent.

1.1 Project aims

The two main objectives of this research project are:

(a) Development of an animated agent for SQL-Tutor;
(b) Evaluation of the effectiveness of the developed animated agent.

An animated pedagogical agent named **Smart-egg** was developed to present feedback from SQL-Tutor. **Smart-egg** is a two-dimensional cartoon figure capable of performing an array of gestures. The gesture space is exploited and combined into complex behaviours to express emotions. The main role of the agent is to present feedback messages to the students in an interesting and enjoyable manner, to increase their motivation and subsequently, the effectiveness of their learning.

To fulfil the second objective, an evaluation study was carried out with second year Computer Science students at the University of Canterbury. Each of them interacted with either the original version of SQL-Tutor or SQL-Tutor with **Smart-egg**. During the interactions, the learning environment logged all the student's problem-solving activities,

and the students sat a pre- and post-tests. They were also asked to complete a system assessment questionnaire. The results from the evaluation are very promising, revealing that students perceived SQL-Tutor with **Smart-egg** as being more helpful and enjoyable in comparison to the original version.

1.2 Report Structure

The following chapter describes the background for this project, outlining Intelligent Tutoring Systems, SQL-Tutor and animated pedagogical agents.

Chapter 3 describes the development of **Smart-egg**, an animated pedagogical agent for SQL-Tutor. It outlines the drawbacks of the original version of SQL-Tutor and the benefits of introducing an animated pedagogical agent. These sections are followed by an account of the design and implantation details of **Smart-egg**.

Chapter 4 is focuses on the evaluation study. This chapter includes an outline of the experiment and a detailed account of the results and analyses.

The final chapter presents the conclusions.

Chapter 2

Background

Animated pedagogical agents are autonomous agents that support human learning in interactive learning environments. They extend and improve Intelligent Tutoring Systems (ITS) in a number of ways. This research project focuses on the areas of ITSs and animated pedagogical agents. The following sections outline ITSs, Web-based ITSs, and SQL-Tutor, an ITS developed to assist students in learning the Structured Query Language. It also includes a description of animated pedagogical agents and some examples of pedagogical agents that inhabit ITSs.

2.1 Intelligent Tutoring Systems

Empirical studies have demonstrated that effective individual tutoring is the most powerful mode of teaching. However, individual human tutoring for each and every student is logically and financially impossible. The creation of Intelligent Tutoring Systems attempts to bring the personal tutoring experience to a broader audience.

The architecture of a typical standalone ITS (illustrated in Figure 1), at a birds eye view of abstraction, consists of a tutorial component, a student model for keeping track of student progress over time, a cognitive simulation of an expert problem solver and the code underlying the interface [Alpe99].

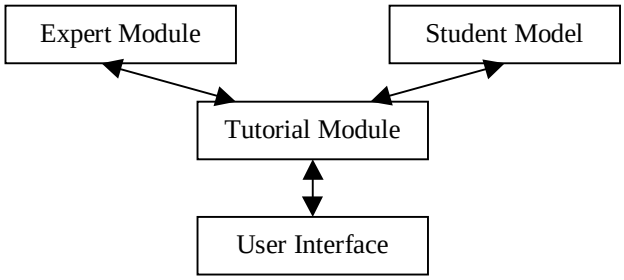


Figure 1: Architecture of an ITS [Alpe99]

The expert module accommodates a cognitive model of an ideal student problem solver. The tutorial module consults the expert module when determining the correctness of an answer and providing hints to a student. Moreover it consults the student model when generating pedagogical actions such as feedback from the system and selecting problems for a student. The tutorial module also requires communication knowledge for effective communication with students. The user interacts with the user interface (UI) in problem solving and getting feedback from the system. The user interface should be carefully modelled so that it makes learning an enjoyable experience rather than a gruelling and frustrating encounter.

2.2 Web-based ITS

ITSs over the World Wide Web (WWW) offer great prospects in moving closer to the goal of providing personal tutoring experiences to a broader audience. They tend to trim down the limitations and complications encountered in trying to distribute the system to a very large audience. Users could have access to individual tutoring from a standard Web browser.

A Web-based ITS can be deployed according to a number of architectures. The common solutions include Java-only, HTML-CGI and distributed Client-Server [Alpe99]. A Java-only solution would be to create the Tutor as an applet and allow students to download it from a specific URL. The users would interact with HTML entry forms in a Web browser in an HTML-CGI architecture. This consists of a server that possesses total functionality. A client-server model, on the other hand, distributes functionality between a client and a server. This would consist of a downloadable applet that delivers the user interaction module and communicates directly with a server application.

A number of ITSs that are based on the Web have been developed in the last few years. These ITSs cover a variety of pedagogical areas, e.g. Mathematics, Computer Science and Medicine, and are intended for a wide range of ages.

Algebrain [Alpe99] is an environment for practising algebraic skills. The standalone version of the Algebrain's equation-solving tutor was developed according to the architecture depicted in Figure 1. Due to the limitations and complications of trying to make the system available for a large audience, it was enhanced to function as a Web-based ITS.

The Web-based version of Algebrain uses a client-server architecture, as illustrated in Figure 2. The UI of the enhanced version is implemented as a Java Applet that can be executed on a Web browser, as this creates a more interactive user experience than HTML alone could afford. For high efficiency, the applet communicates with the central server using a socket that enables direct communication between the client and server code [Alpe99].

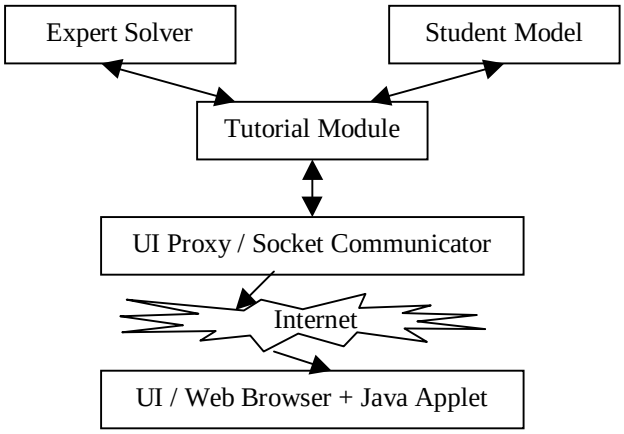


Figure 2: Architecture for the Web-based Algebrain System [Alpe99]

Advanced Distance Education (ADE) is also an ITS, developed to deliver continuing medical education courses over the Web. ADE is mainly aimed at medical professionals in the United States who are periodically required to take continuing education courses in order to maintain their licence to practice. By recording student's actions, the courseware adapts its feedback based on their achievements.

Similarly, SQL-Tutor (Web version) is an ITS that was developed to deliver SQL over the Web, as discussed in the following section.

2.3 SQL-Tutor

SQL-Tutor is an ITS for assisting students in learning the database query language, SQL. The system is aimed at upper-level undergraduates. It is developed as a guided discovery learning environment that provides facilities to verify student's solutions and assist them in solving problems, if required. The system attempts to tailor instructional actions to his/her needs [Mitr98a, Mitr99]. Several aspects of the SQL-Tutor are discussed in the following sub sections.

2.3.1 Domain

SQL is a comprehensive database language consisting of data and view definition statements as well as data manipulation statements. The American National Standards Institute (ANSI) and the International Standards Organisation (ISO) developed the first standard for SQL called SQL1 in 1986. A revised and expanded version of the original standard was adopted in 1992 as a new standard, namely SQL2, which is used in many commercial databases today. Even though there is a movement towards graphical query interfaces, SQL is extremely important in the database world. SQL3, the latest standard, further extends SQL with object-oriented and other recent database concepts [Elma94, Mitr98a].

Students are known to experience problems in learning SQL despite its simplicity and structured nature. Some of these problems come from the burden of having to memorise the database schemas. Students are also known to have difficulty in comprehending concepts such as grouping and restricting grouping. Join conditions and understanding the difference between aggregate and scalar functions are other areas that students struggle with in SQL [Mitr98a].

SQL is usually taught in a classroom environment, complemented by lab exercises. However, students find it difficult to learn SQL by directly working with a DBMS, as the error messages are limited only to syntax errors and typically are not very helpful. SQL-Tutor, on the other hand, is capable of generating better feedback for syntax errors as well as semantic errors [Mitr98a].

SQL-Tutor currently covers only the SELECT statement of SQL. However, this does not prove to be a limitation, as queries are known to cause the most misconceptions for students. Additionally, many concepts covered by SELECT statements are relevant to other SQL statements as well as to relational databases in general [Mitr98a, Mitr99].

2.3.2 Learning environment

SQL-Tutor consists of a user interface, pedagogical module and a student modeller [Mitr98a], as illustrated in Figure 3. The interface for the Web version is given in Figure 4. The main page of SQL-Tutor is divided into four frames. The upper frame displays the problem, allowing the student to easily remind him/herself of the elements requested in the query. The middle left frame consists of the clauses of the SQL SELECT statement, thus eliminating the burden of remembering the keywords and the relative order of the clauses. The lowest part displays the schema for the chosen database, with a link to a description of the currently chosen database and the table descriptions. The table descriptions are shown by their names and attribute names. The primary key is underlined for easy identification. The visualisation of the structure of the tables is important for reducing the working-memory load of the students.

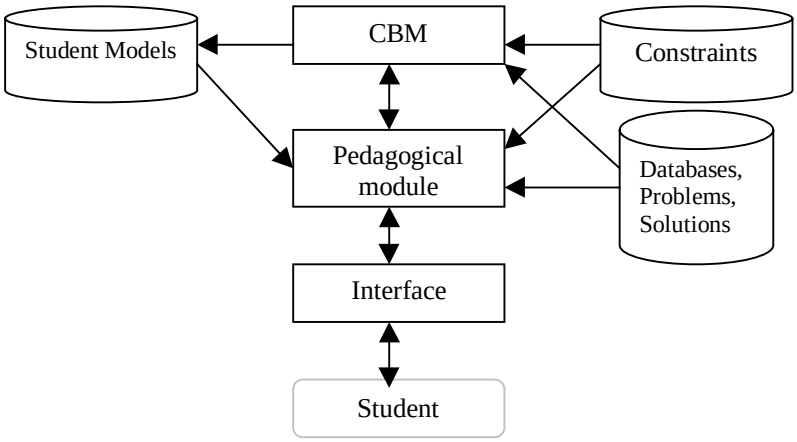


Figure 3: The architecture of the stand-alone version of SQL-Tutor [Mitr98a]

Figure 4: User Interface of SQL-Tutor (Web version)

The pedagogical module (PM) selects problems for the student and generates appropriate feedback. When a solution to a problem is submitted, PM sends it to the student modeller. The student modeller checks whether the solution is correct or incorrect and updates the student model. The feedback is generated and is displayed in the middle right frame of the SQL-Tutor interface.

2.3.3 Diagnosis and Feedback

SQL-Tutor is based on a student modelling approach, called Constraint-Based Modelling (CBM) that focuses on student errors [Mitr98b]. The domain knowledge in CBM is represented as constraints and is used to identify errors. Identification of errors is extremely important for students who are unable to detect errors themselves. The basic assumption of CBM is that the diagnostic information is not in the sequence of student actions, but is in the final state. Thus only the final submitted answer is evaluated for correctness although CBM can also be used in a step-by-step manner [Mitr98b]. This feature increases the efficiency of the system, thus reducing the amount of communication over the Web. The amount of data transfer between the server and the client can be a crucial factor in congested networks.

The student solutions are evaluated by matching them to constraints. The constraints can be categorised into two groups. One group deals with the syntax of the language and the other with the semantics of problems. The semantics are checked by the comparison of the student's solution to the ideal solution. The constraints that are violated are processed by the PM, resulting in the generation of appropriate feedback [Mitr98b].

The level of feedback determines the amount of information provided to the student. The system provides six levels of feedback: **positive/negative**, **error flag**, **hint**, **partial solution**, **all errors** and **complete solution** [Mitr99]. The lowest level of feedback, **positive/negative**, provides only information on whether the solution is correct or incorrect. It also provides the number of errors in incorrect answers. An **error flag** message informs the user about the clause in which the error had occurred (the pedagogical module selects one when the submitted solution consists of more than one error). **Hint** level messages provide more information about the type of the error. **Partial solution** displays the correct content of the clause that contains the error. Hints about all errors in the submitted answer are listed in a message at **all errors** level. The ideal solution is displayed in the **complete solution** level. The system does not provide general help on the syntax of SQL. This could be a drawback of the system, as it does not attempt to help students with the syntax and the concepts of SQL keywords.

When the process of solving a problem is completed, the system consults the student model in choosing a new problem for the user.

2.4 Animated Pedagogical Agents

Pedagogical Agents [Andr97, John98, Lest97a, Pies98, Schö98] are autonomous agents that facilitate human learning by interacting with students in learning environments. They have sufficient understanding of the learning context and subject matter to allow them to perform useful roles in computer-based learning environments. They should also manage their own behaviour in a consistent manner, responding to a variety of environmental stimuli. They must exhibit robust behaviour in rich, unpredictable environments. The environment of an agent includes both the student environment and the learning environment in which the agent resides.

Animated pedagogical agents appear to the student as animated characters. They can engage in a continuous dialogue with the student, and emulate aspects of dialogue between a human teacher and student in instructional settings. They should give the user an impression of being lifelike and believable, producing behaviour that appears to the user as natural and appropriate to the role of a virtual instructor or guide. It is useful to give our agents behaviours that make them appear knowledgeable, attentive, helpful, concerned, etc [John98, Lest97b, Lest99].

A large-scale empirical study of the affective impact of animated pedagogical agents on a student's learning experience was conducted by the North Carolina State University's Multimedia Laboratory [Lest97b, Lest97d]. This study involved one hundred middle school students interacting with **Herman the Bug** (discussed in 2.4.2), an animated pedagogical agent that inhabits the **Design-a-Plant** learning environment. **Design-a-Plant** is a design-centred microworld that provides students with the opportunity to explore physiological and environmental considerations that govern a plant's survival. The study revealed the **persona effect**, which is that the presence of a lifelike character in an interactive learning environment can have a strong positive effect on a student's apprehension of their learning experience [Lest97b]. It also demonstrated

the interesting effect of multiple types of explanatory behaviours on both learning performance and affective understanding. Students appreciated the agent as being helpful, credible and entertaining.

2.4.1 Desirable characteristics

Empirical studies [Lest 97b, Lest 97d] have shown that animated pedagogical agents can enhance the student's quality of learning. There are a number of characteristics that an effective pedagogical agent should display.

Good teachers are often good motivators. As motivation is a key ingredient in learning, pedagogical agents should encourage students to spend more time interacting with the ITS and attempt more problems. Moreover, they should also congratulate users when they successfully solve problems.

Emotions also play an important role in motivation. Therefore pedagogical agents will become more efficient teachers if they appear to 'understand' emotions and respond accordingly. Pedagogical agents should also show that they care about students and their progress [Elli99, Town98a]. This can foster a feeling that the student is not deserted in the learning process and can encourage the student to care about his/her own progress, and the agent's opinion of him/her. Hermann Maurer proposes the use of the "Tamagotchi craze" for teaching purposes. He proposes VR-Friends (virtual friends) who are kept happy if their owners answer questions correctly [Maur98].

The agent should be sensitive to the student's emotions. Whenever the student feels frustrated, the agent should intervene with assistance before the student loses interest. The pedagogical agent should also convey enthusiasm for the subject matter, in order to foster similar enthusiasm in the student [Lest97d].

An agent should also possess a rich and interesting personality, so that it can simply make learning more fun. A student who enjoys interacting with a pedagogical agent will have an increased positive view of the whole learning experience. A student who enjoys a learning environment would undoubtedly spend more time there, which is likely to increase learning [Lest97b].

Animated personas can cause the students to feel that the educational material is less difficult. Pedagogical agents should be visually expressive to clearly communicate problem-solving advice and simultaneously have a strong motivational effect on students. Most importantly, they make it possible to model dialogues and interactions that occur during individual tutoring. Factors such as eye contact, body language and emotional expressions should be modelled and exploited for instructional purposes. An effective agent should posses the ability to increase the student's attention to learning [Lest97d].

2.4.2 Examples

Animated characters in the interface of pedagogical systems have become increasingly popular in the recent years. These characters are based on either cartoon-style drawings, real video or geometric 3D-models [Andr97]. They may either inhabit a virtual world, or a constrained environment that consists solely of the agent. There are many examples of pedagogical agents constructed in research laboratories around the world.

USC/Information Sciences Institute's Centre for Advanced Research in Technology of Education (CARTE) has developed two agents: **Steve** (Soar Training Expert for Virtual Environments) and **Adele** (Agent for Distance Learning – Light Edition).

Steve [Elli99] is designed to interact with students in networked immersive virtual environments. It has been applied to naval training tasks such as operating engines aboard US Navy surface ships. Students can see the agent in stereoscopic 3D and hear him speak. The agent monitor the student's position and orientation in the environment. **Steve** software is combined with 3D display and interaction software, simulation authoring software, speech recognition and generation software to produce a rich virtual environment in which students and agents can interact. **Adele** [John98], on the other hand, was designed to run on desktop platforms with conventional interfaces. The motivation behind the design of Adele was to broaden the applicability of pedagogical agent technology. **Adele** is created to integrate into Web-based learning material and runs on a student's Web browser in a separate window. CARTE is developing Adele-based course material for medical education in family medicine and graduate level geriatric dentistry.

North Carolina State University's Multimedia Laboratory has also developed two pedagogical agents: **Herman the Bug** and **Cosmo**. Both agents were developed to inhabit virtual environments. **Herman the Bug** [Elli99, Lest97a, Lest99] is a 2D cartoon figure that inhabits **Design-a-Plant**, a design-centred learning environment to help middle school students understand botanical anatomy and physiology (Figure 5). This antenna-bearing creature advises students as they design plants to survive in various hypothetical environments. In the process of explaining, he performs a broad range of activities including walking, flying, swimming, shrinking and bungee jumping. . **Cosmo** [Lest97c, Town98a, Town98b], in contrast, is a 3D character that occupies the Internet Advisor, a learning environment for the domain of Internet packet routing. The agent assists students to solve problems such as finding a route avoiding high-traffic to transmit packets between network hosts. **Cosmo** has been used to investigate how to combine various agent behaviours in order to enhance deictic believability [Lest97c, Town98a]. Deictic believability is the ability to refer to objects in their environment through judicious combinations of speech, locomotion and gesture, in a manner similar to humans. **Herman the Bug** has been used to investigate managing mixed initiative dialogues [Lest99]. It was also used in large-scale empirical evaluation studies that have demonstrated the effectiveness of pedagogical agents in facilitating learning [Lest97b, Lest97d].

Figure 5: **Herman the Bug** inhabits **Design-a-plant** environment [Lest97e]

André, Rist and Müller at the German Research Centre for Artificial Intelligence (DFKI) have developed an animated pedagogical agent for interactive WWW presentations, called **PPP Persona** [Andr97]. The persona appears in many forms. Currently there are two cartoon figures and three 3D models. The persona guides the learner through Web-based material using presentation acts (e.g. pointing) to draw attention to elements of the Web pages, and provide commentary via synthesised speech. The PPP system generates multimedia presentation plans for the persona to deliver. **PPP persona** executes this plan adaptively, modifying it in real-time based on user actions such as repositioning the agent on the screen or asking questions.

IBM T.J. Watson Research Centre has developed the Algebrain equation solver [Alpe99] to solve equations for a particular variable. Unlike other mathematical software, Algebrain is not only concerned with the final answer, but supports student's problem solving activities to enhance their problem solving skills. At each step of the solution process the user can ask for hints by clicking on the animated agent. The agent is a dancing 2D cartoon figure. The feedback is given as text combined with animated behaviours of the agent such as applauding.

2.4.3 Types of interactions

Pedagogical agents interact with students in a number of different ways by performing useful pedagogical activities. The behaviour space of an agent should be exploited in a manner that fulfils the objectives of a pedagogical agent.

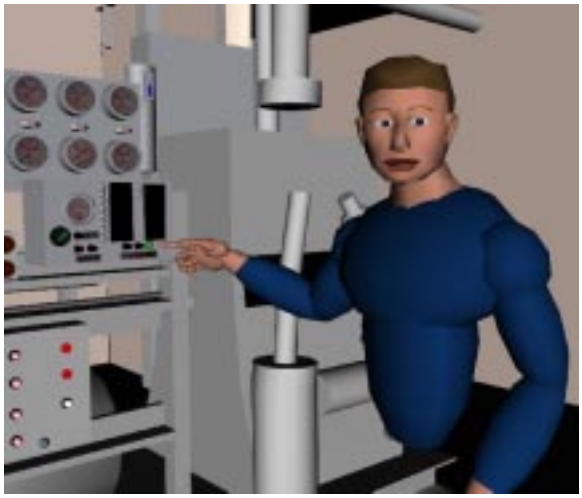


Figure 6: Steve demonstrates how to operate equipment US Navy ship [Steve]

When students are first introduced to a certain topic, it is often necessary to provide a demonstration of how to solve a problem or perform a certain task [John]. Pedagogical agents are well suited for demonstrations, although these alone are not very effective unless the student understands what is being done. The agent should also explain its actions to the student. For example, **Steve** performs demonstrations of how to operate important equipment aboard US Navy ships, and provides explanations in association

with his demonstrations (Figure 6). **Steve** also allows the student to move around in the environment, allowing the student to view the demonstrations from various perspectives [Elli99].

Pedagogical agents should assist students by means of hints to can guide the student struggling with a problem. Implementations such as **Adele**, **Steve**, **Herman the Bug** and **Cosmo** provide such hints [John, John98, Lest97d, Town98a].

Opportunistic instruction (providing instruction when situations arise where it is appropriate) is another type of interaction that pedagogical agents should possess. This feature is important since these instructions provide information that can be used immediately. **Herman the Bug** makes use of opportunistic instructions. When the learner works on specifying the characteristics of a leaf, **Herman** intervenes and provides instructions on the leaf's morphology [Lest97d, Lest99].

Pedagogical agents should be able to generate explanations as required. **Herman** and **Cosmo** do this if the student makes a mistake or seems to have problems [Lest97b, Lest99, Town98a].

Emotive behaviours of agents can help to engage and motivate the learner. They could also relieve student's frustrations by appearing to commiserate with them. **Cosmo** possesses a large emotive behaviour space. Behaviours such as applause are used with congratulatory speech acts. **Cosmo** also uses behaviours such as head scratching, and shrugging when posing rhetorical questions [Town98b] (see Figure 7).

Other capabilities, such as student modelling, that are important for intelligent tutoring systems, are potentially useful for pedagogical agents as well [John].

Figure 7: Cosmo explaining the task [Lest97f]

2.4.4 Architectures

The architecture of a pedagogical agent depends on the range of capabilities that it intends to provide. The current implementations of agents can be categorised into three architectures; behaviour sequence approach, layered generative approach and state machine compilations approach [John].

In behaviour sequencing, behaviours are assembled from a collection of prerecorded primitive animations, sounds and speech elements. These elements are organised in a behaviour space. Useful pedagogical behaviours are generated by a behaviour sequencing engine that composes media elements in real-time. Assembling behaviours from prerecorded segments saves time in creating animations. This approach is well suited for graphics where the camera is fixed. As it does not provide real-time adaptation of behaviour, if the student performs an action during the execution of a sequence, the behaviour sequence will have to be recomputed. **Herman the Bug** was designed according to the behaviour sequencing approach [Lest97a, Lest99].

The layered generative approach generates animations in real-time. The architecture consists of a cognitive decision making layer and a perceptual motor layer that is responsible for monitoring the environment and generating animations. The cognitive layer continually evaluates the state of the environment and makes decisions about the agent's actions. The perceptual motor layer carries out these actions. **Steve**'s architecture is an instance of the layered generative approach [Elli 99].

The state machine compilation approach addresses the issue of real-time adaptation of the agent's behaviour, while limiting real-time animations. This approach is also based on behaviour space in a similar manner to the behaviour sequencing approach. However, these behaviours are executed by a state machine that can adapt at run time to student actions. This approach is exemplified in **PPP Persona** [Andr97].

Chapter 3

Development of the agent

Although shown to be effective [Mitr98a, Mitr99], the original SQL-Tutor can be further improved in order to provide more support for the student, increase their learning rate and enhance the quality of learning. This can be achieved by introducing an animated pedagogical agent to SQL-Tutor.

The following sections outline the deficiencies in the original version of the system and the expected advantages of introducing a pedagogical agent. The design procedure and proposals are detailed in the third section and finally an account of the implementation procedure is included.

3.1 Deficiencies in the original SQL-Tutor

The feedback from the original SQL-Tutor is presented as simple text messages, which appear conforming and monotonous. If students feel that the feedback is dull, they may be discouraged from spending time trying to grasp the concepts behind the messages. Ultimately they may even lose interest in SQL-Tutor, resulting in a decline of the system interaction time. An ITS is ineffective, irrespective of the quality of feedback, if the system is unappealing to the users. Therefore the feedback should be presented in an interesting and lively manner.

Motivation is a key ingredient in effective learning. SQL-Tutor attempts to motivate users by displaying text messages such as “Well done!”, “Almost there”, etc. However, the system does not offer a great variety of motivational messages and the possibilities of motivating the users with pure text messages are limited, as these have very little impact on most humans. Visual messages have been shown to have a much greater effect on learning [Lest97b, Lest97d].

Another aspect of the system that can be improved is its support for self-explanation, which is the process of generating explanations and justifications to oneself when studying an example. Studies have shown that students who self-explain learn more. Moreover, when students are prompted to self-explain, most students will do so and thus increase their learning [Cona99]. Therefore, it is highly desirable for ITSs to encourage self-explanation. The original implementation of SQL-Tutor provides no explicit support for its users to compare their solutions with the hints offered by the system, and justify them.

3.2 Expected benefits of introducing an agent

The introduction of an animated pedagogical agent to SQL-Tutor is expected to assist students in learning SQL in a number of ways. Animated pedagogical agents are known to play an important role in motivation. In SQL-Tutor, a lifelike figure that appears in the background can encourage the user to double check before submitting an answer. The agent could motivate the users to perform at their best and encourage them to spend more time with the system.

The agent's behaviours could show an understanding of the student's emotions, making the users feel that they are not alone in the learning process and that they are 'in things together' with the agent. It could also make the student feel obliged to perform at his/her best so the agent would have a high opinion of them.

The agent should display great enthusiasm for SQL to encourage the student to be similarly enthusiastic. Students with high enthusiasm are likely to spend more time with the ITS. Their eagerness may also drive them to learn and master a greater number of concepts using SQL-Tutor.

One of the most important requirements in teaching is that the students should not become frustrated. Once frustration creeps into the student's mind, overcoming it takes a major effort on both parties (teacher and student). An animated pedagogical agent that appears to monitor every action of the user can reduce the risk of users feeling frustrated by the system. The agent should intervene by offering assistance before the student loses interest.

The formal and monotonous image of SQL-Tutor will be changed by the introduction of an animated character, whose rich and interesting personality should make learning more interesting and fun.

The animated character can be used to model dialogues; the interactions that arise during individual human teaching of SQL. To increase the student's attention during the learning process, the agent's visual behaviours can be utilised to clearly communicate advice from the system to the user. An agent that possesses the ability to hold a high proportion of the user's attention, and delivers clear advice, could even make the student feel that SQL is less difficult.

3.3 Design

The design process of the pedagogical agent for SQL-Tutor consists of a number of steps:

1. Identifying types of interactions between the pedagogical agent and the student and designing the agent's architecture
2. Defining the gesture space
3. Defining how gestures are combined to form behaviours
4. Defining a set of rules for presenting behaviours depending on the interactions

3.3.1 Architecture

The agent will present all messages from SQL-Tutor to the user. This requires a more interesting manner of presenting the text outputted from the system. To fulfil this objective, the agent is designed to perform illustrative gestures while speaking or explaining. As the system only produces text messages depending on the actions of the user, building an agent with an interesting personality is challenging. This was achieved by utilising the gesture space of the pedagogical agent.

SQL-Tutor stores student actions in the form of student models and logs and produces feedback depending on them. Thus, the agent is not required to maintain a history of the student's actions. However, the designed agent records its own past actions, so that it can be used to compose future actions and to ensure that the agent's behaviours do not appear monotonous.

The agent does not produce the text of a feedback message, relying on SQL-Tutor server to produce individualised messages to each user. Thus, the agent should create and maintain a communication link between the central server and itself. Moreover, the agent is also required to get information about the actions performed. This communication link is very important and should be consistent with the user's actions, as the agent relies solely on the messages to synchronise itself with the problem-solving environment of SQL-Tutor. The agent is not required to send any messages to the server because it does not alter the problem-solving environment.

3.3.2 Gestures

One of the initial phases in developing an animated pedagogical agent for an ITS includes identifying interactions between the user and the agent, as these determine the gestures that the agent should perform. Gestures are atomic behaviours that define the capabilities of the animated character. These gestures are combined to compose behaviours that determine the interactions between the agent and the user.

It would be most desirable to have an animated character for SQL-Tutor with a rich and interesting personality. This is achieved by designing an agent that possesses a vast array of gestures, offering greater flexibility in composing behaviours; the basis of an interesting personality. The agent's gesture space also comprises amusing gestures that make the learning experience more fun.

The agent's collection of gestures includes believability enhancing gestures such as blinking and breathing. This group of gestures makes the agent more lifelike, which makes the user feel that an 'alive individual' on the desktop is aware of each action they perform. The users may feel obliged to perform at their best so that the pedagogical agent approves of them.

One of the greatest advantages in introducing an agent is the possibility of improving student performance in response to the agent's display of emotions such as happy, sad or even acting surprised and confused. Gestures such as acting happy or laughing can be used to improve the effectiveness of a congratulatory behaviour. Mixing emotions with feedback messages can create the illusion of a human tutor being present.

The agent's repertoire of gestures also includes those that enable it to clearly communicate advice to the user, e.g. pointing and looking. These gestures can be used quite effectively to focus the student's attention to a certain area in the interface. The agent for SQL-Tutor can employ these behaviours to introduce the user interface of the

system to a new user and even point out the errors made in a submission. It may also present hints verbally to the student.

3.3.3 Pedagogical behaviour space

The pedagogical agent for SQL-Tutor should display a vast array of behaviours that offer useful instruction to the student. Each behaviour is a believable sequence of gestures, e.g. explain and point, smile and speak. These behaviours of the agent should present feedback from the system in a more appealing manner. Its behaviours should be synchronised with SQL-Tutor and should depend on the actions of the user. Moreover, when composing gestures, the duration of each has to be optimised for the students to achieve high problem solving efficiency.

The pedagogical behaviour space of the agent consists of three main categories: **introductory**, **explanatory** and **congratulatory** (see Figure 8).

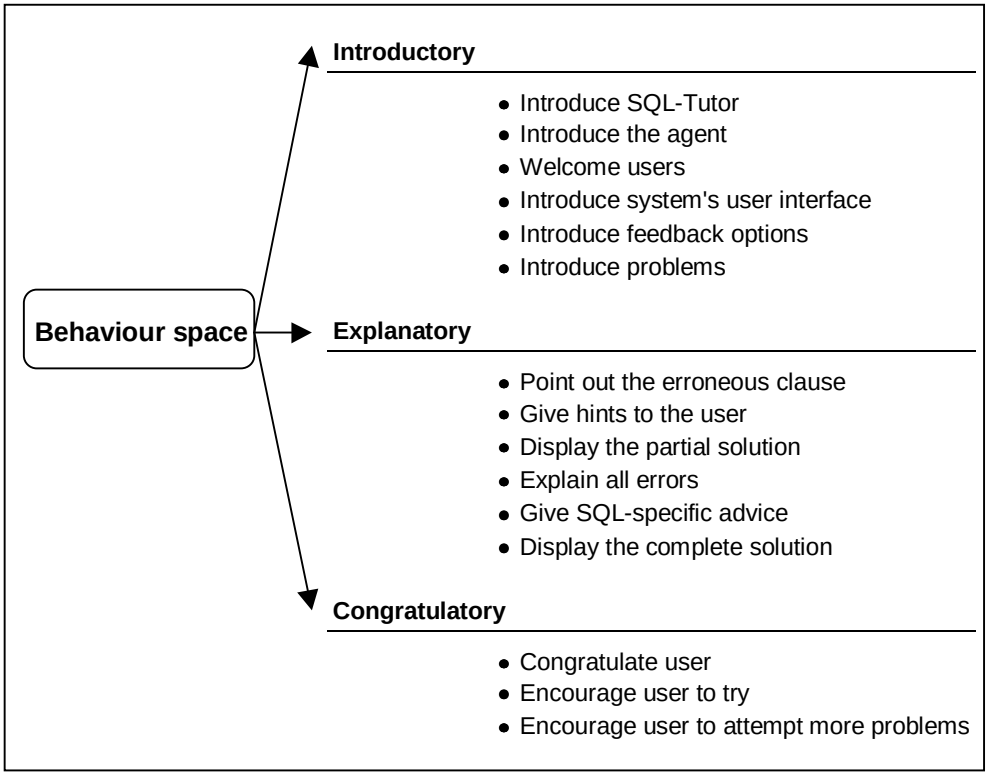


Figure 8: Behaviour space of the pedagogical agent for SQL-Tutor

The **introductory** behaviours mainly focuses on new users, e.g. introducing the user interface of SQL-Tutor and describing the levels of feedback, etc. Gestures such as pointing are used as visual aids in presenting introductory messages. The agent also displays behaviours that welcome the user to the system to grab the user's attention at the start of a session. Introductory behaviours are also used to introduce new problems.

Although the introductory behaviours of the agent for SQL-Tutor are predefined, this feature is unlikely to result in an agent with a dull personality because the behaviours are presented to the student only once in a session. However, introduction of problems should have an adaptive behaviour. As the problems are self-explanatory, the agent only looks at the text of the problem to attract the attention of the student to it. This simple behaviour avoids distracting the user.

Explanatory behaviours incorporate hints from SQL-Tutor. The agent adapts its behaviour depending on the feedback from system. The challenging task is to present hints from SQL-Tutor in an interesting manner. General information and help with SQL keywords such as join and nested selects, which are known to pose problems for most students, should also be provided. Another useful area of explanatory feedback is to offer the student general guidelines on generating queries, particularly for novices. The agent also presents different levels of feedback, e.g. hints, partial solutions and complete solutions. Each level has a particular set of gestures in order to distinguish between the levels of feedback. The variety of levels increases the effectiveness of the instruction/hint, and results in a more interesting personality for the agent.

Explanatory behaviours are the most important, because SQL-Tutor is a problem-solving environment. As the agent will have to repeatedly perform behaviours such as presenting hints to the students, it also needs variety between consecutive presentations to maintain interest. The designed agent is equipped with at least three distinct predefined behaviours for each feedback level. Behaviour is selected from the available set to present the respective messages from SQL-Tutor. The duration of these behaviours depends on the length of the text message.

An important aspect of teaching is motivation. Studies [Lest97b, Lest97d] have shown that animated pedagogical agents can be effectively used to motivate students in problem solving environments. The agent for SQL-Tutor should display **congratulatory** behaviour on a correct submission to try to make the user feel delighted about getting the problem correct. The agent attempts to obtain maximum effectiveness from these behaviours by performing one of the three defined congratulatory behaviours. On the other hand, the agent displays disappointment with an incorrect submission to motivate users to try harder in the next attempt. The motivational behaviours should be configured in such a way that they make learning more fun, so that a user's interest in the system would grow and the system would have a better opportunity to improve the student's problem solving ability.

3.3.4 Rules for presenting behaviours

The designed agent follows a predefined set of rules in choosing a behaviour from its behaviour space. The selection procedure is primarily based on the student's interactions with SQL-Tutor. The interactions of a student during a problem solving session consists of three distinct stages. Firstly, the student logs on to SQL-Tutor. He/she then solves problems supplied by the system. Finally, the student logs out from the system at the end of the session. The behaviours of the agent are triggered depending on the user's actions.

The behaviour of the agent at login depends on whether the student is a new user to the system or has been previously exposed to it. The agent performs behaviours that introduces SQL-Tutor and its user interface, etc. for new users. On the other hand, the agent only performs a short greeting behaviour for users who have had previous exposure to the system.

Most of the behaviours performed by the agent during problem solving incorporate the feedback messages from SQL-Tutor. The agent possesses a number of behaviours for each level of feedback, which can be used to present the respective messages. It relies on a set of behaviour selection rules to pick the ideal behaviour from its behaviour space. The selection is purely based on the level of feedback (see Section 2.3.3 for details on feedback levels). This selection results in a random choice of three candidate behaviours.

The agent also attempts to intervene if the user is experiencing problems with the user interface. Thus, if the user is supplied repeatedly with the same feedback message, the agent intervenes and offers advice on how to get different levels of feedback from the system. Moreover, the agent presents a set of guidelines that can be followed to solve queries for users who repeatedly request the complete solution of a problem. These behaviours are activated by the behaviour selection rules during the selection procedure, when the recently received message is compared with previously presented messages.

Logging out from the system triggers the agent to present a logout behaviour. The main goal of this behaviour is to encourage the user to interact with the system again in the future.

3.4 Implementation

Adele [John98] is an animated pedagogical agent, which was designed to support students working through problem-solving exercises that are integrated into instructional materials delivered over the Web. The animated persona of Adele was chosen to facilitate the implementation of the animated pedagogical agent for SQL-Tutor.

The steps in the implementation of the agent were as follows. An appropriate character was selected, then the gestures were implemented by creating the respective frames and modifying the applet persona. The required behaviours and the behaviour presentation rules were implemented. Finally, the applet persona was incorporated with SQL-Tutor.

3.4.1 Adele

Adele's system consists of two main components: the pedagogical agent and the simulation. The pedagogical agent consists further of two sub-components, the reasoning engine and the animated persona. Adele has been adopted for a case-based clinical diagnosis application, where this is used to highlight interesting aspects of the case, and monitor and give feedback as the student works through a case (see Figure 9).

The animated persona is simply a Java applet that runs on a Web browser. As SQL-Tutor also runs on a Web browser, this component of Adele can be incorporated with SQL-Tutor, and developed into an animated pedagogical agent for it. An added advantage is that the animated persona and the SQL-Tutor user interface can coexist on the same desktop in two separate windows[1].

The agent that was developed for the case-based medical education system is capable of performing a number of gestures. Adele's gestures are produced using two-dimensional drawings and her original gesture space consists of a total of sixteen

[1] We wish to acknowledge the Centre for Advanced Research in Technology for Education for providing the source code for the applet persona of Adele.

gestures. These vary from simple believability enhancing gestures such as blinking, to emotive gestures such as smiling, acting confused, acting surprised, etc. The agent also possesses gestures that can be used in presenting advice and explanations, such as look, point and speak. The agent could also be used to simulate congratulatory behaviours by smiling, nodding, and an animation that checks off items on a clipboard. Other gestures of Adele involve casual advice gestures such as speaking.

Figure 9: Adele introducing herself

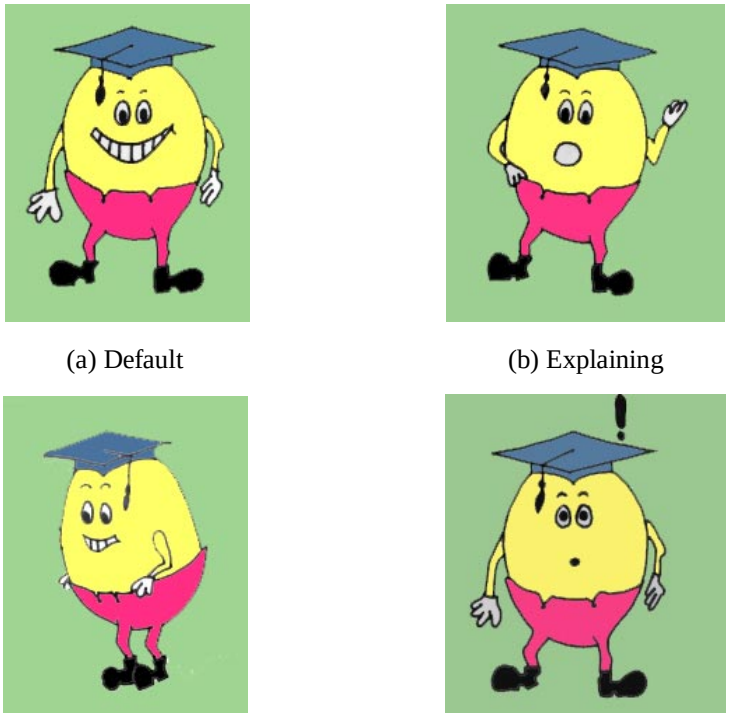
3.4.2 Character

The foremost design task was to choose an appropriate character as the pedagogical agent for SQL-Tutor. As no obvious SQL character could be found, a wise, elderly figure was considered (see Figure 10), who would be able to display a vast knowledge of SQL.

Figure 10: Wise-person character

However, one of the main objectives of introducing an agent is to make learning more fun, and the option of a wise figure appears slightly grim, clearly limiting the user's enjoyment. The option also has an immensely complicated face. As a fully expressive agent requires a large number of frames, sketching the frames would take an artist and would be a huge task.

Hence, Figure 10 is not an ideal choice, so a simpler and more humorous approach was taken. The new character was named **Smart-egg**[2] (see Figure 11). To build a fully functional agent approximately fifty frames had to be sketched. The applet persona of Adele swaps frames and uses techniques such as morphing to perform animations. Each gesture uses about three to six frames to perform the relevant animation.



(a) Default (b) Explaining

(c) Looking down (d) Surprised

Figure 11: Frames of the **Smart-egg** character

3.4.3 Gestures

The host of animated gestures offered by Adele's applet persona is not sufficient for **Smart-egg**. Accordingly, a gesture that displays sadness was created and added to the repertoire of **Smart-egg**'s gestures. This gesture is composed of three frames and is used to display **Smart-egg**'s emotions in situations such as logout.

[2] We wish to thank Nenad Govedarovic for providing the initial drawing of the character.

As Adele was developed for the medical domain, some of her gestures are not applicable for the domain of SQL-Tutor. For example Adele carries a clipboard, appropriate for a doctor character, but not be ideal in this case. These gestures were not adopted for **Smart-egg**.

The gesture space of **Smart-egg** is detailed in Figure 12.

Gesture space of Smart-egg

- Default
- Smile
- One hand back
- Both hands back
- Show left palm
- Speak
- Explain
- Nod
- Disapprove
- Surprised
- Confused
- Sad
- Look
- Point

Figure 12: Gesture space of **Smart-egg**

3.4.4 Architecture

The Web version of SQL-Tutor was developed using a Common Lisp (CL) HTTP server. The user interacts with the server by performing an action on the Web page. After an action is performed, the user's Web browser sends a corresponding message to the CL HTTP server. The server then evaluates the state and creates a new page that results from the user's action. This page is then sent to the user's Web browser where it is displayed. This process is performed whenever the user carries out actions which result in a change in environment, such as submission of an answer or choosing a new problem or database.

The pedagogical agent's Java applet and the server are required to exchange messages in order for the agent to receive messages and know the actions of the user. This was achieved by implementing a Java socket connection between the server and the applet. The developed agent consists of a dedicated thread of execution that waits to receive messages from the server. Whenever this thread receives a message, it processes it by first selecting the appropriate behaviour of the agent by using the behaviour selection rules. It then orders the animated persona to carry out the chosen behaviour.

The architecture of the system that includes SQL-Tutor and the pedagogical agent is depicted in Figure 13.

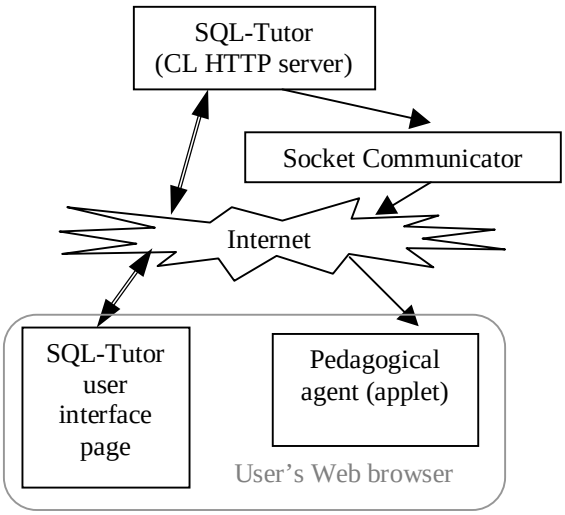


Figure 13: Architecture of SQL-Tutor with pedagogical agent

The **Smart-egg** agent inhabits a separate window beside SQL-Tutor. Some examples of the agent's behaviours are featured in Figure 14.

(a) introduction of **Smart-egg** (b) congratulating a user

Figure 14: Behaviours of the pedagogical agent, **Smart-egg**

Chapter 4

Evaluation study

It is important to assess the contribution that **Smart-egg** makes to SQL-Tutor. Thus an experiment was carried out with second year Computer Science students to evaluate the contribution **Smart-egg** makes to learning. Answers were sought to the following questions:

- Can the persona effect be observed? (i.e. does the presence of **Smart-egg**, have a strong positive effect on a student's perception of their learning experience?)
- Can **Smart-egg** increase the student's learning effectiveness and/or quality of learning?
- By creating the illusion of life, can the captivating presence of **Smart-egg** motivate students to solve more problems and/or with fewer errors?
- Can **Smart-egg** contribute to the student's problem solving abilities or does its activities interfere with problem solving?
- How do the students perceive **Smart-egg's** problem-solving advice?
- To what extent do the students find the feedback helpful and clear?
- Would students prefer to be assisted by **Smart-egg** or to be left alone?

The following section outlines the design of the experiment. The chapter also includes the results and their analysis, and discusses the evaluation study.

4.1 Experiment design

An experiment was designed in which students interacted with two versions of SQL-Tutor. Participants in the evaluation were students who were enrolled in the second year Software & Database Design (COSC 205) course in the Computer Science department of the University of Canterbury. Students were randomly assigned to interact with two versions of SQL-Tutor: with **Smart-egg** (agent group), and without **Smart-egg** (as the control group). The study was conducted in the computer laboratories of the department. Participation was anonymous. There were 26 students in the two groups.

Data was collected over three days during normal lab times and the students chose workstations at random. Each data collection session proceeded in four distinct phases: pre-testing, system interaction, post-testing and system assessment (illustrated in Figure

15). To assess the student's knowledge of SQL before and after interacting with SQL-Tutor, students were asked to take pre- and post-tests on a Web page. These consisted of three multiple-choice questions, with corresponding questions of each test being of similar difficulty (copies of the pre- and post-tests are given in Appendix A). Each question was carefully constructed to evaluate the student's knowledge of SQL. Students were asked how confident they were of their answer after each question to get a measure of their confidence in SQL. Out of the students who participated in the study, 23 students submitted answers to the pre-test; the remaining three submitted empty tests.

When the student had submitted the pre-test, he or she was left to interact with a version of SQL-Tutor. The students used the system in a two-hour session. However, the system interaction times varied depending on student preferences. They were free to choose problems and determine the path of their learning experience. Each action of the student was recorded automatically by the system in the form of student logs. Following the end of the session, they completed the post-test. Only four students completed and submitted post-tests. The remainder of the students had quit the system without logging out.

Finally, the students were asked to complete a system/agent assessment questionnaire, which is given in Appendix B. To reduce biased responses, students were strongly encouraged to record all their responses, because the researchers wanted to use these to improve the system. Students were also given complete privacy as they completed the questionnaire. Each questionnaire consisted of sixteen questions, where most relevant to this study[3]. These questions contained five response categories (1 to 5) and asked them to record free-form responses. Out of the total of 26 who participated in the study, 22 students completed questionnaires.

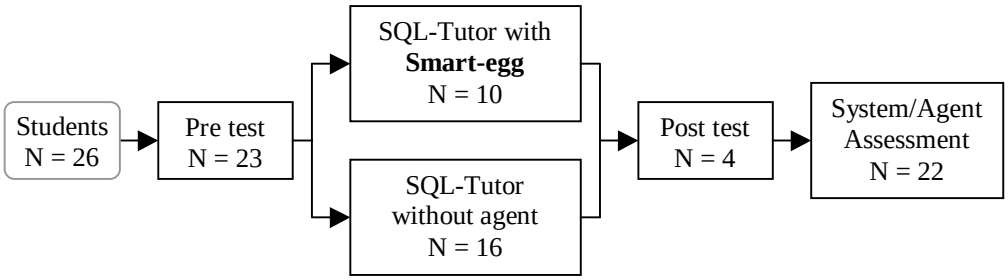


Figure 15: Experiment outline

4.2 Results and analysis

Data from the evaluation study were collected from three sources: system/agent assessment (questionnaire), problem-solving logs, and pre- and post-tests. The data from these sources were subjected to subjective, objective and performance comparison analyses respectively.

[3] This study was a subset of the study carried out to evaluate SQL-Tutor and its other improvements.

4.2.1 System/agent assessment

The analysis of the responses to the user questionnaires revealed that the students liked **Smart-egg**. When asked to rate how much they enjoyed the system on a scale of 1 (not enjoyable) to 5 (very enjoyable), the average rating from the students who used the agent was 4.5, compared with the control group rating of 3.83 (Table 1). The range for the group using the agent varied from 3 to 5, and the control group range varied from 2 to 5. The majority (60%) who used the agent chose option 5, whereas only 33% of the control group chose option 5 (detailed results of the questionnaire are in Appendix C).

Both groups were equally comfortable with the interface. Of the students who were using the agent, 60% reported that they needed less than five minutes to learn the system interface. Similarly, 54% of the students in the control group had spent less than five minutes learning the functionality of the interface. When asked to rate the ease of using the interface on a scale of 1 (very hard) to 5 (very easy), the agent group averaged 4.1, whereas the control group averaged 3.73. The responses of the agent group varied from 3 to 5, and the control group varied from 2 to 5. Only 20% of the students who used the agent chose option 3 for ease of using the interface, whereas 42% of the students in the control group chose option 3 or less. It can be concluded from these results that students who used the agent found it easier to use the interface.

The students were also asked to rate the amount learnt from the system on a scale of 1 (nothing) to 5 (a lot). Both groups chose similar values; the average rating for the group with the agent was 3.8 and for the control group was 3.92.

	Agent group	Control group
Enjoyment rating	4.50	3.83
Time to learn interface (min)	11.00	10.83
Ease of using the interface	4.10	3.73
Amount learnt	3.80	3.92
Usefulness of feedback	4.80	4.09

Table 1: Mean responses for the two groups

When asked to rate the usefulness of feedback on a scale of 1 (useless) to 5 (very useful), the average response for the agent group was 4.8 and for the control group was 4.09. The majority (80%) of the students who used the agent rated the system as very useful (option 5), and only 42% of the control group chose the same option. As both versions of the system presented the same problem-based messages, it is clear from these findings that the students who used the agent found it easier to comprehend the feedback from the system.

In order to find the statistical significance of the differences in the mean ratings, a hypothesis test was then carried out to find the confidence levels of stating whether the agent group ratings were higher than the control group or vice versa (Table 2). It can be stated, with 93% confidence, that students using the agent rated the system as more enjoyable and, with 97% confidence, that they rated feedback as more useful. Moreover, students who used the agent found it easier to use the interface (with 68% confidence). The differences of other categories are statistically insignificant.

Students in the group who used the agent had written positive comments about the agent. The positive comments specific to **Smart-egg** were:

- "I liked the Smart egg."
- "The Smart-egg is great."
- "I enjoyed the maniacal smile of the egg."

Other students had general comments on the system:

- "Good problem practice with feedback."
- "Liked hints and feedback in particular."
- "Very helpful in learning SQL."
- "Interactive."
- "Great getting feedback."
- "Feedback was excellent."
- "Very easy to use."
- "Informative."

However, one student did say, "I didn't like the egg". This comment may have been influenced by the fact that he had experienced three system crashes during a half an hour session. In short, the majority of the students liked interacting with **Smart-egg**.

	Enjoyment	Time to learn Interface	Ease of using the interface	Amount learnt	Usefulness of feedback
Agent group					
Mean	4.50	11.00	4.10	3.80	4.80
Standard deviation	0.71	10.22	0.74	0.79	0.42
Control group					
Mean	3.83	10.83	3.73	3.92	4.09
Standard deviation	1.03	9.25	1.01	0.67	1.04
Z	1.79	0.04	1.00	-0.37	2.15
Confidence level that hypothesis is correct	**93%**	**3%**	**68%**	**29%**	**97%**

Table 2: Confidence levels of stating the agent group ratings were better than the control group ratings or vice versa

4.2.2 Problem-solving logs

As explained earlier, each action performed by a student during interaction was recorded in a log, along with a time-stamp. This was used to analyse each student's learning. Appendix D gives detailed results of the various kinds of analyses performed on student logs, while Table 3 presents a summary.

The first thing to be analysed was the effect of the agent on the overall interaction time. The students who used the pedagogical agent spent 55.9 minutes on average

interacting with the system, and the control group spent 49.6 minutes on average. It is evident from the results that the students who used the agent were more willing to spend time interacting with the system. Thus, the agent motivated the students to interact with the system for a longer period.

As the group using the agent spent more time with the system, they attempted and solved more problems. Out of the average 14 problems attempted by the students who used the agent, they managed to successfully solve an average of 11.6 problems. The control group only attempted 11.56 problems on average and solved an average of 10.94 problems correctly. Students who used the agent attempted 2.44 more problems on average and solved an average of 0.66 more problems (see Appendix D for details). Taking only the number of practice problems attempted and solved, it can be assumed that students who used the agent learnt more compared with the control group.

Moreover, the group who used the agent took fewer attempts on average to solve problems, compared with the control group. The group with the agent required 30.90 attempts on average, whereas the control group required 32.56 attempts. This was slightly reflected in the average number of attempts taken to solve a single problem. The group with the agent required 2.87 attempts on average to solve a problem, and the control group required 2.78 attempts per problem. These results suggest that the students who used the agent understood the feedback more than the control group, although these results may be affected by the knowledge of the students in the group. However, the problems that were solved in the first attempt are similar for both the groups (an average of 5.1 for the group with the agent and 4.56 for the control group), and as the students did not get any direct help from the system in solving these problems, this observation suggests that students in both groups have a similar knowledge of SQL. Moreover, students in both groups required a similar number of attempts to solve problems that could not be solved on the first attempt (the system gave problem-specific hints for these problems).

	Mean		**Standard deviation**	
	Agent group	**Control group**	**Agent group**	**Control group**
Total interaction time (mins)	55.90	49.63	17.30	26.70
No. of attempted problems	14.00	11.56	5.27	6.49
No. of solved problems	11.60	10.94	4.35	6.36
Total no. of attempts taken to solve the problems	30.90	32.56	14.13	23.97
Problems solved on the first attempt	5.10	4.56	2.60	2.73
Problems solved per time (prb/min)	0.22	0.27	0.07	0.21
Attempts per solved problem	2.78	2.87	1.23	1.33
Attempts to solve problems that could not be solved on the first attempt (attempts/prb)	2.90	2.91	1.61	1.34

Table 3: Mean interaction details

The average number of attempts taken to solve problems that could not be solved on the first attempt was very similar. The agent group required on average 2.90 attempts and the control group 2.91 attempts. This dimension is a good measure of the usefulness of the feedback content in problem solving. As both versions of the system (with or without

the agent), were offering the same feedback content, it is obvious why students of both groups required the same number of attempts.

The interaction times of the agent group varied from 32 to 80 minutes, with a standard deviation of 17.3. On the other hand, student interaction times of the control group varied from 4 to 90 minutes, with a standard deviation of 26.7. Similarly the standard deviations of the control group was higher than the agent group for all but one dimension (attempts to solve problems that could not be solved on the first attempt). As the students were randomly assigned to these groups, it can be seen from the results that the agent affects a wider audience, and does not target a specific group of students.

In order to establish the effect of the agent on the student's learning over time, we plotted the average number of attempts taken to solve the n^{th} problem for each group (illustrated in Appendix D). To reduce individual bias, the problems solved by less than 50% of the participating population were discarded (Figure 16). Although no trends for each group can be seen, the students who used the agent required, on average, 0.2 fewer attempts to solve each problem compared with the control group.

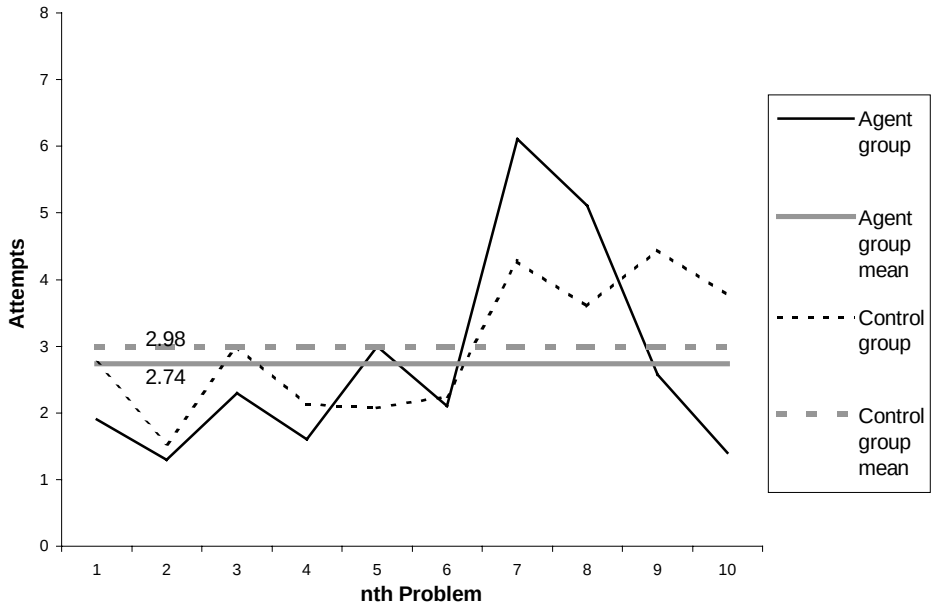


Figure 16: Mean number of attempts taken to solve the n^{th} problem

The number of problems successfully solved per unit of time was similar for both groups. Students who used the agent recorded on average 0.27 correct answers per minute and the control group managed 0.22.

According to the hypothesis test, it can be stated with 53% confidence level that the total interaction time of the group with the agent was higher than the control group (Appendix D). Moreover, they attempted more problems than the control group (with 71% confidence level). Nonetheless, students who used the agent solved fewer problems per minute in comparison with the control group (with a confidence of 63%).

4.2.3 Pre- and post tests

Most students involved in the study participated in the pre-test. Nine students out of ten from the group who used the agent and fourteen out of sixteen in the control group submitted valid pre-tests. These students scored equally well on average in the pre-test. The first, second and third questions of both tests were allocated 1, 5 and 1 marks respectively. The students in the agent group scored higher on questions one and two but scored lower on the final question in comparison to the control group (see Appendix E for details). They had a mean total score of 3.56 and the control group had 3.36 (Table 4). As the difference between the scores is low, it is statistically insignificant (only 41% confidence).

When asked to rate their confidence level of their answer, on a scale of 0 (not confident) to 2 (very confident), students of both groups has similar ratings. The students who were assigned to use the agent were less confident about their answers for questions one and two but were more confident about the final question in comparison to the control group. The mean total rating of students who were later assigned to the agent group was 3.00 and the mean total rating of students who were assigned to the control group was 3.25.

		Agent group	Control group
Scores			
	Q1	0.33	0.14
	Q2	2.56	2.50
	Q3	0.67	0.71
	Total	3.56	3.36
Confidence level			
	Q1	1.22	1.38
	Q2	0.67	0.79
	Q3	1.11	1.08
	Total	3.00	3.25

Table 4: Mean scores for the pre-test

Although there was high participation for the pre-test, only four students from both groups sat the post-test. Three of these students had used the agent. A definite increase in their performance and their confidence can be seen (see Appendix E for details). Students in the agent group have scored higher in the first two questions in comparison to the control group. Students in both groups had scored zero for the final question. The mean total score of the agent group was 4.33 and the control group was 2. Mean total confidence ratings for the agent group was 4.33 and the control group was 4.

However, as there was only one student in the control group who sat the post-test, unbiased comparisons on the mean performances can not be made.

4.2.4 Summary of results

The study demonstrates that animated pedagogical agents can yield important educational benefits. Students in the agent group enjoyed interacting with SQL-Tutor more than the control group did and perceived that the feedback from the agent was more useful. Students also found it easier to use the SQL-Tutor interface when the agent was present.

The motivational effect of the agent was also evident from the problem-solving logs. Students who used the agent spent more time interacting with the system compared with the control group. Moreover, they attempted and solved more problems on average. However, students using the agent solved fewer problems per unit time. Another significant finding was that standard deviations were higher in almost all analyses in the case of the control group, thus exhibiting that **Smart-egg** affects a wider audience and does not target a specific group of students.

It is evident from the pre-test results that the student population had similar knowledge of SQL and were equally confident. No conclusions can be drawn from the post-test scores, as the participation was very poor.

4.3 Discussion

The findings of the study are encouraging. The ratings for the group who used the agent were significantly higher than the control group. They established that the presence of an animated pedagogical agent has a strong effect on the student's perception of the learning environment (referred to as persona effect in [Lest97b]). It was further evident from the problem-solving logs that the students were better motivated by the agent.

The positive ratings for the agent are not likely to have been caused by biased responses as the students were randomly assigned to each group and the anonymity of participants was preserved. Moreover, the students were encouraged to provide their honest opinions in order to improve the agent, and privacy was granted during data collection. However, it should be noted that the numbers involved in the study were small.

The fact that the students who used the agent were solving fewer problems per unit of time may be a consequence of the system running on old computers. SQL-Tutor with the agent requires more processing and memory compared to the system for the control group.

A further study should not be held in the last week of lectures, as many students chose not to participate.

Participation for the post-test could be improved by requesting students to explicitly log out from the system as they leave. Moreover, in the event of a system crash, students were automatically logged out and they were not given the post-test afterwards. This can easily be amended by checking the student's log when giving a post-test.

The length of the evaluation study is insufficient to make solid claims of an improvement in student performance. This study should be extended over a number of weeks, so that a better measure of the student's performance could be monitored.

However, as the initial results are very promising, the pedagogical agent can be extended from simply presenting feedback to supporting the student when learning by self-explanation. This support can be in terms of dialogues with the student, where the agent prompts questions to guide the students. The ultimate pedagogical agent should be able to answer all the questions, within the subject domain, posed by the students.

Chapter 5

Conclusions

Animated pedagogical agents are lifelike creatures used in Intelligent Tutoring Systems to increase student's motivation and enhance their quality of learning. They provide timely, customised advice to support student's problem solving. Their strong visual presence can also increase the student's enjoyment of their learning experiences.

This project developed an animated pedagogical agent, **Smart-egg**, to improve the feedback presentation of SQL-Tutor. **Smart-egg** is a cartoon character that performs animations using two-dimensional frames. It combines its gestures to form complex behaviours that present feedback messages. **Smart-egg**'s main role is to motivate students and subsequently increase their learning effectiveness.

Designing **Smart-egg** involved identifying the types of interactions between the agent and the student and designing the agent's architecture. **Smart-egg**'s gestures space, behaviours space and a set of rules for presenting these behaviours were defined and implemented. The implementation of **Smart-egg** was facilitated by the animated persona of Adele [John98]. Finally, **Smart-egg** was incorporated with the Web version of SQL-Tutor by implementing a Java socket connection between the agent applet and the server.

The evaluation study, carried out with second year students to investigate the effectiveness of **Smart-egg**, produced very promising results. It revealed that students who used the agent perceived SQL-Tutor as more enjoyable and helpful in comparison to the version of SQL-Tutor with no agent. The study also revealed that **Smart-egg** motivated students to spend more time interacting with the system and to attempt more problems. Moreover, the study showed that **Smart-egg** does not target a specific group of students.

This work is a promising first step towards developing an effective pedagogical agent for SQL-Tutor. The pedagogical agent can be improved by guiding self-explanation while the student is learning from examples. This can be implemented in terms of dialogues between students and **Smart-egg**. The pedagogical agent can be further improved by enabling the agent to process natural language. Consequently, students may query the agent whenever they incur problems.

Acknowledgements

Thanks to Antonija Mitrovic for all her support as a supervisor, Jane McKenzie for her helps with technical writing and Kurt J Hausler for help with the implementation of SQL-Tutor. Special thanks to Centre for Advanced Research in Technology for Education for providing the source code for the applet persona of Adele and Nenad Govedarovic for providing the initial drawing of Smart-egg.

Bibliography

[Alpe99] Sherman R. Alpert, Mark K. Singley and Peter G. Fairweather. Deploying Intelligent Tutors on the Web: An Architecture and an Example. *International Journal of Artificial Intelligence in Education*, pp. 183-197, 1999

[Andr97] André Elisabeth, Rist Thomas and Müller Jochen. WebPersona: A Life-Like Presentation Agent for Educational Applications on the World Wide Web. *Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society*, Kobe, Japan, 1997

[Cona99] Conati Cristina and VanLehn Kurt. Teaching meta-cognitive skills: implementation and evaluation of a tutoring system to guide self-explanation while learning from examples. *Artificial intelligence in Education*. S.P. Lajoie and M. Vivet (Eds). IOS Press, pp. 297-304, 1999

[Elli99] Elliot Clark, Rickel Jeff and Lester James C. Lifelike Pedagogical Agents and Affective Computing: An Exploratory Synthesis. *Artificial Intelligence Today, Lecture Notes In Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, Special Volume 1600, M. Wooldridge & M. Veloso (Eds.), pp. 195-212, Springer-Verlag, Berlin, 1999

[Elma94] Elmasri Ramez and Navathe Shamkant. *Fundamentals of database Systems (2nd ed.)* Benjamin/Cummings, Redwood, CA, 1994

[John] Johnson W. Lewis. Pedagogical Agents. http://www.isi.edu/isd/carte/ped_agents/pedagogical_agents.html

[John97] Johnson W. Lewis and Shaw Erin. Using Agents to Overcome Deficiencies in Web-based courseware. *Proceedings of 8th World Conference of AIED*, Japan, August 97

[John98] Johnson W. Lewis, Shaw Erin and Ganeshan Rajaram. Pedagogical Agents on the Web. *Workshop on WWW-based Tutoring, ITS '98*, San Antonio, Texas, 1998

[Lest97a] Lester James and Stone Brian. Increasing Believability in Animated Pedagogical Agents. *Proceedings of the First International Conference on Autonomous Agents*, pp. 16-21, California, February 1997

[Lest97b] Lester James, Converse Sharolyn, Kahler Susan, Barlow Todd, Stone Brian, and Bhogal Ravinder. The Persona Effect: Affective Impact of Animated Pedagogical Agents. *Proceedings of CHI '97*, pp. 359-366, Atlanta, March 1997

[Lest97c] Lester James, Voerman Jennifer, Towns Stuart and Callaway Charles. Cosmo: A Life-like Animated Pedagogical Agent with Deictic Believability. *Working Notes of the IJCAI '97 Workshop on Animated Interface Agents: Making them Intelligent*, pp. 61-69, Japan, August 1997

[Lest97d] Lester James C., Converse Sharolyn A., Stone Brain A., Kahler Susan E. and Barlow Todd S. Animated Pedagogical Agents and Problem-Solving Effectiveness: A Large-Scale Empirical Evaluation. *Proceedings of the Eighth World Conference of AIED*, pp. 23-30, Japan, August 1997

[Lest97e] The IntelliMedia Initiative Projects: Design-A-Plant, http://www.csc.ncsu.edu/eos/users/l/lester/www/imedia/DAP.html, November 1997

[Lest97f] The IntelliMedia Initiative Projects: Internet Advisor http://www.csc.ncsu.edu/eos/users/l/lester/www/imedia/IPA.html, November 1997

[Lest99] Lester James C., Strone Brian A. and Stelling Gary D. Lifelike Pedagogical Agents for Mixed-Initiative Problem Solving in Constructivist Learning Environments. *User Modeling and User-Adapted Interaction*, 9(1-2), pp. 1-44, 1999

[Math98] Mathé Nathalie, Chen James R. and Wolfe Shawn R. Organizing and Sharing Information on the World-Wide Web using a Multiagent System, proc. *ED-MEDIA '98*

[Maur98] Maurer Hermann. On Two Aspects of Improving Web-Based Training. *Proceedings of ED-MEDIA '98*, pp. 973-977, 1998

[Mitr98a] Mitrovic Antonija. Learning SQL with a Computerised Tutor. *Proceedings of the 29th SIGCSE Tech. Symp*, pp. 307-311, 1998

[Mitr98b] Mitrovic Antonija. Experiences in Implementing Constraint-Based Modeling in SQL-Tutor. *Proceedings of ITS '98*, pp 313-423, August 1998

[Mitr99] Mitrovic Antonija and Stellan Ohlsson. Evaluation of a Constraint-Based Tutor for a Database Language, *International journal on AIED*, 10, 3-4, 1999

[Pies98] Piesk J. and Trogemann G. Presenting Educational Contents in Nonlinear Narrative Structures by Conventional Virtual Actors. *Proceedings of ED-MEDIA '98*, pp. 1885-1888, 1998

[Schö98] Schöch Volker, Specht Marcus and Weber Gerhard. "ADI"- An Empirical Evaluation of a Tutoring Agent, *Proceedings of ED-MEDIA '98*, pp. 1271-1282, 1998

[Steve] Steve in Action, http://www.isi.edu/isd/carte/carte-demos.htm

[Town98a] Towns Stuart, Callaway Charles, Voerman Jennifer and Lester James C. Coherent Gestures, Locomotion, and Speech in Life-Like Pedagogical Agents. *Proceedings of the Fourth International Conference on Intelligent User Interfaces*, pp. 13-20, San Francisco, January 1998

[Town98b] Towns Stuart G., FitzGerald Patrick J. and Lester James C. Visual Emotive Communication in Lifelike Pedagogical Agents. *Proceedings of the Fourth International Conference on Intelligent Tutoring Systems*, San Antonio, Texas, pp. 474-483, August 1998

Appendix A

Pre- and post-tests

Appendix B

Questionnaire

Appendix C

Results of Questionnaire

The agent group

	Enjoyment	Time to learn Interface	Ease of using the interface	Amount learnt	Usefulness of feedback
1	3	10	3	4	4
2	4	10	4	2	4
3	5	5	4	4	5
4	4	5	4	4	5
5	5	5	4	4	5
6	5	5	4	3	5
7	5	5	3	4	5
8	5	5	5	4	5
9	4	30	5	4	5
10	5	30	5	5	5
Mean	**4.5**	**11**	**4.1**	**3.8**	**4.8**
Standard Deviation	**0.71**	**10.22**	**0.74**	**0.79**	**0.42**

The control group

	Enjoyment	Time to learn Interface	ease of using the interface	Amount learnt	Usefulness of feedback
1	4	10	5	4	5
2	3	30	3	4	2
3	5	5	4	4	4
4	2	5	2	3	3
5	5	10	4	4	4
6	3	10		3	
7	5	5	4	4	4
8	3	5	3	3	5
9	4	5	5	4	5
10	5	5	5	5	5
11	3	30	3	5	3
12	4	10	3	4	5
Mean	**3.83**	**10.83**	**3.73**	**3.92**	**4.09**
Standard Deviation	**1.03**	**9.25**	**1.01**	**0.67**	**1.04**

Appendix D

Results of problem-solving logs

Agent group

	Total interaction time (min)	No of attempted problems	No of solved problems	Total no of attempts taken to solve the problems	Problems solved on the first attempt	Problems solved per time (prb/min)	Attempts per solved problem	Attempts to solve problems that could not be solved on the first attempt (attempts/prb)
1	32	8	6	16	2	0.19	2.67	2.50
2	55	18	11	18	6	0.20	1.64	1.40
3	73	15	14	34	7	0.19	2.43	2.86
4	61	19	18	39	9	0.30	2.17	2.33
5	27	9	9	13	6	0.33	1.44	1.33
6	58	8	8	39	3	0.14	4.88	6.20
7	80	13	9	44	2	0.11	4.89	5.00
8	67	21	15	27	7	0.22	1.80	1.50
9	42	9	8	22	2	0.19	2.75	2.33
10	64	20	18	57	7	0.28	3.17	3.55
Mean	**55.90**	**14.00**	**11.60**	**30.90**	**5.10**	**0.22**	**2.78**	**2.90**
Standard deviation	**17.30**	**5.27**	**4.35**	**14.13**	**2.60**	**0.07**	**1.23**	**1.61**

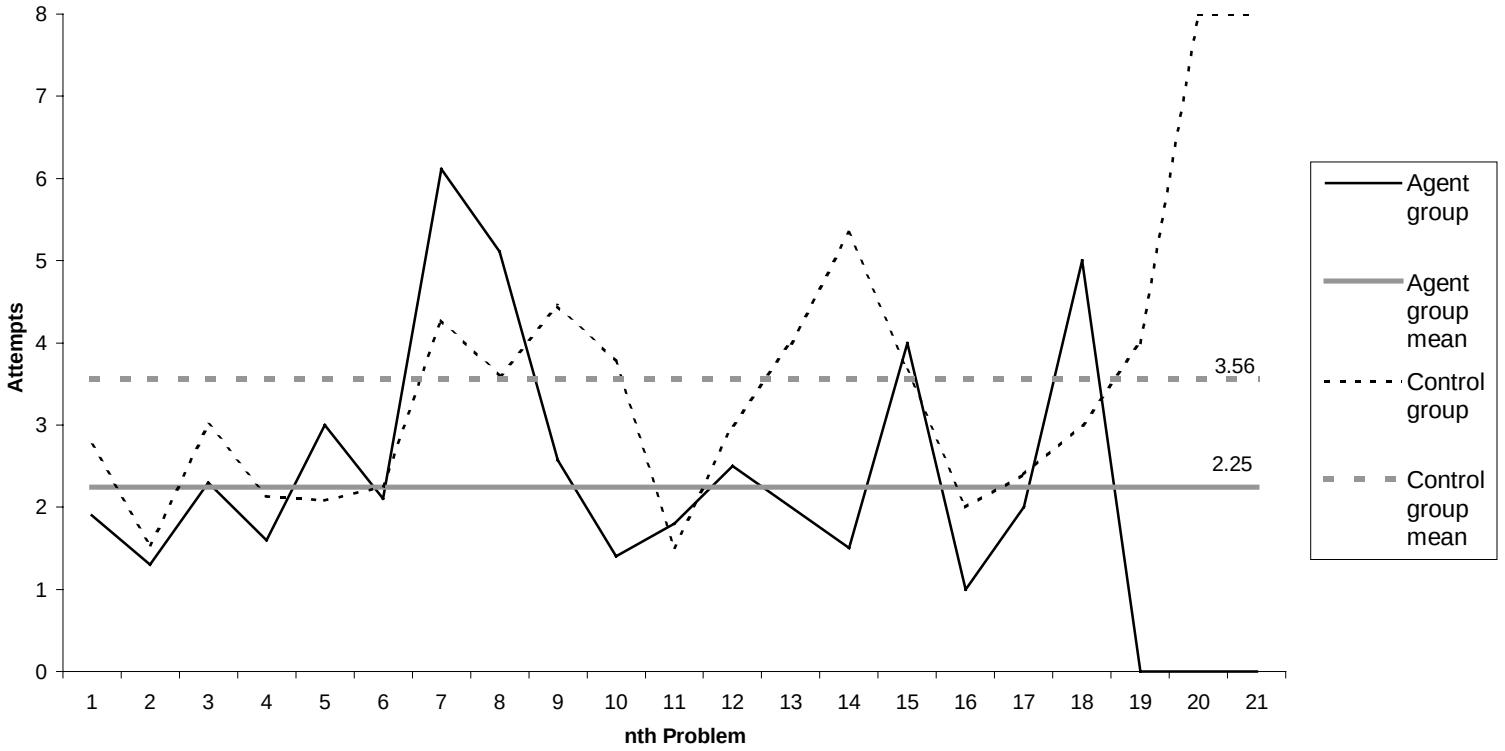
Control group

	Total interaction time (min)	No of attempted problems	No of solved problems	Total no of attempts taken to solve the problems	Problems solved on the first attempt	Problems solved per time (prb/min)	Attempts per solved problem	Attempts to solve problems that could not be solved on the first attempt (attempts/prb)
1	32	11	10	21	6	0.31	2.10	2.75
2	52	20	18	51	8	0.35	2.83	3.30
3	43	5	4	26	0	0.09	6.50	5.50
4	73	19	19	47	9	0.26	2.47	2.80
5	90	22	21	60	8	0.23	2.86	3.00
6	23	5	4	9	1	0.17	2.25	1.67
7	72	7	6	14	3	0.08	2.33	2.67
8	61	10	10	41	3	0.16	4.10	4.43
9	67	10	10	26	5	0.15	2.60	3.20
10	4	1	1	2	0	0.25	2.00	1.00
11	72	18	18	72	6	0.25	4.00	4.50
12	20	6	5	6	4	0.25	1.20	1.00
13	8	8	8	10	6	1.00	1.25	1.00
14	59	16	15	36	6	0.25	2.40	2.33
15	36	8	8	20	4	0.22	2.50	3.00
16	82	19	18	80	4	0.22	4.44	4.43
Mean	**49.63**	**11.56**	**10.94**	**32.56**	**4.56**	**0.27**	**2.87**	**2.91**
Standard deviation	**26.70**	**6.49**	**6.36**	**23.97**	**2.73**	**0.21**	**1.33**	**1.34**

Confidence levels

	Total interaction time (min)	No of attempted problems	No of solved problems	Total no of attempts taken to solve the problems	Problems solved in first attempt	Problems solved per time (prb/min)	Attempts per solved problem	Attempts to solve problems that could not be solved in first attempt (attempts/prb)
With agent								
Average	55.90	14.00	11.60	30.90	5.10	0.22	2.78	2.90
Standard deviation	17.30	5.27	4.35	14.13	2.60	0.07	1.23	1.61
No agent								
Average	49.63	11.56	10.94	32.56	4.56	0.27	2.87	2.91
Standard deviation	26.70	6.49	6.36	23.97	2.73	0.21	1.33	1.34
Z	0.73	1.05	0.32	-0.22	0.50	-0.90	-0.16	-0.02
Confidence level that hypothesis is correct	**53%**	**71%**	**25%**	**18%**	**38%**	**63%**	**13%**	**1%**

Mean number of attempts taken to solve n^{th} problem



Appendix E

Results of Pre- and Post-tests

Pre-test

Agent group

	Scores				Confidence level			
	Q1	Q2	Q3	Total	Q1	Q2	Q3	Total
abp18.log	1	4	1	6	2	1	2	5
Alan.log	0	4	1	5	1	1	0	2
dah70.log	0	3	0	3	0	0	0	0
ejd32.log	0	4	1	5	2	1	2	5
Michae_~1.log	0	2	1	3	2	1	2	5
mpj18.log	0	1	1	2	1	1	1	3
ndm17.log	1	1	0	2	1	1	0	2
sgb39.log	1	2	0	3	1	0	1	2
sph37.log	0	2	1	3	1	0	2	3
Mean	0.33	2.56	0.67	3.56	1.22	0.67	1.11	3

Control group

	Scores				Confidence level			
	Q1	Q2	Q3	Total	Q1	Q2	Q3	Total
bgh27.log	0	1	0	1	1	1		2
bgr30.log	0	4	1	5	1	1	1	3
bjh74.log	0	3	1	4	0	0	1	1
bmf25.log	0	3	1	4	2	1	1	4
bruce.log	0	3	0	3	2	0	0	2
cgc30.log	0	1	0	1	2	1	2	5
hlh16.log	1	3	1	5	1	1	1	3
jls68.log	0	4	1	5	2	2	2	6
Luke.log	0	4	1	5	2	0	2	4
Mike.log	0	2	1	3	2	0	2	4
mjp86.log	1	1	1	3	1	1	1	3
pjc82.log	0	2	0	2	1	1	0	2
pta19.log	0	2	1	3	1	1	0	2
t.log	0	2	1	3		1	1	2
Mean	0.14	2.50	0.71	3.36	1.38	0.79	1.08	3.25

Post-test

Agent group

	Scores				Confidence level			
	Q1	Q2	Q3	Total	Q1	Q2	Q3	Total
Alan.log	1	4	0	5	2	1	2	5
Dah70.log	1	4	0	5	2	1	1	4
Sgb39.log	0	3	0	3	2	1	1	4
Mean	0.67	3.67	0	4.33	2	1	1.33	4.33

Control group

	Scores				Confidence level			
	Q1	Q2	Q3	Total	Q1	Q2	Q3	Total
t.log	0	2	0	2	2	1	1	4