INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

The Virtual Chef

An Animated Pedagogical Agent

Sara Denise Soares Passos

Dissertação para Obtenção de Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente: Prof. Dr. Nuno João Neves Mamede

Orientador: Profª. Dr.ª. Ana Maria Severino Almeida e Paiva

Vogais: Profª. Dr.ª Maria Luísa Torres Ribeiro Marques da Silva Coheur

Prof. Marco Paulo de Sousa Correia Vala

Setembro 2008

Resumo

Os agentes pedagógicos estão a tornar-se cada vez mais uma parte do nosso dia-a-dia. Eles podem ajudar e assistir os alunos, tendo um papel cada vez mais importante nos ambientes de aprendizagem. Estes agentes, não só conseguem fornecer algumas dicas, mas também, em alguns casos, podem executar demonstrações práticas, podendo até substituir um professor humano. Alem destas características, os agentes pedagógicos estão em constante crescimento apresentando-se como uma grande promessa nos ambientes de aprendizagem. No entanto, ainda há espaço para melhoramentos nesta área. Existem muitos domínios que ainda não foram explorados e quando forem poderão causar um grande impacto.

Esta dissertação foca-se essencialmente nas estratégias de ensino e como é que elas podem ser aplicadas a um agente pedagógico, em particular, um cozinheiro. Foi desenvolvido um modelo onde foram descritas todas as características necessárias para a construção de um cozinheiro virtual, quais as estratégias que devem ser utilizadas e porquê. O principal objectivo deste trabalho era provar que as estratégias utilizadas por este cozinheiro virtual eram adequadas e assim é possível melhorar o processo de aprendizagem, tornando-o mais eficaz e agradável.

Depois de implementar um agente que abrangesse todos os requisitos impostos no modelo conceptual, foram feitos alguns testes preliminares onde o utilizador observa uma demonstração feita pelo cozinheiro e, depois, tenta executa-la interagindo com o sistema. Os resultados alcançados sugerem que os utilizadores gostaram de interagir com o agente e que as ajudas fornecidas pelo agente foram eficazes e oportunas. Com estes resultados é possível inferir que a presença deste cozinheiro virtual influencia positivamente o processo de aprendizagem. No entanto, de modo a obter resultados mais refinados, mais alguns testes precisam ser feitos.

Palavras-Chave: agentes pedagógicos, estratégias de ensino, ambiente de aprendizagem, processo de aprendizagem

Abstract

Pedagogical agents are becoming more and more a part of our daily lives. They can assist and collaborate with students, playing a very important role in learning environments. These agents not only can provide proper advices but also perform practical demonstrations and even replace the human teacher. Moreover, animated pedagogical agents are in constant growth and offer a great promise to interactive learning environments. However, there is still room for improvements in this area. There are several domains which have not been covered yet, and whenever that happens may cause a great impact.

The focus of this dissertation is tutoring strategies and how can they be applied into an animated pedagogical agent, in this case, a virtual chef. It was developed a model which describes the main characteristics this virtual chef should have, which strategies should be taken in consideration and why. The main goal of this work was to prove that the tutoring strategies used by the virtual chef were adequate and could improve the learning process, by making it more effective and enjoyable.

After implementing an agent that could fulfil the demands of the conceptual model, some preliminary experiments were performed where the users could watch a cook's demonstration and, afterwards, execute the recipe previously learnt. The results achieved suggested that the users thought the interaction with the agent was pleasant and the assistance provided was appropriate. With these results is possible to infer that the virtual chef influences positively the user's learning process. However, some more experiments still need to be done.

Keywords: Pedagogical agents, tutoring strategies, learning environment, learning process

Acknowledgements

Finally, it is getting to an end this journey of my life.

A new one will begin and I will carry with me lots of memories, good memories…

First of all, I would like to thank my family, especially my mother for all the effort she made all over these years, so I could have a good education; for all her never ending support throughout my whole life. Thank you! To my grandmother, a very special person, I would like to thank her, just for being there! For her patience, her experienced advices… thank you for everything! It is great to have you as my grandmother!

Over the last 6 years, I have met some great people in this university, friends that I know I will keep for years and years. All the nights up doing projects, the study groups for tests, the dinners at Bairro Alto… the great companionship found among our colleagues is something I will never forget.

However, there were two of them that really made a difference in my journey: Ricardo and Diogo. These were my partners on most of my projects and quickly became my great friends. I thank you both for all that you taught me, for all the great time we spent together, even when we were working!!

I would also like all members of GAIPS for always being available at any time during these last two years, to Prof. Ana Paiva, and João Dias for helping me with the Language Engine. And I couldn't forget Leila, my office partner, who always listened to all my thoughts! Thank you!

I would like to leave a very special thanks to Marco Vala, for helping me when the people I could turn to was sparse. Most of this work couldn't be done without you. Thank you very much!

Last, but not least, I would like to leave some words to my boyfriend, João. He was always my support, especially in these final months. When everything looked bad, there he was turning it better! Above everything, thank you for being my friend, my best friend, thank you for always being there!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Motivation

Pedagogical agents are becoming more and more a part of our daily lives. They can be essential in assisting and collaborating with students [12] and they play a very important role in learning environments. In such applications, these agents not only can provide proper advices but also, in some cases, perform practical demonstrations.

In particular, embodied pedagogical agents can also assume the role of “partners”: they can perform multi-person tasks, play the role of a missing member or even replace the human teacher. One of the key elements for the success of embodied pedagogical agents is making them lifelike and believable. Thus, the learning process becomes more effective, motivating and enjoyable [4, 6].

Moreover, this new generation of learning technologies can have a great impact on education and training. Embodied pedagogical agents are in constant growth and offer a great promise to interactive learning environments. It is enormous the evolution that this area has suffered over the last years, where those textual dialogue systems on a terminal have been successfully replaced with all these complex lifelike animated agents.

However, there is still a lot of work to be done. Despite these first efforts that resulted in some amazing pedagogical agents, there is still room for improvements in this area. There are several domains which have not been covered yet, and whenever that happens may cause a great impact.

“Cooking” is one of these unexplored domains. At some point in our lives, we definitely need to learn how to cook, even if it’s just an omelette! Of course there are lots of conceived recipes’ books (“bibles” such as *Pantagruel* or *Boston Cooking Book*) which teach us everything we need to know to become an excellent cook. But, nowadays, there are several sites on the Internet with an enormous variety of recipes and tips from all over the world. There are also appearing several new cooking programs on TV. There are some chefs that became famous just because of their programs, such as Jamie Oliver. These cooking programs are having a lot of success maybe because it is much more interesting and enjoyable to see a recipe being done, instead of reading it on a book. Besides, it is much easier to memorize something if we actually see it being done.

So, since cooking is such an important asset and it is becoming more and more a part of the new technologies, why don't introduce animated pedagogical agents in this domain and try to turn this learning process more interesting?

1.2. Objective

Having this context, this work tries to find out the main strategies used on the most conceived examples of animated pedagogical agents, and which of them can be considered to be the most effective on each domain. Taking these strategies into account, this work aims to create an animated pedagogical agent, a virtual chef, capable of providing an effective learning process. Thus, this thesis addresses the following research question:

What are the main requirements for building an animated pedagogical agent, in particular, a virtual chef?

Giving this, and since the research focuses mainly on pedagogical strategies, the hypothesis this work aims to prove is:

By analyzing the existent animated pedagogical agents and its tutoring strategies, if the virtual chef employs the strategies that best adequate to the cooking domain, then the user may experience a better learning process, more motivating and pleasant.

In this work, the agent is embedded in a scenario where he tries to explain a certain recipe through gesticulation and textual speech and implements some tutoring strategies that may be applied in the cooking domain. Then, it will be performed a preliminary evaluation where it will be examined the tutoring capability of the agent by analysing the users' reaction to their interaction with the different modes of the system. Thus, it will be possible to perceive if the agent influences the user's learning process and how.

1.3. Outline

This document is divided into six different sections.

Chapter 2 is the Related Work which makes a research on animated pedagogical agents, more precisely, on their tutoring strategies. First of all, it's made a brief description of the most important strategies. Then, it's presented a study of the state of the art in this area, analyzing the strategies used by each of them. At the end, it is made a comparison between the agents resulting in the conclusion.

Chapter 3 describes the conceptual model. In this chapter, the main idea behind the conception of the virtual chef is presented. Here is detailed the main aspects that should be taken into account when conceiving a virtual chef, including the tutoring strategies that should be followed and why.

Chapter 4 is the implementation. Here it is depicted how the conceptual model previously defined was implemented. It is related the technology used, some implementation decisions and the details of the integration with the other platforms used.

Chapter 5 details the Evaluation. Here is described all the methodology followed in order to perform the evaluation: how the tests were planned, how they were executed and a description of the participants evaluated. In the end, the preliminary results achieved are presented and discussed.

Finally, Chapter 6 presents the Conclusion. Here is made a brief description of the work covering the path since the initial problem, the hypothesis to prove and the respective conclusions. There are also made some final remarks concerning future work that can be made as an extension to this one.

Chapter 2

Related Work

This section presents the research made in the areas of pedagogical agents and its tutoring strategies.

First, important aspects of pedagogical agents are described, focusing on both tutoring strategies and elements that aid learning for achieving an effective learning process.

Then, using these strategies as guidelines, several examples of animated pedagogical agents were evaluated. These agents are Steve, Herman the Bug, Cosmo, Adele, AutoTutor and a Reactive Virtual Trainer, and they represent state-of-the-art work in this area.

Finally, a comparison between these agents is made, followed by a brief conclusion.

2.1. Characteristics of Pedagogical Agents

Pedagogical agents for knowledge-based learning can provide customized support and advice for student's problem solving. They have two essential advantages: increasing the communication between students and computers and increasing the computer's ability to motivate students [6]. As mentioned before, the important features of a pedagogical agent can be divided in two groups: tutoring strategies and elements that aid learning.

2.1.1 Tutoring Strategies

The main purpose of a pedagogical agent is to teach or train a student on a specific domain. Just like a human, a teacher needs to be able to transmit and provide knowledge in a simple and easy way. That is how we distinguish a good from a bad teacher.

Pedagogical agents appeared with the goal of facilitating and improving the learning process and experience. In some contexts, they can substitute human teachers and, therefore, must be as effective as possible, trying always to be distinguished as the "good teachers".

In order to succeed as a tutor, the agents must implement some important features, which are described below.

Interactive Demonstrations. When an agent inhabits the virtual world, it is possible to perform "live" demonstrations and, therefore, explain how to execute a certain physical task. The fact of seeing a task being done, leads to a better retention. The demonstrations can also include spoken explanations, so the agent can describe its actions while actually performing it. Interactive demonstrations provide several advantages [13]: the agent can adapt the demonstration to different situations; the students can move around in the virtual environment

and view the demonstration by different perspectives; and the students can also interrupt with questions.

Navigational Guidance. When the environment is complex, students frequently get disoriented and, therefore, navigational guidance is very useful. Using this feature, the agent can guide the students, leading them around, showing the relevant objects and places and how to get there, preventing them of getting lost. Agents that can also serve as guides provide an important support for students and their learning effectiveness.

Virtual Teammates. When a certain task is very complex, it requires different members to perform coordinated tasks. For example, teamwork is critical in an emergency room or in a battlefield. Some complex virtual environments require several teammates, where each one performs a certain task which needs to be coordinated with all other team elements' tasks. In order to execute the task without conflicts, it's important that every element dominates its individual task or role and learn how to coordinate it with all the other members. Distributed virtual reality is a promising vehicle for training teams. In this kind of training, the students may interact as team members in different locations, although cohabiting a common virtual environment, where they practice together realistic simulations. In this case, agents may perform two different roles: (1) serve as teachers for individual students or (2) substitute missing team members.

Adaptive Pedagogical Interaction. Pedagogical agents must be as dynamic and adaptive as possible instead of being pre-planned and sequential. It is useful if they have the ability to generate explanations, ask questions, track the learner's skill levels or generate dynamic and appropriate advices. The agents must also be able to track and respond to students' actions while they're performing the interactions described above. Another characteristic very important in adaptive pedagogical interaction is opportunistic instruction. In this case, the pedagogical agent takes advantage of a certain situation and provides appropriate instruction (hints, advices or explanations).

2.1.2 Elements that aid learning

The instruction strategies defined above aren't enough for a proper and effective learning. There are some other elements that enhance learning. These elements have the main purpose of capturing and focalizing student's attention, providing appropriate feedback and encouraging and motivating the students.

Attentional Guides. In a learning environment, it is very important to maintain the student always focused on the correct object or task. Agents may use many strategies to guide the user's attention. In animated agents the most common strategy is using gaze and gestures as attentional guides. By using these features, the agent can capture more easily the students'

attention where deictic gesture has a very important role. Pointing at objects while explaining what they're for, looking directly at an object just before manipulating it or looking at students waiting for them to speak or to execute a task, are some examples of attentional guides.

Feedback. One of the most important characteristic in a tutor is the ability to provide feedback to students' actions. With feedback, students may easily understand when they are executing correctly a task or making some mistakes. The agent can provide verbal feedback that may occur, e.g., immediately after a student's turn. There is positive ("Right", "Yeah"), neutral ("Okay", "Uh-huh") and negative ("No", "Not quite") feedback. Additional to verbal feedback, an agent can also offer nonverbal feedback. For example, while a user is doing a certain task, the agent can nod or smile as a sign of approval, or shake the head when an error has been made. Nonverbal feedback can prevent the user from committing a mistake and, sometimes, it is more appropriate than a verbal feedback, since is less perturbing.

Conversational Signals. These signals are very useful in face-to-face conversations, helping it to flow more easily and smoothly. The agent may use pitch accent to indicate the degree and type of salience of words and phrases in an utterance. It is possible to highlight these utterance elements by accompanying pitch accents with short movements of eyebrows or head or blink of the eyes [1]. The agent can also combine nonverbal signals (such as eye contact) with spoken utterances to regulate turn talking in mixed-initiative dialogue. For example, during a pause, a speaker will either break eye contact to retain to the floor, or make eye contact to request feedback [1].

Conveying and Eliciting Emotion. Motivation is one of the key elements in learning. An emotive pedagogical agent can motivate students, conveying enthusiasm for the subject matter [11]. Animated agents can improve student's learning in various ways [6]: (1) if an agent appears to care about the student's progress, may encourage the student to care about his own progress; (2) when an agent is able to convey enthusiasm for the subject matter, then he may potentiate the same levels of enthusiasm in the learner and, finally (3) if an agent has a rich and interesting personality, the learning may become more fun. By combining body placement, facial expressions and hand movements, the agent can give advices, encourage students and even increase their motivation [6]. When the user enjoys interacting with the agent, the learning process becomes more effective.

All the features described above are considered the most important ones in order to provide a learning process as effective and good as possible.

It's important to mention that none of the current pedagogical agents implement all features described. Each characteristic, per se, is a real asset. Combining them differently can provide great and interesting results.

Some of the most conceived pedagogical agents will, now, be described.

2.2. Examples of Pedagogical Agents

2.2.1 Steve

Steve (Soar Training Expert for Virtual Environments) is a pedagogical agent that inhabits a 3D virtual environment architecture called VET (Virtual Environments for Training project), functioning both as tutor and collaborator for students [12,13], constantly monitoring and controlling the environment. Steve not only can teach students how to perform certain tasks such as repairing and maintaining complex devices, but also can function as a team member even replacing the human instructor if necessary. One of the most important features of Steve is that his capabilities are domain-independent, using a declarative representation of domain tasks. Thus, this representation language permits that people with domain knowledge can easily create pedagogical agents for virtual environment training (due to its user-friendly interface).

Next, it will be presented the main features of this pedagogical agent.

2.2.1.1. Tutoring Strategies

Interactive Demonstrations. Steve has the ability of performing interactive demonstrations and explaining how to do some physical tasks such as operation and repair of equipment (Figure 1). This agent has the advantage of providing verbal explanations while demonstrating some task, describing important actions, as well as pointing at relevant objects related to the current task. As Steve is constantly monitoring changes in the virtual world, he can revise plans for completing a task and, therefore, demonstrate them under different initial states as well as helping students to recover from errors.

In order to perform demonstrations Steve must always be aware of the current action and choose the next move accordingly. To implement all actions and tasks Steve uses hierarchical plans. Each plan consists of a set of steps, which may be primitive or complex actions. There can be some ordering constraints among the steps, which mean some steps must be executed following a certain order. Finally, the role of each step in a plan is represented by a set of casual links (specifies that one step in a plan achieves a goal that is a precondition for another step in the plan).

Steve's architecture is composed by three main modules: perception, cognition and motor control. The perception module is the responsible of providing a coherent view of the state of the virtual world to the cognition module, through a sensorimotor component. This component serves as Steve's interface to the virtual world, allowing the cognitive component to perceive the state of the world and cause changes on it. On the other hand, the cognitive module assesses the input received, carries our plans to achieve goals, chooses the next Steve's actions and sends high-level commands to control the agent's voice and body (motor control module).

Figure 1 – Steve pressing a button

Navigational Guidance. Steve inhabits a complex shipboard environment with lots of rooms, hence navigational guidance has a very important role. When the agent is demonstrating some task, he leads the students around the environment showing them all the relevant places, objects and how to get them. Due to Steve's internal representation of the ship, he can always find the shortest path from his current position to the next relevant object or place.

Virtual Teammates. Steve can perform team training, being each team composed as any combination of Steve agents and human students. It's worthwhile to mention that Steve agents have important nonverbal communication when it concerns team actions: they can look at a teammate when waiting for them or speaking to them, they can react to their teammates' actions, and even nod in acknowledgement when they say something a teammate says to them [13].

Adaptive Pedagogical Interaction. Steve can adapt his demonstrations in "real-time", when students perform some actions that interact with the demonstration. He can also reply to students' interruptions.

Steve monitors constantly students' actions. This is possible due to the communication process in Steve's architecture. Students, agents (as Steve) and the simulator that controls the virtual world run in different processes, communicating by messages. For example, when a student presses a button on a virtual control panel, the button sends a message indicating it has been pressed. The simulator, after receiving, will update the state of the world and send messages to the other process indicating any state changes.

2.2.1.2. Elements that aid learning

Attentional Guides. Steve uses both gaze and gestures in various situations in order to guide students' attention. For example, he points to an object when referring to it (Figure 2), looks at objects while students are manipulating them, looks at students when waiting for them, listening or speaking to them and even can track objects' movements. However, Steve has another attentional guide: body orientation. When he moves towards an object, he ends his locomotion with his body facing it. During an explanation, although the gaze consequently shifts from the student to the object, the lower body continues facing the object so the student can keep his focus on the desired object. Only when the focus shifts to a new object the orientation changes.

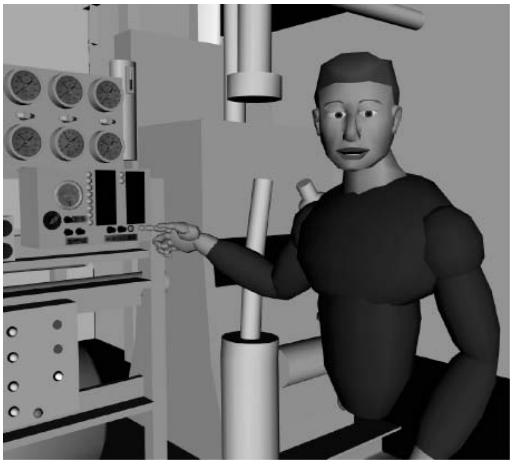


Figure 2 – Steve pointing at an object

Feedback. Steve uses nonverbal feedback as complement his verbal explanations. He nods showing his approval when students make correct actions and shakes his head when they make a mistake showing disapproval.

Conversational Signals. Conversational signals are very useful in systems that support speech recognition as well as speech output, such as Steve. He provides back-channel feedback such as head nods showing his comprehension of a spoken utterance, and also uses eye contact to regulate mixed-initiative dialogue.

2.2.2 Herman the bug

Herman the bug [7, 10] inhabits a learning environment called Design-A-Plant, which was built with the main purpose of investigating interactive problem-solving with animated pedagogical agents. In this learning environment, students can learn the fundamentals of botanical anatomy and physiology. This agent advises students as they try to build and design plants to survive in different environmental conditions (Figure 3). If a plant does not survive,

students must modify their components to improve its suitability, and this process repeats until they have built a robust plant that prospers in that environment.

Herman is a talkative insect that flies around the environment and dives into plant structures (leafs, roots, stems) while providing some explanations and hints as students perform their tasks. He possesses a big variety of actions such as walking, flying, shrinking, expanding, fishing, bungee jumping, swimming, teleporting and acrobatics! Herman's main features will be described below.

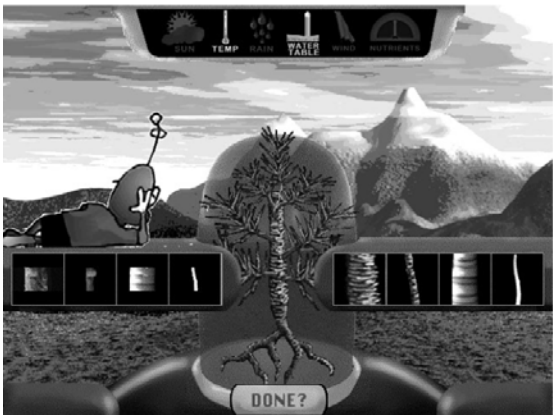


Figure 3 – Herman the bug

2.2.2.1. Tutoring Strategies

Adaptive Pedagogical Interactions. Delivering instruction based on the current situation is an important characteristic in pedagogical agents and, therefore, is becoming more and more usual. Herman the bug uses problem-solving context as opportunities for instruction. For example, when a student is trying to find a leaf to include in a plant, Herman takes the opportunity to explain about leaf topology. These context-related explanations and hints occur for the various plant structures.

Herman constructs his actions by using a behaviour space. The behaviour space contains: *manipulative* behaviours (agent manipulates objects), *visual attending* behaviours (e.g. agents watches a student), *re-orientation* behaviours (e.g. the agents stands up), *locomotive* behaviours (e.g. agent moves from one point in the environment to another), *gestural* behaviours and *verbal* behaviours.

In this framework, behaviours are dynamically sequenced by a *competition-based* sequencing engine. Each behaviour has associated an *exhibition strength*, a measure that determines the behaviour's current degree of appropriateness for exhibition. Thus, these behaviours compete with one another for the right to be exhibited.

2.2.2.2. Elements that aid learning

Feedback. When a student is performing a task, Herman remains seated on the considered plant structure looking at it and waiting for the student's move. If this move takes too long, he starts tapping his foot. When a student finishes a task Herman congratulates him verbally, but, sometimes, he even cartwheels along the screen, showing his happiness and satisfaction.

2.2.3 Cosmo

Cosmo [8,9] is an animated agent that teaches Internet packing routing. He has a head with movable antennas, expressive eyes and its torso it's similar to an accordion. Students interact with Cosmo, navigating through a series of subnets. They need to make decisions concerning address resolution, traffic congestion and, therefore, learn the basis of network topology and routing mechanisms. He provides explanations on how computers are connected, how routing is performed and how traffic considerations fit in.

Cosmo was implemented to study the *deictic* believability in pedagogical agents, that is, the ability to combine gestures, locomotion, and speech to refer to objects in virtual environments unambiguously.

Next, Cosmo's main characteristics will be explained.

2.2.3.1. Tutoring Strategies

Adaptive Pedagogical Interaction. This feature is manifested in Cosmo by the way he provides assistance to students. This assistance occurs in three distinct situations: when a student asks for help; when a student pauses for a long period of time; when a student commits an error. In these cases he explains the related concepts and also provides hints whenever thinks it is necessary.

2.2.3.2. Elements that aid learning

Attentional Guides. Cosmo employs a deictic behaviour planner, so he can coordinate gestures, locomotion and speech. He implements a variety of behaviours such as pointing, blinking, leaning, clapping, and raising and bending his antenna (Figure 4). During a problem-solving session, Cosmo stays onscreen and makes use of these attentional guides to show he is still "alive" and aware of all students' actions. Usually, he shows his awareness by blinking his eyes, leaning on one of the routers and moving up and down through the virtual environment.

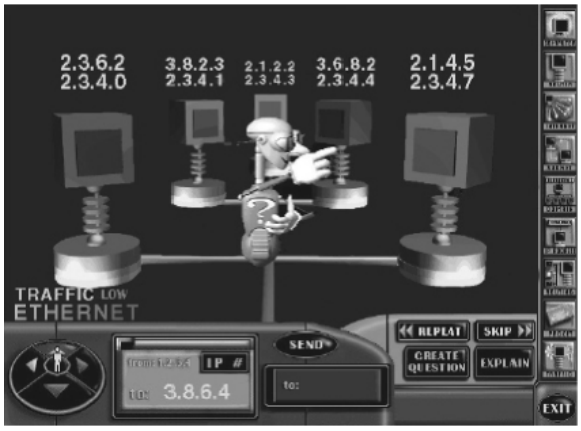


Figure 4 – Cosmo pointing at an object

In order to provide deictic believability, Cosmo has an agent behaviour planner that consists on an explanation and a deictic planner.

The explanation planner determines the content and structure of explanations, constructing a sequence of explanatory behaviours and utterances which will compose the advice that will be delivered.

Whenever the behaviour planner determines that an explanation must refer an object, the deictic planning is called. For each utterance that makes reference to an object in the environment, the explanation planner invokes the deictic system supplying it with the intended referent. This system is responsible for planning all the agent's gestures, speech and locomotion, which is composed by three steps: (1) ambiguity appraisal, (2) gesture and locomotion planning and (3) utterance planning.

In ambiguity appraisal, the deictic system determines if a reference R is potentially ambiguous. The ambiguity is analyzed based in focus (an object may be ambiguous if is not in focus or if is in focus but if one of multiple objects in focus). This ambiguity may be eliminated by a combination of locomotion and gestures. Therefore, this assessment will be essential in gesture, locomotion and speech planning decisions.

The deictic system has a world model, which represents the relative positions of the agent and the objects. This representation is, of course, very useful in planning all gestures and locomotion needed. Based on the ambiguity assessment, the system determines if the agent should point to R, move to it and how to move.

Finally, to determine what the agent should say to refer to an object (R), the deictic system considers focus information, ambiguity assessment and world model. The result of utterance planning is a referring expression consisting of the appropriate proximal/non-proximal demonstratives and pronouns.

The deictic system passes the agent's behaviours and utterances back to the agent planner, which integrates them in the final behaviours' specifications.

Feedback. When it concerns feedback, Cosmo uses, mostly, the nonverbal one. He implements this feature differently from other agents described, combining the usual facial

expressions with some gestures and body movements. For example, when a student solves a problem, he not only smiles, but also uses his entire body to applause his success. In contrary, when a student makes a sub-optimal decision, he takes on a sad facial expression and drops his hands.

Conveying and Eliciting Emotion. Cosmo is a great example of an emotive pedagogical agent. He employs a vast repertoire of emotive "full-body" behaviours to advise, encourage and empathize with the student. For example, when a student finishes a task he claps his hands enthusiastically or when a student makes a mistake he uses a slumping body language. Cosmo's "funny" visual aspect and his body animations' repertoire is, definitely, an asset to encourage and empathize with students.

2.2.4 Adele

Like Steve, Adele (Agent for Distance Education: Light Edition) was developed by CARTE following an agent-oriented approach [5,15]. She was designed to run on desktop platforms with conventional interfaces, operating in a Web-based environment, which is used in health science courses. In a typical use of Adele, students can examine the simulated patient, ask questions, order and interpret diagnostic tests as well as make diagnoses and create treatment plans. She constantly monitors all the students' actions, providing appropriate hints and explanations and asking relevant probing questions.

The most important characteristics of Adele are described below.

2.2.4.1. Tutoring Strategies

Adaptive Pedagogical Interactions. When it concerns pedagogy, Adele supports some additional capabilities comparing with Steve. She provides some adaptive interactions such as opportunistic learning, contextual reference and student assessment.

Figure 5 – Adele providing an explanation

Adele is able to recognize pedagogical opportunities when a student performs a certain task. For example, when a student orders some tests, Adele can ask him about its results, adapting her questions to verify the student's understanding.

This agent provides materials that are relevant to the instructional goals. These materials can be documents, videos, images or even animations. They are available as a part of a Web-based library, and the results are shown based on the current context. For instance, if a student is examining a patient, Adele knows that the context is physical examination. So, if he referenced the library at that time, the "home page" would contain topics related to this subject.

Adele is also monitoring the students constantly. She records the student's interaction with the simulation and, during the task, verifies if all the steps are done in the correct order, giving the appropriate feedback. When a student finishes the task, Adele analyzes the student's record, making his evaluation. For example, in a clinical situation, this assessment is made concerning three aspects: evaluation of the diagnosis, evaluation of the diagnoses costs and evaluation of the steps taken.

Adele has also a *Hint* and a *Why* button. When a student is unsure about some action, he can ask for a hint, preventing him from taking an incorrect action. He can also ask for an explanation by pressing the *Why* button (Figure 5).

This method doesn't restrain the student from making a mistake but prevents from it, using a less disturbing method.

Adele's architecture consists of two main components: the pedagogical agent and the simulation. The pedagogical agent consists in two other sub-components: the reasoning engine and the animated persona. A third component may exist, the session manager, but only when the system is run in multiple-user mode.

The reasoning engine is responsible for all monitoring and decision making. The decisions are made based on the student model and current mental state (updated as the student works through the case), a case task and an initial state (downloaded from the server when a case is chosen). When the student finishes the task, all actions made by him are recorded to the server, where the assessment of the student's level is made and will determine how Adele will deal with him in future cases.

When it concerns knowledge, the representation scheme was designed to be as simple and general as possible, which are essential aspects in order for the reasoning engine run efficiently in the client side.

Adele's knowledge representation focuses on the steps that the student should take and the dependencies between them, represented in a task plan. All the task plans are described in such a general way that allows students to perform the actions in their preferred order, as long as they follow the critical ordering constraints.

When it concerns plan representation, Adele follows a similar approach of Steve's. All procedural tasks are represented using a hierarchical plan, composed by steps, each one being

either a primitive or a complex action. The plan hierarchy is evaluated at each step to track efficiently student's actions, due to its unpredictability. Actions, whose goals are still undone, are automatically re-executed, while the ones that were satisfied are skipped. Thus, Adele's task plan is always dynamically updated.

The task plan described above, although effective in, e.g., diagnostic tasks, does not extend well to dynamic settings such as trauma or critical care. Therefore, Adele has a *Situation Space*, which structures a set of states (situations) associated with a domain. A situation is defined by a name, world state, goal expression, priority and set of transitions. The priority is essential to order situations when more than one is appropriate. As a result, Adele's reasoning engine is a *situation-monitoring* task. There's always a current situation defined in Adele's reasoning engine, which monitors the world changes that may cause a transition to another situation. Thus, the current plan that forms the basis of tutoring may change accordingly to the situation.

2.2.4.2. Elements that aid learning

Attentional Guides. Although Adele runs in a separate window, she uses gaze and gestures to give the impression she's integrated and aware of what's happening in the other windows running on the desktop. Adele turns her head to see where a student has clicked (Figure 6) and even points to objects with a pointer similar to the one used by PPP Persona [16]. In a way, this can compensate her inability to manipulate objects.

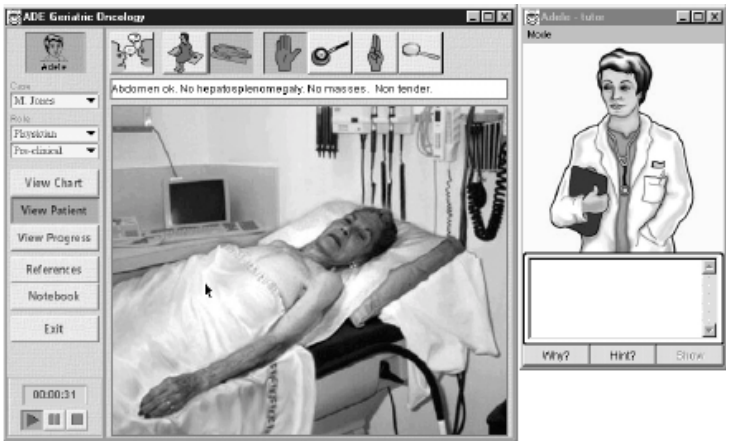


Figure 6 – Adele looking at the mouse selection

Feedback. Adele has a repertoire of several facial expressions and body postures representing emotions. When a student performs correctly a task, she nods or smiles showing her approval and shows pleasant surprise when the students finishes a task. In opposite, when a student makes a mistake, she presents a look of puzzlement and disappointment.

2.2.5 AutoTutor

AutoTutor [2,3] is a computer tutor that simulates the discourse patterns and pedagogical strategies of a typical human tutor. This agent provides learning in computer literacy, focusing the fundamentals of hardware, operating systems and Internet. It is a very complete agent, where the integration of expertise from many different fields was required. AutoTutor has a talking head, serving as a conversation partner with the learner (Figure 7). This agent assists students in construction of improved answers, stimulating them to actively generate explanations, justifications and functional procedures instead of just reciting small pieces of knowledge.

AutoTutor is composed by seven modules: (1) curriculum script, (2) language extraction, (3) speech act classification, (4) latent semantic analysis, (5) topic selection, (6) dialog move generation and (7) talking head.

Recently, has been developed, a new version of AutoTutor with an Interactive 3-D simulation.

AutoTutor's main features are described below.

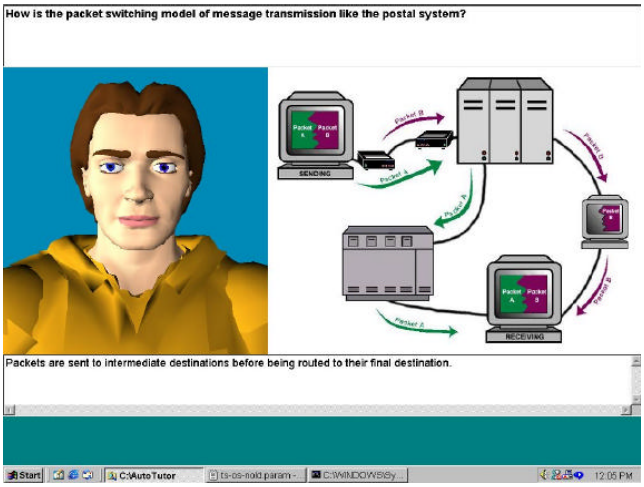


Figure 7 – AutoTutor with Marco Talking Head

2.2.5.1. Tutoring Strategies

Adaptive Pedagogical Interaction. During a conversation with a student (lesson), AutoTutor adaptively selects hints, prompts and assertions. This agent uses some important strategies when it concerns adaptive pedagogical interaction: Pumping, Prompting, Hinting, Elaborating, Correcting, Summarizing and *Requestioning*.

Relatively to pumping, the tutor pumps the student for more information during early stages of answering (e.g. "What else?", "Tell me more"), encouraging students to create answers by themselves.

When it concerns Prompting, the tutor supplies the student with some fill-in-the-blank questions. This strategy is used when the student is reluctant to supply information.

When the agent perceives a student is having difficulties in answering some question, he provides some hints or remakes the problem.

In the case of *elaborating*, the tutor provides some information that he considers important and might be missed by the student.

Whenever a student makes a wrong intervention, the tutor detects it and, immediately, corrects him.

When a student finishes a task, the agent summarizes all the important aspects needed to be referred in a problem resolution.

Finally, *requestioning* is a strategy used when the student is deviating from the supposed answer. When this happens, the tutor re-asks the original main question.

In order to implement correctly all the strategies of adaptive pedagogical interactions described above, AutoTutor uses a curriculum script which is responsible for organizing the topics and content of the tutorial dialogue. This script includes didactic descriptions, tutor-posed questions, examples problems, figures and diagrams. Each topic in the curriculum is represented either as structured set of propositions or as a free text format, and is scaled on three levels of difficulty (easy, medium, difficult). Associated with each topic in the script, is a set of good and bad responses, which grows with tutoring experience, just like in human tutors.

The topics are selected by unclear production rules that are sensitive to the structure of the curriculum script and to the learner's capacity. Within each macro-topic (hardware, operating systems and Internet), the difficulty level of the topic is matched with the student's ability, i.e., the good students receive difficult questions, while poorer student get the easy ones. This student's ability is based on the learner's Assertions in the current tutorial session.

The language modules analyze the words in the messages that the students type in the keyboard during a conversation. There is a large lexicon with nearly 10.000 words. Each lexical entry specifies its alternative syntactic classes, called "tags". Many words may be categorized into more than one synthetic class being, thus, resolved by the context.

2.2.5.2. Elements that aid learning

Attentional Guides. AutoTutor mostly uses gestures as attentional guides. Some topics require graphical displays and animations and, in these cases, the agent points out some important components. By being a talking head, that synchronizes speech, facial expressions and gestures, AutoTutor concretely grounds the conversation between the student and the tutor.

Feedback. As a conversational agent, the verbal feedback is the mostly used. AutoTutor gives feedback immediately after a student turn. This feedback can be positive ("Yeah", "Right"), neutral ("OK", "Uh-huh") or negative ("No", "Not quite"). The ability to provide feedback is very

important because not only motivates students but also creates the impression the tutor is listening to what the student is communicating. This verbal feedback is complemented by some facial expressions, e.g., if a student answers incorrectly the agent expresses puzzlement.

AutoTutor performs speech act classification, by a segmentation of the string in words and punctuation marks within a learner's turn into speech act units. The current system relies on punctuation to perform this segmentation. A neural network is used to classify each speech act unit into one of the follow five speech ac categories: Assertion, WH question, Yes/No question, Directive and Short Response. Besides punctuation, the first three words of the speech act are also taken in consideration in speech classification.

After the learner types in the content of his turn, the agent needs to generate dialog moves. Sometimes AutoTutor needs to answer some questions, but mostly he needs to respond to and build on the assertions of the learner. Most of the AutoTutor's turns consist in two parts: short immediate feedback (positive, neutral or negative), followed by a more substantive contribution that prompts the student for more information, adds information or corrects a student error. The categories of the substantive dialog moves during AutoTutor's turn are determined by a set of production rules: (1) the quality of student's assertions in the preceding turn, (2) parameters that refer the ability, verbosity and initiative of the student and (3) if the good answer aspects of the topic have been covered.

Conveying and Eliciting Emotion. AutoTutor always tries to motivate students by delivering pumps, trying to encourage active learning and question answering from the student. Whenever the agent feels that is necessary to bring back the student, he provides some clues, trying to encourage the student and preventing him from giving up.

2.2.6 Reactive Virtual Trainer

The Reactive Virtual Trainer (RVT) is an Intelligent Virtual Agent (IVA) that is able to present physical exercises which are performed by a human, monitoring the user and providing feedback at different levels [14]. This virtual trainer may be applied to several contexts, e.g., performing general fitness exercises to improve a physical's condition or physiotherapy exercises with medical indications.

RVT has a great advantage: she is universal, i.e., she may be applied (with minor investment) to various different users and objectives. She shares the most essential characteristics of real trainers and physiotherapists: to be reactive, both on professional and psychological basis. RVT possesses three major capabilities: (1) perceiving the user's performance in non-lab environments using non-intrusive devices, (2) evaluating the reaction of the user and deciding how to react and (3) generating reaction accordingly and present it in a natural and subtle way.

RVT's main features are described below.

2.2.6.1. Tutoring Strategies

Interactive Demonstrations. This agent would not make any sense if did not implement this feature. RVT's main goal is to explain a certain exercise and, therefore, demonstrations are essential so the user can confirm if he is executing the exercise correctly (Figure 8).

In order to compose all the exercises it is used *basic motions* which are the smallest units of motion which can be combined to build more complex ones. Gestures may be defined as *motion expressions*, built up also from basic motions.

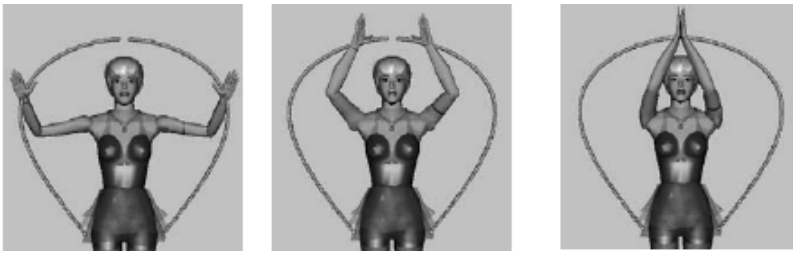


Figure 8 – A clapping exercise

Adaptive Pedagogical Interaction. RVT must adapt to the user and context is being used. She may be programmed for a general exercise program or for a medical critical application and the agent must provide exercises accordingly. RVT must monitor constantly the user's performance in a non-obtrusive way. This can be achieved by the input given by an every-day camera connected to the PC hosting RVT. By tracking the movements performed, the facial expressions and its coloration and reflection (sweat), the physical state of the user may be perceived. Acoustic signals are also important once it is possible to detect if the user is jogging at the right tempo or if he's out of breath. Through these perceptions it is possible to readjust exercises. For instance, if the RVT observes the user is very tired, she may slow down the tempo (helping the user to maintain synchronized), shorten the number of repetitions still to come or finish the current exercise.

Acoustic cues is a feature already implemented. Audio is processed for tempo and beat information which is used to adapt the RVT's movements properly. The beat and tempo tracking is performed by an improved implementation of Klaputi's beat tracking algorithm, capable of real time performance.

2.2.6.2. Elements that aid learning

Feedback. RVT provides both verbal and nonverbal feedback. The verbal one has an extreme importance once the agent should address the user time to time, during or in between performing an exercise. RVT must show the user the tempo for performing the exercise (e.g., counting or clapping).

Once feedback is a crucial component for success, it should follow the next three aspects [14]: (1) should be used a set of different natural languages in order to achieve variety, accompanied or replaced by facial expressions and gestures, (2) some utterances must be synchronized to the tempo of the exercises and (3) addressing the user should be made clear by gaze behaviour and head movement.

Conveying and Eliciting Emotion. This feature is very important due to the nature of this domain. Not every people like to do exercise, especially when it concerns physiotherapy. In this case, the users do exercises just because they are forced to do it (medical issues). Here, motivation plays an important role and the RVT must take that in consideration by adapting the exercises preventing the users from giving it up (e.g., due to its difficulty).

2.3. Comparison

In this section, a comparison between all the agents described above will be done. Table 1 summarizes all the agents' instruction strategies, while Table 2 resumes all the elements that help the learning process.

Features/ Agents	Interactive Demonstrations	Navigational Guidance	Virtual Teammates	Adaptive Pedagogical Interaction
Steve	√	√	√	√
Herman the Bug	×	×	×	√
Cosmo	×	×	×	√
Adele	×	×	×	√
AutoTutor	×	×	×	√
Reactive Virtual Trainer	√	×	×	√

Table 1 – Implemented instructions strategies

Features/ Agents	Verbal Feedback	Nonverbal Feedback	Attentional Guides	Eliciting Emotion	Conversational Signals
Steve	√	√	√	×	√
Herman the Bug	√	√	×	×	×
Cosmo	√	√	√	√	×
Adele	×	√	√	×	×
AutoTutor	√	√	√	√	×
Reactive Virtual Trainer	√	√	×	√	×

Table 2 – Implemented elements that aid learning

Steve is, probably, the most complete agent. He implements all instruction strategies described, and eliciting emotion is the only element not present. Besides recognizing voice, Steve provides verbal explanations, as well as navigational guidance and virtual demonstrations. These demonstrations are a real asset, once they permit the student to participate and/or finish the task, interrupting at any time of the explanation. This provides an enormous flexibility and variety in the learning process, which is very advantageous for students.

From all the agents described above, Herman the Bug is the one with fewer features. However, these characteristics are very well explored. He possesses verbal and nonverbal feedback and, as well as Adele, adaptive pedagogical interactions as his only instruction strategy. Herman provides context based problem-solving, always trying to introduce useful hints to enhance students' learning. He is, also, very flexible and offers the possibility to implement multiple levels of advice (task-specific or principle-based).

Cosmo is also a very complete agent. He provides both verbal and nonverbal feedback and has a very important feature in pedagogical agents: Conveying and eliciting emotion. This is a very important aspect once motivation is a key element in the learning process. Cosmo provides a variety of gestures, facial and verbal expressions that elicit emotion and encourage students. This is, definitely, one of the agent's main features. However, the most important feature in Cosmo is *deictic* believability: he is able to move through the environment, point to objects and

refer to them appropriately and unambiguously while providing problem-solving advice. Like the other agents described, Cosmo also provides attentional guides and the instruction strategy chosen is adaptive pedagogical interactions, however this last aspect is better explored in the agent Adele.

When it concerns Adele, although she has been created by the same Institute as Steve (CARTE), she presents more limitations. One reason is that they were developed to perform in very distinct contexts. Due to Steve's environment complexity, the agent needs to implement some aspects such as navigational guidance and interactive demonstrations. In Adele's case, these characteristics are not so relevant.

One of the major assets in Adele it's her adaptive pedagogical interactions. Her capacity of providing context-related feedback is much superior to Steve's. This feedback includes not only student's monitoring/assessment but also a library that always presents the context related topics and hints. Although verbal feedback isn't present in Adele, the repertoire of attentional guides and nonverbal feedback is very complete, which perfectly compensates this lack.

AutoTutor is also a very complete agent. Although he only implements one instruction strategy, this is very well explored, where the agent presents a variety of strategies to perform an effective adaptive pedagogical interaction. It simulates perfectly a human tutor, which is a result of deep researches on human teaching strategies.

When it concerns elements that help and enhance the learning process, only conversational signals is not implemented in AutoTutor. All other elements are very well implemented, especially the agent's capability of providing appropriate feedback.

RVT is the most actual agent described in this paper, however is still in development. Although not all characteristics are implemented, it is a very promising work. The instruction strategies appear very effective and this agent will, probably, be a reference in the pedagogical agents' domain.

All the examples above are conceived pedagogical agents. As this area is relatively recent, presently there's still a lot of research, where improvements are naturally expected.

2.4. Concluding Remarks

In this chapter was made a research of the tutoring strategies in the area of pedagogical agents. Several conceived pedagogical agents were evaluated according to its tutoring strategies and elements that aid leaning.

Looking at these examples, it is possible to reason that all the characteristics explained play an important role depending on the domain they are used. A good pedagogical agent must, at least, have adaptive pedagogical interactions and feedback. Both verbal and nonverbal feedback are crucial to a student perform a task correctly. Adaptive pedagogical interactions are

also very important since the tutor must be able to adapt his tutoring strategies to all different students and levels.

Although in an early stage of development, this new generation of learning technologies seems to be obtaining significant impact on education and tutoring. Nowadays, pedagogical agents are beginning to realize their full potential using multimodal skills and performing with a great lifelike behaviour when compared to the old-fashioned dialogues on a terminal.

However, there is still room for improvements in the area. The use of pedagogical agents in learning environments was not explored in several domains and may create an instant impact. Perhaps, these agents are the missing link to achieve an effective communication with students in some of those domains.

Chapter 3

Conceptual Model

This chapter explains the main idea behind the conception of a virtual chef. It describes the necessary elements to build a virtual cook, the strategies that should be taken into account as well as some aspects concerning the recipe and the character's features.

3.1. The System

On Chapter 2, several strategies used in the area of pedagogical agents and some representative examples where described.

Using these examples, and since cooking is an unexplored domain in pedagogical agents, an interactive system was idealized, a virtual chef, which tries to contemplate the characteristics of real cooks, taking always into account the strategies seen on Chapter 2. Figure 9, below, illustrates the system's basic behaviour.

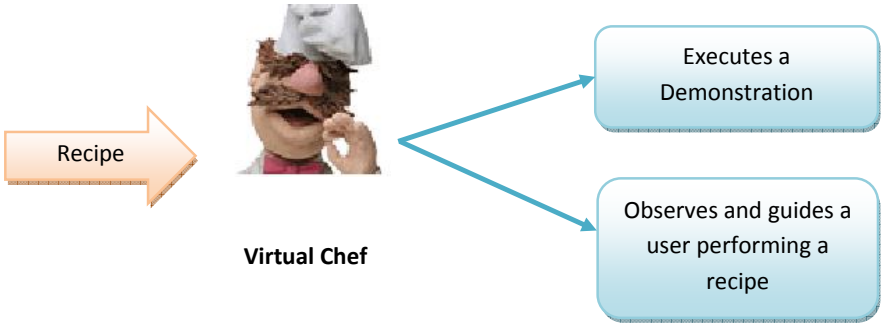


Figure 9 – Virtual Chef's basic behaviour

This virtual chef is an animated pedagogical agent, focused on the cooking domain, which receives a recipe, and, based on it, can perform two distinct tasks: execute a demonstration or observe and guide a user performing a recipe.

In the first one, the agent performs a demonstration of the recipe using gestures and textual explanations, being always aware of the user's feedback. The user may ask the agent to repeat some step whenever he needs and the agent must interrupt his demonstration and step backwards in his explanation. The agent must also be able to explain the recipe in several different ways or levels of detail to adapt his demonstration to the user's knowledge,

In the second interaction mode the agent observes the user performing a recipe. Although the agent doesn't assume an active posture in this interaction mode, he must always be aware of the user's input. According to the user's performance the agent may need to help and guide him in order to complete his task. This help may be reflected through some hints or tips or even another demonstration of that particular step, which is performed using, again, gestures and textual explanations.

So, in both interaction modes, the agent is constantly receiving feedback from the users that directly influences his reactions, choices and demonstrations.

The following sections will describe the main requirements in order to build a virtual chef. First, the structure of the recipe will be described as well as the main concepts behind it. Afterwards, both interaction modes will be depicted, where the strategies adopted in each of them are explained. Finally, the Virtual Chef's strategies are summarized and compared with the other examples of pedagogical agents studied on Chapter 2.

3.2. The Recipe

The recipe assumes a very important role in the agent's behaviour, since all the actions and gestures performed by the virtual chef are based on it. The recipe's basic structure is illustrated on Figure 10 below.

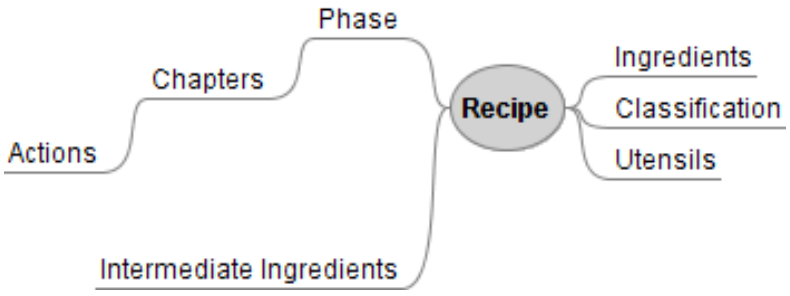


Figure 10 - Recipe's basic structure

The several concepts of this recipe are explained below and followed by some examples for a better comprehension of its structure.

Ingredients represent each ingredient of the recipe. These are the basic ingredients and, comparing with a normal recipe, they can be for example, sugar, wheat flour, eggs, among others. It is obvious why this concept needs to be included on the Virtual Chef recipe's structure. Ingredients are the starting point of every recipe. After all, we cannot make an omelette without breaking the eggs!

On the other hand, **intermediate ingredients** are the ones that result from the combination of other ingredients. For example when we mix sugar and yolks it will resolve into a cream. It is

important for a virtual chef to distinguish the plain ingredients from the resulted ones. Only that way, is possible to link the several actions by a logic (and correct) sequence.

Concerning **classification**, this element keeps some characteristics of the recipe such as duration or difficulty. These characteristics appear in the beginning of every recipe on the cooking books and represent the amount of work needed and how long it will take. This concept doesn't play a very important role in the virtual chef's conception however it has the purpose of contextualizing the user at the beginning of the recipe.

Each recipe may have one or two **phases**: preparation and/or confection. The preparation phase occurs before even starting the recipe; for example, let's imagine we want to prepare a dish with codfish. Almost every recipe using codfish starts by soaking it over night. This is considered the preparation phase. On the other hand, the confection phase, and again using the codfish dish as an example, will start when the chef starts adding ingredients to the codfish such has garlic, spices, olive oil, among others. Just as the classification, this concept has an information purpose only, although it is important for the user to know what phase of the recipe he is learning.

Each phase contains a set of **chapters** which group a logic sequence of actions. Using the previous example of the sugar and yolks, the cook needs to separate the eggs before using the yolks. So the action of separating the eggs is directly related to the one of mixing the sugar and yolks and will appear right before this last one. This concept can reveal to be very useful for the virtual chef whenever he needs to repeat an action that is directly related to another one.

Concerning the **action**, this is the basic concept of the recipe. It contains information relative to each step of the recipe such as the description, the kind of the action (stir, melt …) or the ingredients involved. The description contains what you will be performing with this action. For example, when we want to beat egg whites stiff, the kind of action will be beat while the ingredients involved, in this case, will be egg whites. This concept is very important for the virtual chef. First, it contains all the information about each step to be done which can be used in the explanation text. Then, each type of action can represent a direct link to each agent's gesture.

Finally, the **utensils**. These are needed in every recipe and are a precious help to perform certain actions, for instance, when a cook needs to put together sugar and yolks, he will use a mixer. This information can be very useful to the virtual chef's explanations, since he should be able to describe the utensils needed in each step.

3.3. Demonstration Mode

On Chapter 2, some pedagogical agents and their tutoring strategies were scrutinized. These strategies were divided into tutoring strategies and elements that aid learning, and they differ from agent to agent depending on their goals and domain.

Using this as a guideline and focusing on the coking domain, each interaction mode of the Virtual Chef employs some tutoring strategies and elements that aid learning that may improve the user's learning process in this particular domain.

In this section, will be presented the strategies used in the first interaction mode, the demonstration one, what were the reasons to employ them in this particular agent and how they can be applied.

3.3.1 Tutoring Strategies

The tutoring strategies chosen for this mode of Virtual Chef were: Interactive Demonstrations and Adaptive Pedagogical Interaction.

When an agent inhabits a virtual world, it is possible to perform "live" **demonstrations** and, therefore, explain how to execute a certain task. These demonstrations can include spoken explanations so the agent can describe the tasks while performing them. Considering the purpose of this interaction mode it is quite direct the choice of using this strategy. Virtual Chef needs to perform practical demonstrations. A kitchen may be considered a complex environment with all its appliances, utensils and ingredients and, therefore, it is easier to follow and people tend to pay more attention to demonstrations because they can actually see all steps being done.

So, the Virtual Chef needs to explain a certain recipe just like is seen on those TV cooking programs. Therefore, he should be able to perform interactive demonstrations. For that, the agent should include in his explanation the use of gestures. It is very important for users to see the actions being done.

Adaptive pedagogical interaction is one of the key strategies in pedagogical agents. They should be as dynamic and adaptive as possible instead of being pre-planned and sequential. These agents should be able to generate explanations, ask questions and provide proper advices according to the learner's level. Making a recipe it's a very complex task. Not only due to the diversity of ingredients and possible actions but also due to the large number of steps a recipe may have. Having this, it is important that the Virtual Chef can collaborate and assist the users whenever needed, adapting his explanations and guiding his demonstrations according to the user's level and performance and providing some adequate tips or hints whenever necessary.

In order to execute the recipe, the cook must have a tree of actions that represents the path he needs to follow. However, this path shouldn't be static. The agent may need to change his

route and should permit the user to interrupt his explanation whenever needed, having the possibility of repeating each step. Another important feature is that the cook should adapt his explanation according to the student's level. Depending on the user's knowledge the agent may need to provide more detailed explanations.

3.3.2 Elements that aid learning

In this particular interaction mode, the element that seemed more adequate was the attentional guides.

In a learning environment, it is very important to maintain the student always focused on the correct object or task. Agents may use many strategies to guide the user's attention. In animated agents the most common strategy is using gaze and gestures as attentional guides. By using these features, the agent can capture more easily the students' attention where deictic gesture has a very important role. Therefore, it is important that the Virtual Chef includes attentional guides so the user can stay focused on the recipe's execution. The agent should be able to point at an ingredient or utensil unambiguously. Thus, the user may easily follow the agent's explanation simultaneously with the interactive demonstration.

3.4. Try-it! Mode

This is the second possible interaction with the Virtual Chef: observing a user executing a recipe.

In this section, will be presented the strategies used in Try-it! Mode, what were the reasons to employ them in this particular agent and how they are applied.

3.4.1 Tutoring strategies

The tutoring strategy chosen for this second mode of Virtual Chef was Adaptive Pedagogical Interaction. Although this strategy has already appeared in Demonstration Mode, in this one it is used and applied in different conditions.

As explained before, **adaptive pedagogical interaction** is one of the key strategies in pedagogical agents. The agents should be able to generate explanations, ask questions and generate proper advices according to the learner's level.

In this interaction mode, the cook presents a more passive behaviour. But "passive" doesn't mean indifferent. Although the agent doesn't have an active role, he must always be aware of all students' movements. This may be the difference between a good and a bad tutor. The fact of observing the user's actions may prevent several mistakes. The cook must be able to "feel" whenever the user is in trouble. When this happens, he should be capable of providing

appropriate tips or hints, i.e. the tips must be directed to the problem the user is facing. If necessary, the cook must intervene and explain again the step or action to the student.

3.4.2 Elements that aid learning

In this interaction mode, the elements explored were: Feedback and Conveying and Eliciting Emotion.

A very important characteristic in a tutor is the ability to provide **feedback** to students' actions. With feedback, students may easily understand when they are executing correctly a task or making any mistake. The agent can provide verbal and nonverbal feedback, being this last one, in some cases, more effective since it is less disturbing. This strategy should be included in all systems where the user can actually interact with it, independently the domain chosen. Therefore, it was important that Virtual Chef implemented this strategy. When the user is executing a recipe he may commit mistakes and, by providing appropriate feedback, the cook not only explains the correctness of his actions but also may prevent the user from committing some more mistakes in the future.

Finally, motivation is one of the key elements in learning. An emotive pedagogical agent can motivate students and, therefore, **conveying enthusiasm** for the subject matter [10]. By combining body placement, facial expressions and hand movements, the agent can give advices, encourage students and even increase their motivation [6]. Concerning the Virtual Chef, it is important he includes this strategy. Due to the large number of steps a recipe may have, the user may become bored or demotivated, so a constant and pleasant encouragement can make a difference in the user's learning process.

Virtual Chef includes feedback in several situations. When the user is performing an action he should always have the information if he is performing it in a correct way or not. This feature can prevent the user from committing another similar mistake in the future. It is important that the hints or tips are integrated in the feedback given, so the agent can not only inform the mistake but also help the user to recover from it. Moreover, the feedback should always include some encouraging and motivating comments, independently if it is a good or a bad one. Thus, when the feedback is good and the agent still motivates the user, he gets even more encouraged and confident. But, when the feedback is concerning a mistake, an encouragement always feels nice and incentives the user not to give up.

3.5. Concluding Remarks

This chapter tried to explain the main idea behind the conception of a virtual chef.

Using tutoring strategies and elements that aid learning studied in Chapter 2 as guidelines, this section tries to infer the most adequate strategies to employ in a virtual cook.

Using the structure of Chapter 2, this section will summarize the tutoring strategies used by the Virtual Chef and compares it with the other pedagogical agents studied. Table 3 will summarize the instruction strategies used while Table 4 resumes the elements that aid learning.

Features/ Agents	Interactive Demonstrations	Navigational Guidance	Virtual Teammates	Adaptive Pedagogical Interaction
Virtual Chef	√	×	×	√
Steve	√	√	√	√
Herman the Bug	×	×	×	√
Cosmo	×	×	×	√
Adele	×	×	×	√
AutoTutor	×	×	×	√
Reactive Virtual Trainer	√	×	×	√

Table 3 –Instruction strategies

Features/ Agents	Verbal Feedback	Nonverbal Feedback	Attentional Guides	Eliciting Emotion	Conversational Signals
Virtual Chef	×	√	√	√	×
Steve	√	√	√	×	√
Herman the Bug	√	√	×	×	×
Cosmo	√	√	√	√	×
Adele	×	√	√	×	×
AutoTutor	√	√	√	√	×
Reactive Virtual Trainer	√	√	×	√	×

Table 4 – Elements that aid learning

As seen on Chapter 2, there are two strategies that should always be present in a pedagogical agent: Feedback and Adaptive Pedagogical Interaction. The Virtual Chef includes them both which are definitely his strongest aspects. The agent's demonstrations are quite adaptive to the user knowledge level as well as the tips provided that always focus the real problem the user is facing.

In addition, Virtual Chef is always worried on providing adequate feedback so the users never feel lost. Motivation is a constant feature included in the feedback, where the agent tries not to lose his students, so constantly encourages them.

However, the other two strategies are not so strong. Virtual Chef's interactive demonstrations only include textual explanations and the execution of representative gestures, when it could be interesting if the agent could actually move around the environment just like Steve does. Concerning attentional guides, although Virtual Chef points at the objects and ingredients, he doesn't include facial expressions and gaze. This can be a very important issue when concerning this strategy and Cosmo represents a good example of that.

Chapter 4

Implementation

The following lines will describe the most important implementation issues behind the virtual chef's system. It will explain the choices made in order to implement all the strategies previously described on Chapter 3 and a brief overview of the technology needed.

Figure 11, below, illustrates the basic architecture of this system.

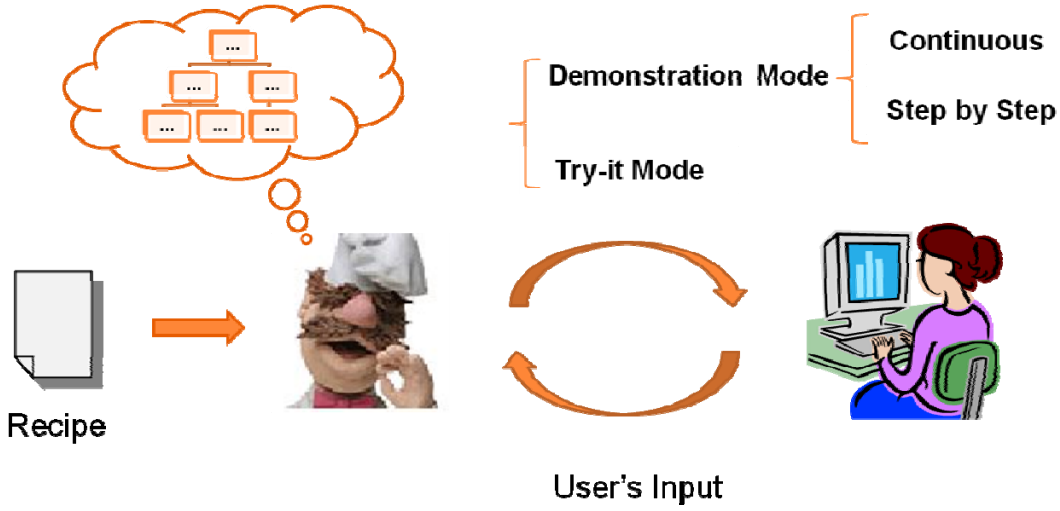


Figure 11 – System's basic architecture

The following sections, will detail the most important aspects concerning this architecture.

4.1. The recipe

The recipe is the starting point of this system. It contains all the information the agent will need and therefore, supports all his utterances, gestures and actions.

On Chapter 3, Section 3.2, the elements a recipe should contain were described: Ingredients, Intermediate Ingredients, Classification, Phase, Chapters, Actions and Utensils.

Figure 12, below, illustrates an overview of the recipe's internal structure, with all these elements and the relations between them.

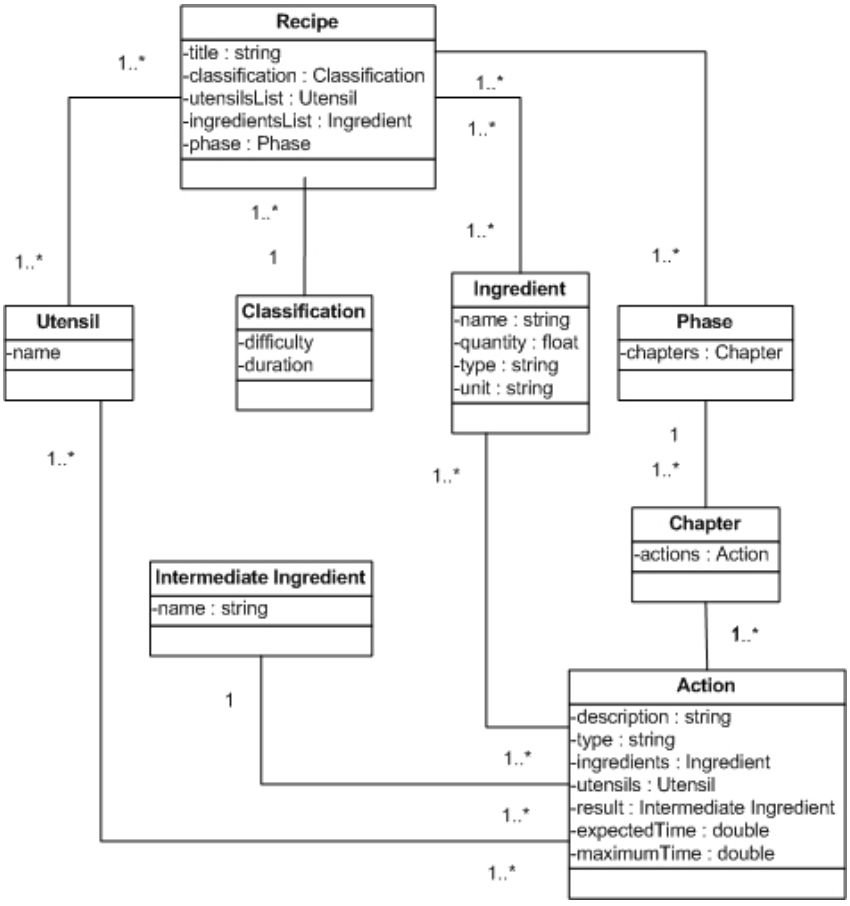


Figure 12 – Recipe's structure

The conception of the recipe's structure was based on an existing ontology made especially for the cooking domain: *OntoChef* [17]. Although some concepts weren't included in the ontology, the existent ones had a direct match to the previous list. For each concept it is shown an example of its correspondence in the recipe XML file.

The **Ingredient** contains the information of each ingredient such as its name, quantity and unit. This ontology has a huge food library, where it is possible to find almost every type of ingredient. However, and because this was not the main focus of this work, there was the need to create a type where similar ingredients are aggregated in some more comprehensive groups. They were created based on the "Food Wheel", although not so restrictive so it could be easily understood by any kind of user. These groups are:

- Sweets – chocolate, sugar, condensed milk…
- Fats – olive oil, oil, butter, margarine…
- Hydrates – cereals, pasta, rice, beans…
- Proteins – meat, fish, eggs...

- Fruits – apple, banana…
- Vegetables – carrot, lettuce, tomato, cabbage…

Next, is possible to see how this concept is represented in the XML File.

```
<ingredient>
    <name>chocolate</name>
    <quantity>250</quantity>
    <unit>gram</unit>
    <type>sweets</type>
</ingredient>
```

The **Classification** contains information of the recipe such as duration and difficulty. The following example shows an excerpt of a recipe where the classification tag contains values for the duration and difficulty, 30 min and easy, respectively.

```
<classification>
    <duration>30 min</duration>
    <difficulty>easy</difficulty>
</classification>
```

The recipe presents two distinct **phases**: preparation and conception. Each of them contains chapters and the mainly difference between them in the structure of the XML file of the recipe is the tag given to each,

```
<confection>
(…) <!--Here it would go a list of chapters-->
</confection>
```

or

```
<preparation>
(…) <!--Here it would go a list of chapters-->
</preparation>
```

The **chapter** groups the several logic sections of a recipe. Each chapter is composed by a set of related and ordered actions. This was conceived with the purpose of supporting a tutoring strategy. When a user doesn't understand a certain step of a recipe it can be useful to repeat not only the last action but also the related ones because they represent a logic sequence. The respective representation in XML file can be seen below.

```
<chapter>
(…)<!--Here it would go the list of actions -->
</chapter>
```

The **action** contains information relative to each step of the recipe such as description, type of the action, ingredients involved, utensils and result. In order to support a tutoring strategy there was the need to include the information about the time expected and maximum time of an action. Having these time metrics, it is possible to infer if the user is experiencing some kind of trouble or difficulty (e.g. if the user exceeds the time expected of a certain action) and provide the appropriate assistance. All the information is represented in the XML file of the recipe under the node with the tag action. The actions' information previously described on Chapter 3, Section 3.2 has their own tags on the XML file so it could have a direct association with each of them as it is shown in the following example.

```
<action>
  <description>Put the chocolate and butter in a pan</description>
  <type>join</type>
  <what>chocolate</what>
  <what>butter</what>
  <where>pan</where>
  <how></how>
  <result>chocolate and butter</result>
  <expectedTime>
    <time>00:00:30</time>
  </expectedTime>
  <maxTime>
    <time>00:01:20</time>
  </maxTime>
</action>
```

The utensils contain a list of the utensils needed all over the recipe. They are used in the XML file under the tag utensil and it contains the name of the several utensils that are required. The following example illustrates how the XML text for the utensils looks like.

```
<utensil>
  <name>spoon</name>
</utensil>
```

Besides all these concepts, it was needed to create some other one not present in the ontology in order to support the Virtual Chef's needs.

There was the need to establish a difference between base and **intermediate ingredients**. These last ones are the result of a combination between normal ingredients and/or intermediate ones. This separation is essential to find out the dependencies between the actions and, therefore, to create a sequence of actions for the agent to follow (represented by a tree of actions). In the XML file these ingredients appear in the result field of the action as the example given shows.

```
<action>
(…)
  <result>chocolate and butter</result>
</action>
```

When the application starts, the user chooses a file which contains the recipe's information. This file comes in XML format and was conceived based on the structure previously described. An example of this file can be seen on Appendix A.

4.2. Recipe Reasoning Engine

The entire recipe's information needs to be organized for a future use by the agent. He is responsible for choosing the right path to guide his demonstration as well as adapting it depending on the user's "performance".

All the information processing is done by a Recipe Reasoning Engine module.

This module has the goal of building a tree of actions based on the recipe chosen by the user. This module is separated in two different stages: Parsing Data and Build Actions Tree.

The first stage, reads the XML input file, parses all information and organizes it for future use on the next stage.

The second stage is responsible for building the tree which the agent will follow and use in real-time, using the information saved on the previous step.

To build this tree it was necessary to create an effective data structure where the runtime access was quick and simple, so the agent could have access to the information promptly. In particular, the cook should be able to return to the correct path whenever interrupted by the user. To save this information the following data structure was created:

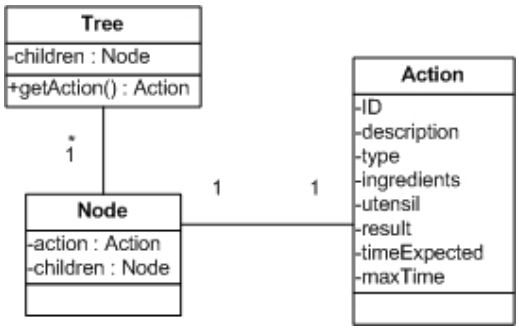


Figure 13 – Structure of the Tree

The tree has list of nodes where each one contains an action and the respective children nodes. To turn this structure more effective, each action has a unique ID (identifier) and, while the tree

is being built, a list of the ID's (ordered) is being created as well. This last list is the one that the agent has access in runtime, keeping always the context of the current action ID.

So, to build this tree, it was created an algorithm, which is depicted below, that evaluates the several actions and orders them depending on their relations with each other:

- The root element of the tree is the last action of the recipe. In other words, the algorithm starts with the final result, for example, a Chocolate Mousse;
- Afterwards, we need to find the ingredients needed to achieve that result;
- Next, it is necessary to find the action that provides those ingredients (i.e. find the action where the final ingredients are the same as the ingredients needed by the last node).
- Add the action found as a new children node as illustrated in Figure 14.

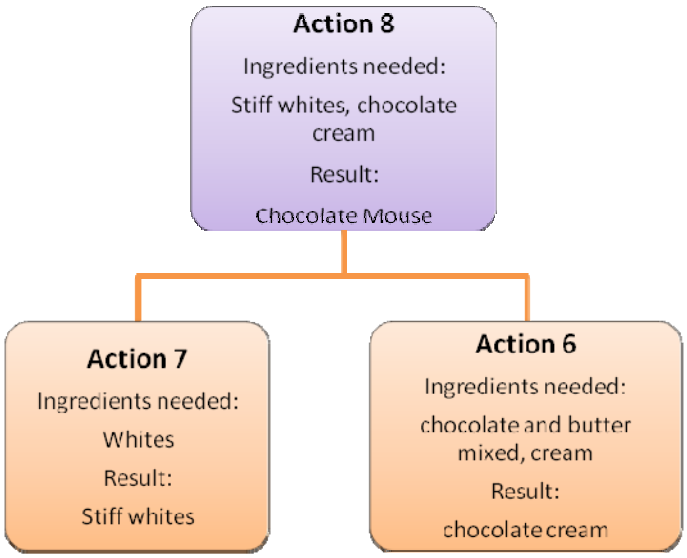


Figure 14 – Example of the tree after adding the child nodes

- Repeat these steps recursively for each node.
- The algorithm ends when the ingredients needed of a node are the basic ones (Figure 15).

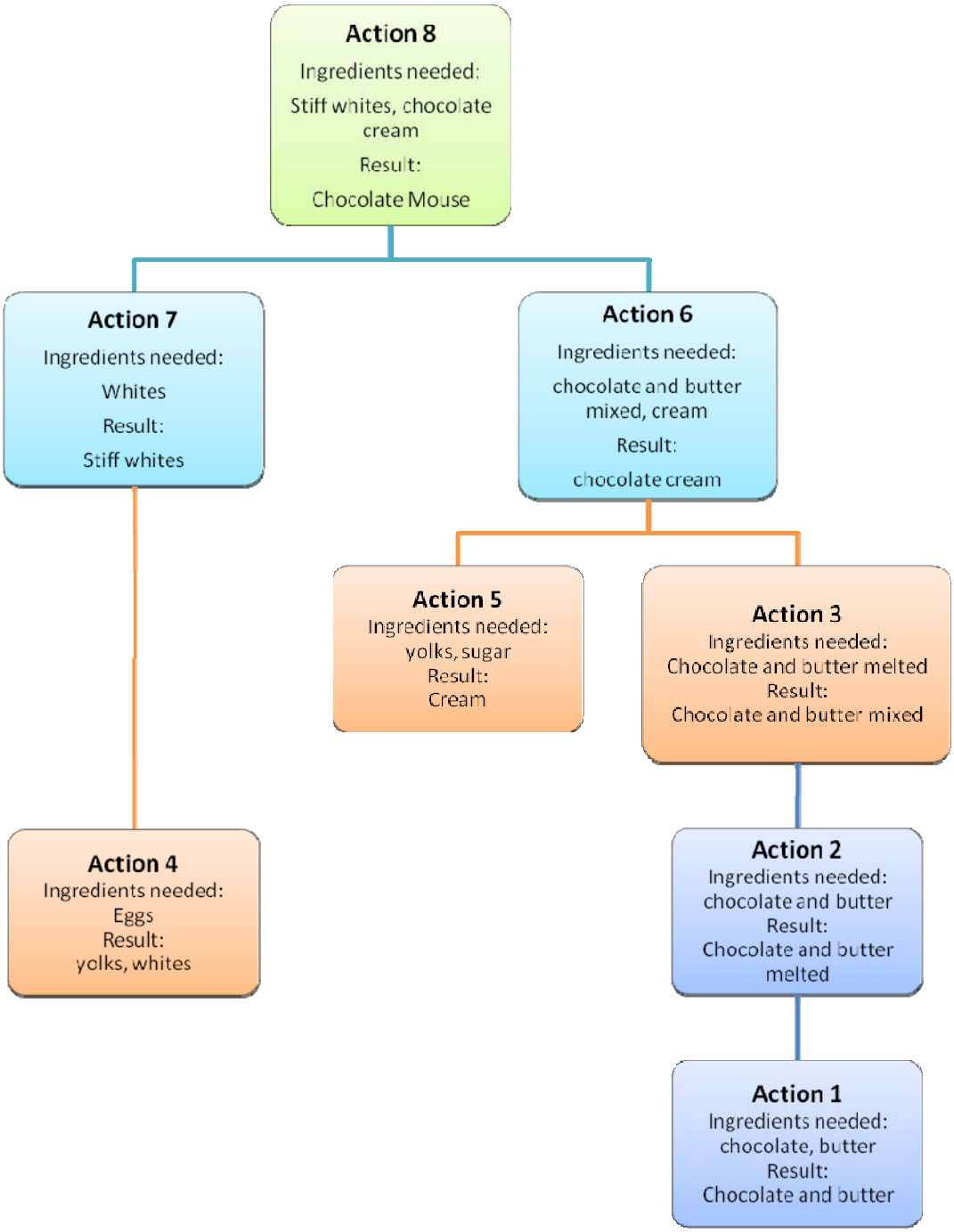


Figure 15 – Example of the tree after the last iteration

4.3. Demonstration Mode

As explained on Section 3.1, demonstration is one of the cook's possible interactions with users. This section pretends to explain how the strategies described can be implemented in the system.

In this mode, the agent tries to follow the methodology used in the real cooking programs where he attempts to explain a certain recipe to the user. The demonstration main cycle is illustrated on Figure 16 below.

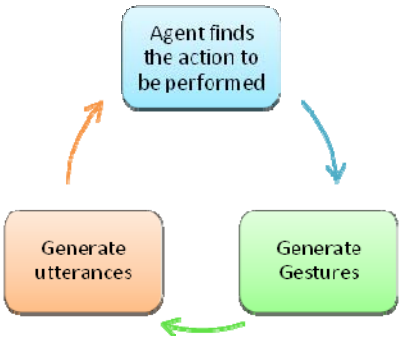


Figure 16 – Main cycle of the Demonstration Mode

So, the logic of this mode is quite simple. The agent has always present the path he needs to follow and, for each action, generates the appropriate gestures and utterances which must be as explicit as possible. The gestures' generation is made using an Animation Engine and the utterances are produced by a Language Engine. Both of them are described in detail on Section 4.5.

Concerning the strategies used in this interaction mode, adaptive pedagogical interaction was one of them.

To support this strategy two "sub-modes" or levels were created (Figure 17).

The first sub-mode is the "normal" one where the agent explains the recipe in a continuous mode. During this demonstration, the user has the opportunity to interrupt the agent whenever he doesn't understand some step of the recipe and ask him to repeat. Then, the agent tries to explain again, however in a different way. There are two difficulty levels visible on the type of explanation. The default one is the medium level where the agent explains the essential. When the user shows some difficulty in understanding the recipe the system switches to the easy mode where the explanations are more detailed.

Figure 17 – Demonstration Mode initial screen

In this mode, one aspect that was taken in consideration was the time that each sentence stays visible on the screen. It couldn't be too quick because the user might get lost and loose his interest, but it also couldn't be too slow because the user could get bored. So, to calculate the estimated time that each utterance stayed on the screen, some tests had to be made. Example texts were chosen and given for some people to read. The time was measured and, then, was calculated the mean time needed for each word (based on time spent on each sentence).

Therefore, for each utterance generated, it was calculated the respective time needed, and the demonstration only steps forward the next action when the time was reached.

Still, most inexperient users may think this sub-mode is a little hard to follow. Therefore, it was created another one: "Step-by-step". In this level is the user who defines when he wants to proceed and can even repeat the same action as many times as he wants. Thus, all the demonstration is adapted to the user's rhythm and not otherwise.

To support Attentional Guides strategy, this mode implements an important feature. Parallel to the demonstration cycle there is a panel with several icons which represent the actions and ingredients of the recipe. These icons become highlighted whenever the respective ingredients or actions are referred in the explanation (Figure 18). The ingredients are constantly giving place to the new ones that are created (intermediate ingredients). These icons help the user to memorize which ingredients and actions were used as well as their respective combinations so he can identify them more easily.

Figure 18 – Agent during a demonstration

4.4. Try-it! Mode

Here is where the user can put in practice everything he learned with the cook's demonstration. In this interaction mode the agent observes the user's movements and gives the appropriate feedback. Figure 19, below, explains the main cycle behind this mode.

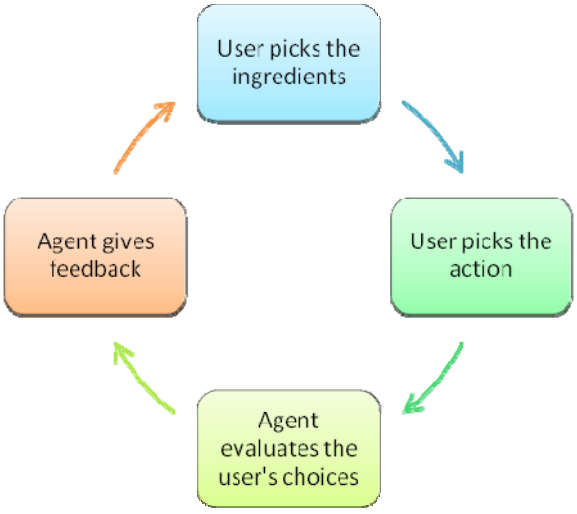


Figure 19 – Main Cycle of the Try it! Mode

All the logic of this mode was conceived focusing the user. The agent waits until the user picks the ingredients and action. Afterwards, he compares the user's choices with the expected ones and gives appropriate feedback.

As seen in Section 3.4.2, Feedback is one of the covered strategies.

Feedback can be used not only to inform the user if his choice was correct but also to prevent him from committing mistakes. This system uses them both.

So, whenever the user's choices are correct the agent claps his hands and shows his happiness, stepping forward to the next action (Figure 20).

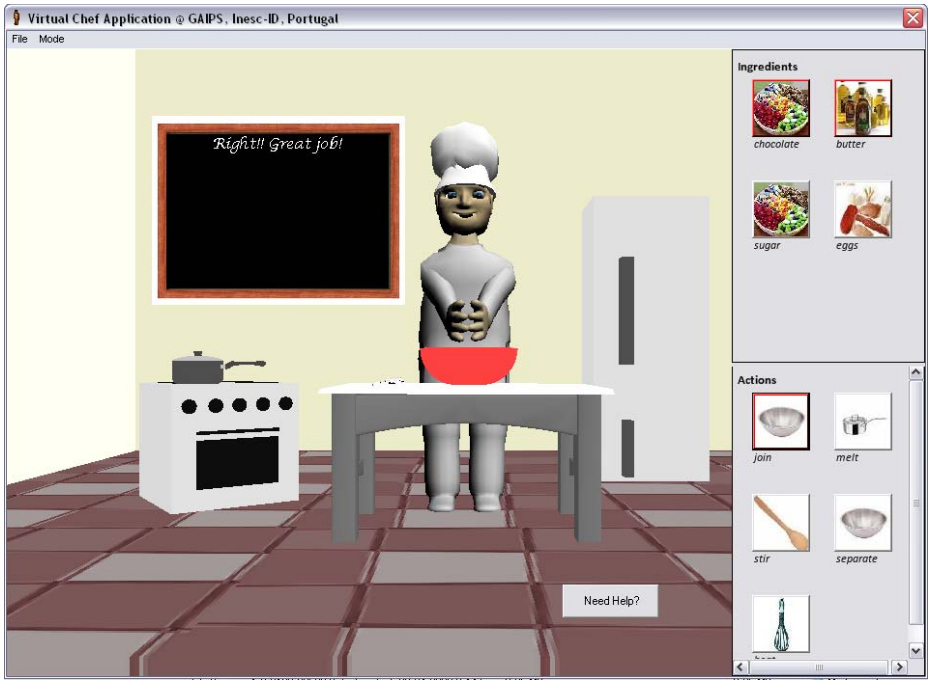


Figure 20 – Virtual Chef congratulating the user

However, when the user commits a mistake he makes a signal with his arms showing his disapproval, but always encouraging him to try again. ()

There was also the need to prevent the user from committing mistakes. Therefore, to support this strategy two features were implemented: the first one is having a Help button always present. Thus, whenever the user feels lost he may ask for help and the agent provides some useful hints. With these hints, the user may remember the action and, executes it without committing any mistake. The second strategy relies on time, i.e., it is defined an acceptable time for performing an action and, whenever the user exceeds that time, the agent assumes he is

having some kind of difficulty. To implement this solution was created a timer which its interval is updated according to the expected time of the current action. Whenever the timer exceeds the expected time of a certain action, the agent assumes the user is facing some troubles and tries to help him by providing some tips or even performing another demonstration.

Figure 21 – Agent showing his disapproval to a user's action

These hints or tips support not only the Feedback strategy but also the Adaptive Pedagogical Interaction one. This happens because the tips are created based on the type of mistake. For instance, if the user chooses the correct ingredients but the wrong action, the agent will give a tip concerning the action. Moreover, different types of help are triggered off also depending on the number of mistakes of the user as explained next:

- When the user makes one mistake the agent just informs that he made a wrong decision and incentives him to try again;
- At the second mistake, the agent gives him a hint;
- If the user still makes a wrong choice for the third consecutive time, the agent performs a "live demo" of that specific action just as in Demonstration Mode.

4.5. The Character

In order to perform all the strategies described on the previous sections, the agent has the constant support of two important features: utterances and gestures.

Utterances are generated by a Language Engine [18] and it only produces sentences that will be written on the screen. It will not be used synthesized speech. On its turn, gestures are produced by an Animation Engine. Both elements are used in parallel having, for each action, simultaneously an utterance and a gesture.

By having different roles, these two features complement each other. It is proved that people tend to lose their attention if they spend some (considerable) time reading something, for example, slides on a presentation. In this case it is possible to apply the same theory. If the user only had sentences to read during the entire recipe, the probability of getting lost or giving it up would be much higher. Thus, the gestures not only have the pedagogical role, but also the attractiveness one. The animations that are produced may help the user to keep interested and might enhance his enthusiasm.

The following sections will describe both Language and Animation Engine modules.

4.5.1 Animation Engine

The animation engine is the responsible for all the agent's gestures.

To perform the gestures, it is used a module which is a part of a master thesis work called *Gesticulation in Virtual Humans* [18].

Before this module can actually execute the gestures needed, it is necessary to create a gestures' library.

At the beginning, the system receives a XML file with all the gestures definitions.

To construct this file it was taken in consideration all the information needed by the gestures' module. Next, will be described the information contained in the gestures definition file, followed by an example of it correspondence in the XML file.

The first information needed is the **name** of the gesture**.** The system would from this moment on recognize this name as a gesture and it is defined in the XML file using the tag name, as it is shown on the next line.

```
<name>greeting-right</name>
```

Other important information for the representation of the gestures is the **position.** The position represents the final position of the arm. It is defined in Cartesian coordinates in three-dimensional space, using world references. Under the tag position in the XML file we can see three values separated by semicolons which correspond to the position that the arm should be in the end of the gesture.

```
<position>-32,6;90;-5</position>
```

Another feature of the Gesture module to be taken into account is the **orientation**. It defines the orientation of the hand through the palm axis using yaw, pitch and raw. These three values are inserted in the tag orientation in the XML file as we shown in the example below.

```
<orientation>-90;25;90</orientation>
```

The feature **hand** defines whether the gesticulation keyframes applies to the left (1), right (0), or both hands (2). In this last case it is defined a dominant hand and symmetrical positions are calculated for the non-dominant one. To define in XML file which hand to use we use the tag “hand” with the value of the hand or both. In the example given the gestures is made using the right hand.

```
<hand>0</hand>
```

In order to know how long the gesture would last, the gesture module has a support to indicate the **time,** in other words, it indicates the duration of the gesture in seconds. For each gesture we have to define in the XML file how long it lasts. This is done with the next instruction. In this example the gesture would take one second before it stops.

```
<time>1</time>
```

Other feature that is part of the specification of the gesture module is **shape.** Although isn’t much explored in this system, with this feature it is possible to define any Portuguese Sign Langue hand shape. The default (and used) one is the “neutral” shape which is possible to see on Figure 22. In the example below it is used the neutral shape.

```
<shape> </shape>
```

Each gesture may contain 1 or 2 frames. Each **frame** has all elements previously described (position, orientation, hand, time and shape) and it refers to only one hand (or both when symmetrical). If the need to use both hands (with non symmetrical positions) at the same time urges, two different frames have to be defined (one for each hand). Below we can see its representation on the XML file.

```
<frame>
	<position>-32,6;90;-5</position>
	<orientation>-90;25;90</orientation>
	<hand>0</hand>
	<time>1</time>
	<shape></shape>
</frame>
```

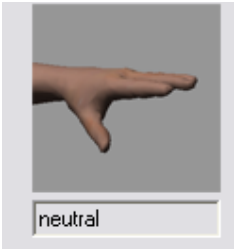


Figure 22 –Neutral hand shape

After creating this structure, there was the need to fill it with the definition of some gestures. The only way to delineate them was by experimentation. The position and orientation of each and every gesture was calculated through an interface present in the gestures module which is illustrated below. Figure 23 shows the virtual agent on the initial position while Figure 24 shows the cook after a position was defined.

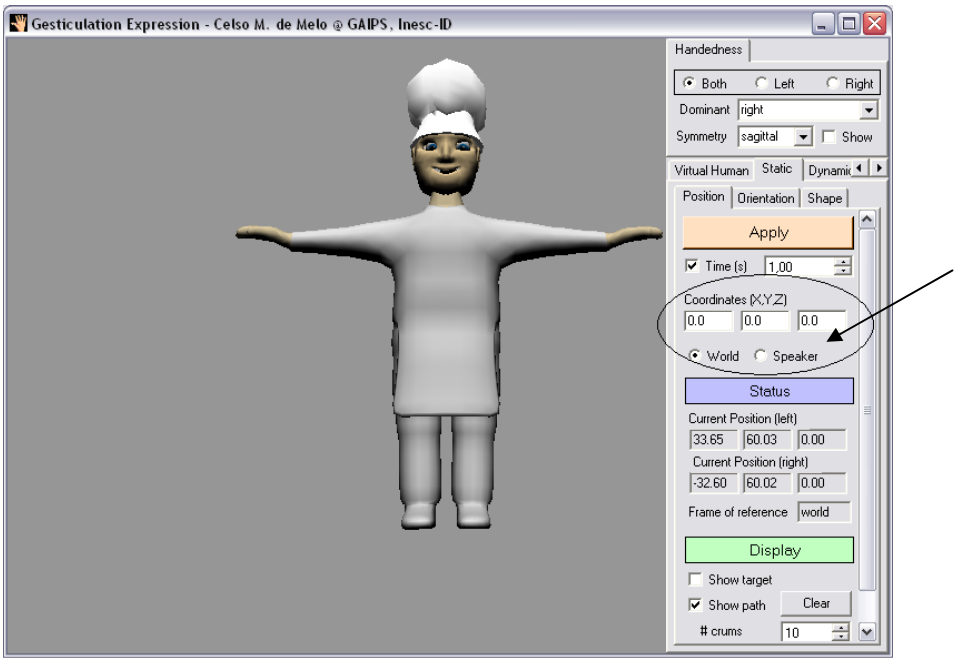


Figure 23 – Interface before any position values were chosen

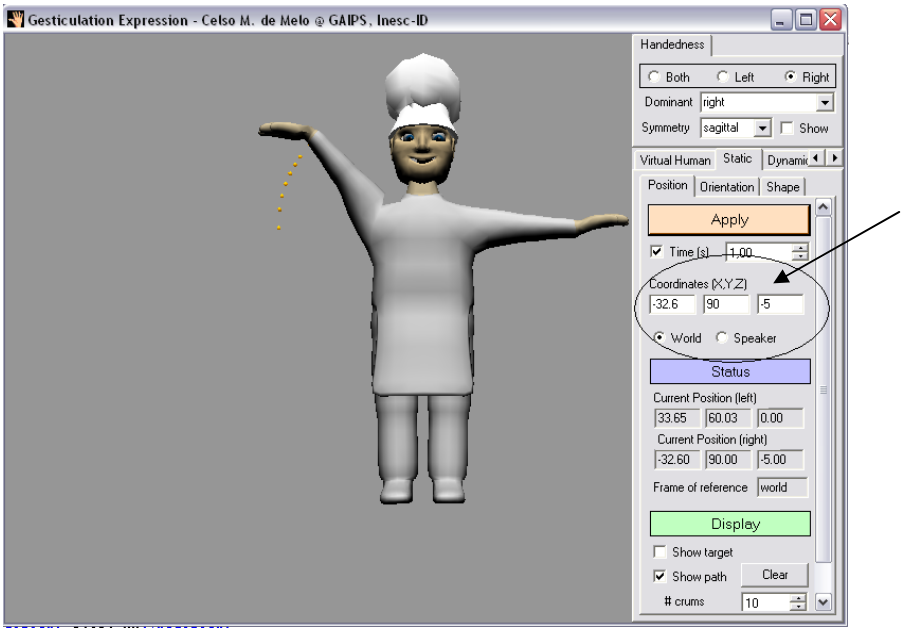


Figure 24 – Interface after choosing the position values

As described before, it is also possible to define the hand orientation values. The following image (Figure 25) illustrates the interface after some orientation values were chosen.

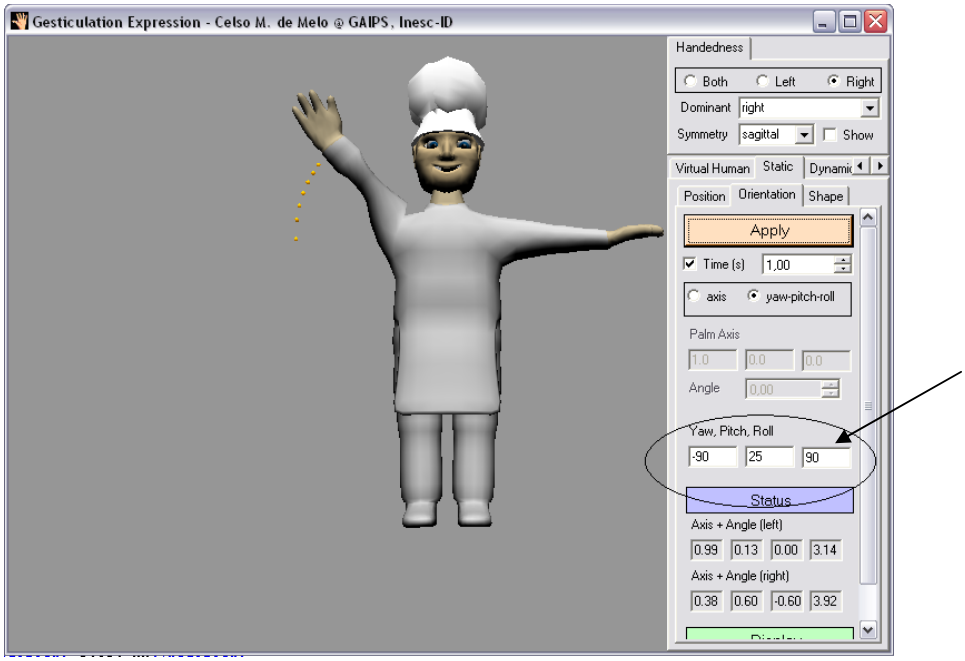


Figure 25 – Interface after choosing the hand orientation values

This gestures definition method was used for every gesture in the application. After all gestures were defined, they were inserted into the XML gestures file which is possible to see an excerpt of it on Appendix C.

To manage all the actions and gestures, there are two important classes: *ActionManager* and *GesturesManager*. The first one has access to the tree of actions and is responsible for getting the action the agents needs. The *GesturesManager* is responsible for getting the gesture associated with the current action, for the animation module to perform it.

This correspondence can be made because at the beginning of the application, it is loaded a file with all the relations between gestures present in the gestures' library and the actions. For instance, the gesture "put" is related to the action "put", "add" or "join". Thus, when the *GesturesManager* receives the action to be explained, it looks for the respective gesture on the "gestures' library" and returns it.

Thus, every time the main cycle needs to execute an action, it will check what gesture(s) has to be used. When that information is retrieved, the application will call the method of the animation module responsible for the gestures execution.

This method is called for every frame of each gesture, resulting, in the end, in a continuous movement trying to be the most similar to a human gesture.

4.5.2 Language Engine

The language engine is responsible for generating all the utterances. The module uses a part of an existing work called *Actor with Attitude and Component Specification for Computational Autobiographic Memory* [19]. This engine is based essentially in three files: *lexicon*, *types* and *templates*.

In *Lexicon* is possible to define all words and synonyms so the utterances are generated as dynamic as possible. For example, to specify synonyms for the word "correct" we need to include these lines:

```
correct,,,,correct,
perfect,,,,correct,
right,,,,correct
```

In *types*, all types and function defined in templates are declared. For example, to declare a variable named "firstAction" and a function called "first" we need to add the following lines:

<class name="firstAction" extends="localContext"/>

<class name="first" extends="Internal" />

All variables and types must be declared as "localContext", while functions as "Internal".

Finally, in *templates* is defined the whole structure of each type of utterance. The position of the variables, the gender of the "speaker", the "dynamic" words (those that have synonyms defined) are examples of what we can specify. It is even possible to define functions as an integrated part of a type of an utterance. These functions can define various utterances which are picked

randomly by the engine. It's possible to specify several levels of complexity. We can create very simple modules (each returning a sentence) and, then, combine them to compose a more complex phrase.

So, every time the agent needs to display a certain sentence, this module should be invoked. Figure 26, below, illustrates the basic structure of the language engine.

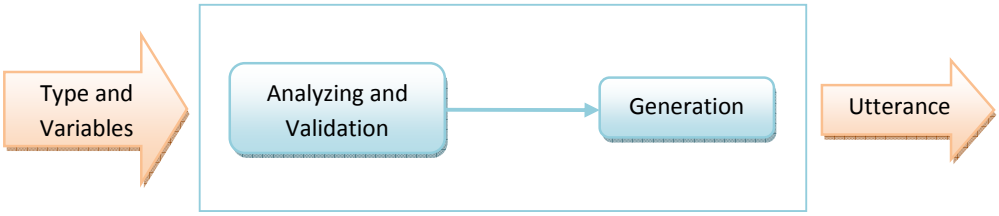


Figure 26 – Basic structure of the Language Engine

This language engine has the goal of generating an utterance based on the variables received. This module is separated in two different stages:

Analyzing and Validation – First of all, this stage checks if the types and variables received are correct by assuring they are according to the ones declared and defined on *types* and *template* files. If everything is as expected, the respective template is filled in with the received variables, as well as the synonyms, when needed.

Generation – This final stage generates the final utterance according to template previously constructed.

To integrate this module it was created a small application to allow the communication with the main system. All information is exchanged through sockets. The main system initializes a connection with the language engine sending the current action and the respective ingredients as arguments and waits for an answer with the utterance generated. The following examples explain how it is possible to construct a sentence using this language engine.

.

- **Example 1**

```
Type(value:greeting)
-> Utterance(value:(Synonym(semCat:hello),"!"))
```

This is a very simple example where is generated a sentence with a synonym of "hello". The result can be "hello" or "hi", etc. (depending on the synonyms defined)

- **Example 2**

```
Type(value:greeting) $Y=you(value:$V)
-> Utterance(value:(Synonym(semCat:hello)," ",$V,"!"))
```

This example is a little more complicated than the other. Besides the synonym, it receives a variable. The resulting sentence will contain a synonym of “hello”, followed by the variable received (in this case is a name of a person) and finally the character “!”. The result could be: “Hi Sara!”

- **Example 3**

```
Type(value:firstAction) $A=ac(value:$AV) $T=what(value:$TV)

->first() Utterance(value:(Lex(semCat:ppn_2,number:sg,gender:mask,case:nom),"
")) Utterance(value:(Synonym(semCat:need), " ")) Utterance(value:$AV)
Utterance(value:" the ") Utterance(value:$TV) $A(value:$AV) $T(value:$TV)
```

This is a complex type. As we can see, it is composed by several simple modules, but returns a much complex sentence.

Most of the modules were already explained before (synonyms and variables). However, we can find two new components: “Lex” and the function “first”. Functions can define a set of utterances, which are randomly chosen into the final sentence. This is how can be defined a function:

```
first()

-> Utterance(value:"To start ")

first()

-> Utterance(value:"To begin ")
```

Just as any other type, functions can be defined, as well, with synonyms, variables or any other component instead of a plain text utterance.

Finally the “Lex”. This component allows us to introduce a person pronoun in the sentence. In this case, is being used the pronoun “you”: “ppn_2” means second person pronoun, and “gender_mask” means masculine (in this case the gender is irrelevant).

At the end, the resulting utterance would be something like: “*To start you need to add the chocolate.*”

As we can see with these three examples, this language engine can be a “world” of sentences, we just need a little of creativity in their creation.

4.6. Concluding Remarks

In this section was explained how it was implemented the conceptual model defined on Chapter 3.

First, it was presented a basic architecture of the system.

Then, the following sections described each of the system’s components and their implementation choices.

To support the mind of the agent, the programming environment used was .Net with the programming language C#. To support the gestures module the programming language and

environment was the same, however another technology was introduced: DirectX. In order to communicate with the Language Engine, it was created a small application in Java, since this module was already built in it.

On Appendix B it is possible to find a schema describing all the important classes used in this system, as well as the relations between them and the external modules.

Chapter 5

Evaluation

This chapter describes the experiments and tests made in order to evaluate the implementation depicted on the previous chapter. The first section will explain the methodology used all over the experiments. Then, it will be exhibited the results achieved with the respective analysis and, finally, in the last section it will be the presented the conclusions reached.

The scenario that will be used in the tests consists on a user interacting with the virtual chef trying to learn a certain recipe. Due to its several modes, the scenario is appropriate to any kind of user with different levels of knowledge. The main goal is to let the user interact with the virtual chef and see if the learning process is good and effective. In particular, and using the initial hypothesis, we are trying to prove that, by interacting with virtual chef, the user will be the subject of his tutoring strategies and, therefore, he will experience a better learning process, more motivating and enjoyable.

5.1. Methodology

To validate the hypothesis presented above, some tests were performed. There were two different kinds of tests, which were equally distributed by all participants. All tests had the same goal: see if the virtual agent can positively influence the learning process.

The experiment consisted in two different phases: in the first one the agent performed a demonstration of a recipe. This demonstration could be performed in the continuous mode or in the step-by-step mode.

The second phase consisted in the user performing the recipe that had just learned. Thus, measuring some aspects such as number of error committed or how many times the user asked for help, it was possible to evaluate if the demonstration made by the agent was effective and helpful and which of its “sub-modes” revealed to be more useful in the learning process.

5.1.1 Measurements

According to the challenge presented by the research question, the tests that will be described next pretend to measure two distinct aspects: the user's learning process, as a consequence of his interaction with the virtual chef in the second phase of the experiment (after seeing a demonstration); and the user's enjoyment interacting with the interface.

By evaluating the user's learning process, we will, consequently, be analyzing if the agent's tutoring strategies are being effective and helpful in both modes. There are two ways of assessing this variable. The first and immediate one is measuring quantitative aspects such as time or mistakes the user committed while performing a recipe after seeing the virtual chef's demonstration. In this case, it was measured the following aspects:

- Number of mistakes committed by the user
- Number of tips given by the agent
- Number of demonstrations given by the agent
- Number of times the user asked the agent to repeat
- Number of times the user exceeded the expected time
- Number of times the user asked for help ("Help" button)

The second way of evaluating this variable was by using a questionnaire. At the end of the tests, it was distributed a questionnaire that tries to assess the impact of the virtual agent and it's strategies on the user's interaction. It is possible to see this questionnaire on Appendix D.

Concerning the second variable (user's enjoyment interacting with the interface), the only way of evaluating it is, besides the questionnaire mentioned above, by observation. During the experiments, the users were carefully observed with the purpose of getting the most information as possible concerning the user's reactions and comments. Thus, it was possible to know what the user was thinking about the system and his level of enjoyment.

5.1.2 Participants

A total of 12 participants took part in this experiment (7males and 5 females). All of them were students on the IST – Technical University of Lisbon and their ages were comprehended between 20 and 27 years old. All the participants were native Portuguese and the only criterion followed in the selection of them was to choose people that have never interacted with Virtual Chef before.

The participants were divided into two groups of six each where the difference between them relied on the first phase: The first group interacted with the continuous demonstration mode while the second one interacted with the Step-by-Step demonstration mode.

All the tests were performed in the facilities of IST – Technical University of Lisbon, Taguspark campus.

5.1.3 Setting

As mentioned on section 5 the experiment consisted in two distinct phases.

In the first one the user watched a demonstration of a recipe made by the Virtual Chef. This demonstration can be performed in two different modes:

- Continuous mode, where the agent performs a demonstration continuously, allowing only the “repeat” option.
- Step-by-step mode where the agent performs a demonstration according to the user’s will.

The demonstration mode is, therefore, the independent variable.

In the second phase the user needed to interact with the Virtual Chef’s interface and execute the recipe that was previously explained by the agent.

The recipe used in the tests was the same for both modes: a Chocolate Mousse.

5.1.4 Procedure

Each participant was provided with a brief description of the Virtual Chef’s system as well as the procedure that would be followed.

Afterwards, they were seated in front of the Virtual Chef and whenever they were ready the test would begin. During the entire test, the experimenter was near the user in order to extract as most information as possible so he could be able to measure the metrics defined on Section 5.1.1. At the end, the user was asked to fill in a questionnaire (Appendix D) in Portuguese Language. The questions used in this questionnaire not only tried to evaluate the Virtual Chef’s interface and the interaction with it but also the appropriateness of the tutoring strategies used.

Each participant concluded his evaluation session in approximately 20 minutes.

5.2. Results

The last sections described the methodology followed all over the experiments. After all tests were performed, it was analyzed the information that came out of it.

In this section, it will be presented the results of our preliminary experiments. To start, will be presented the results that assess if the agent’s tutoring strategies were adequate to this model. This evaluation was obtained through our questionnaire results as well as measuring the quantitative metrics described on section 5.1.1. Then, concerning the user’s enjoyment, the evaluation was made not only through the questionnaire but also by observation of the user’s behaviour during the test and by their comments at the end of the experiment.

Concerning the questionnaire, the user had to answer the questions using a five-level Likert scale where 1 is considered to be total disagreement and 5 total agreement. This questionnaire can be divided into two main groups. The first five questions try to infer the usability of the system and its components while the following ten questions pretend to evaluate the agent's tutoring strategies, which is the main goal of this work. So, in order to analyze the results achieved concerning the effectiveness of the tutoring strategies chosen, we will only take in consideration the questions from 6 to 15.

Questions 6, 7 and 10 pretended to evaluate the demonstration mode, in particular, if it's hard to follow the explanations or if they are easy to understand. Figure 27 presents the results obtained in these questions. It is possible to see that several people though it was a little hard to follow the agent, not due to the speed of explanations but mostly due to the big amount of steps. However, most of the participants agreed that the explanations were very explicit and simple. Finally, a considerable part of the participants affirmed that, at the end, they were able to understand the recipe presented.

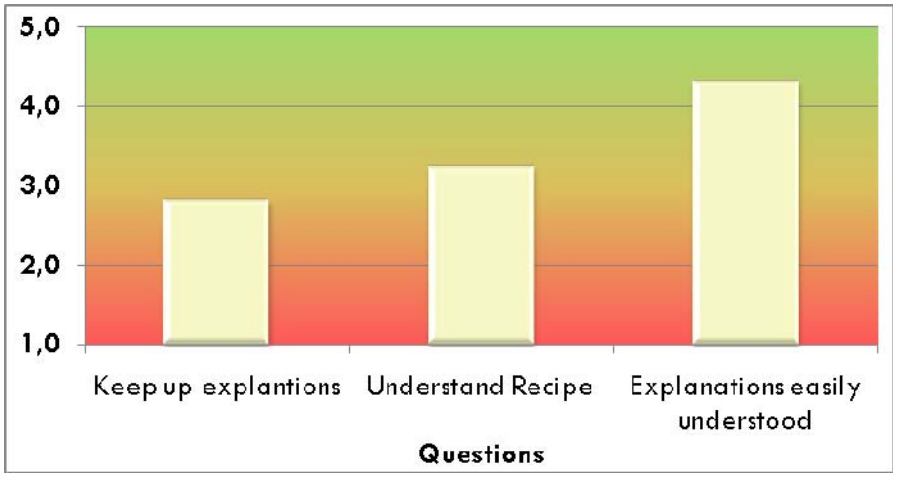


Figure 27 – Mean values obtained from the user questionnaire from Question 6, 7 and 10

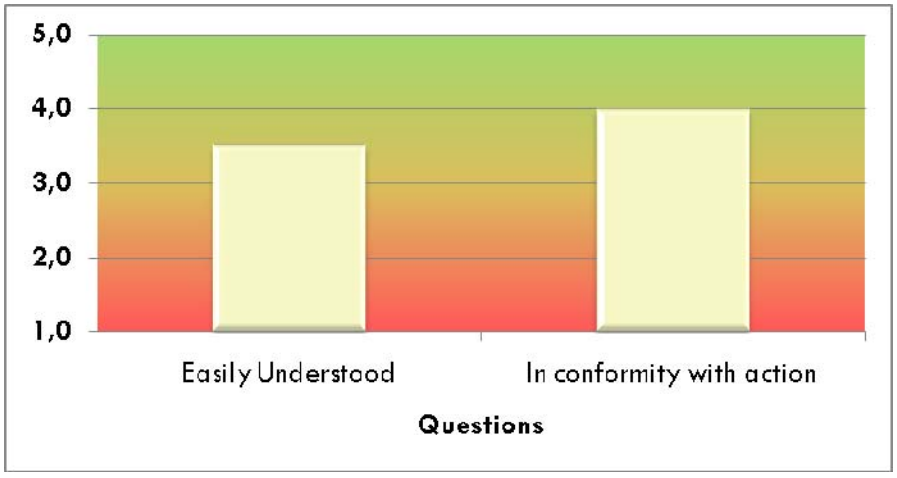


Figure 28 – Mean values obtained from the user questionnaire from Questions 8 and 9

Questions 8 and 9 pretended to evaluate if the gestures used in the interactive demonstrations were explicit and appropriate. presents the results of these questions. Most of the users agreed the gestures were adequate to the actions and a great part of them thought the gestures were also easy to understand.

Questions 11, 14 and 15 tried to infer if the Chef is effective on assisting and helping the user when he is in trouble. The majority agreed that the agent always help them when they were facing some difficulties and the tips were adequate and useful. Moreover, a considerable part of the users agreed that, in the end, they were able to learn the recipe. Figure 29, below, presents the results relative to the questions described before.

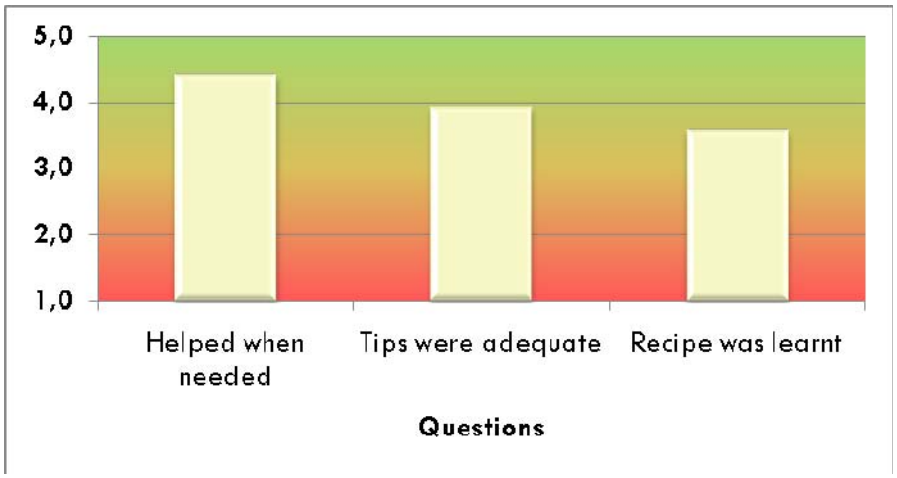


Figure 29 - Mean values obtained from the user questionnaire from Questions 11, 14 and 15

Figure 30 - Mean values obtained from the user questionnaire from Questions 12 and 13

Finally, questions 12 and 13 pretend to asses if the quantity of tips provided by the agent were adequate. Observing is possible to conclude that the majority of the users thought the tips given were sufficient and, therefore, the agent doesn't need to provide help more often. Although the agent provides several tips (whenever needed), all users agree that the agent wasn't intrusive at all.

As said before, the effectiveness of the Chef's tutoring strategies was assed not only using the questionnaire, but also using the quantitative metrics described on Section 5.1.1. As previously described, these metrics are assessed during the second phase of the experiment where the user tries to perform the recipe previously learnt. Thus, it was possible to infer not only if the explanations on the demonstration were effective, but also if the type of demonstration mode influenced the user's learning process and which of them influences the most. The results achieved are illustrated on Figure 31.

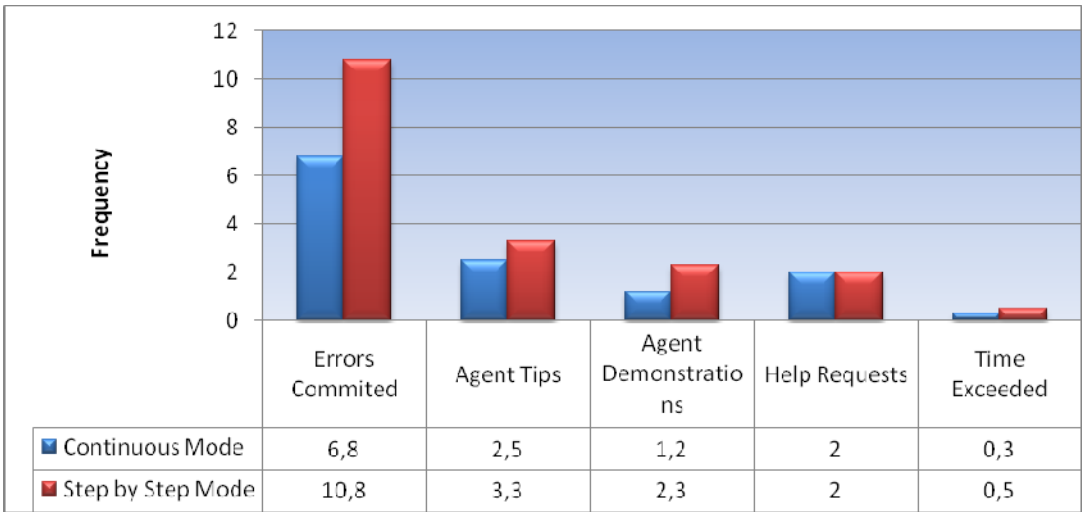


Figure 31 – Mean values for errors committed, agent tips, agent demonstrations, help requests and time exceeded

The idea of creating the Step-by-Step mode was to provide an alternative to the less experient users. So, supposedly this mode was much easier to follow than the continuous one. But, unexpectedly, the users who watched the demonstration in the continuous mode presented better results than the ones in step-by-step mode. So what can be the reasons for this result?

Well, in step-by-step mode, the fact of having the control over the agent's explanations may increase the user's distractions. This happens because the user isn't pressured to follow the agent's demonstration and can easily be influenced by external factors, having conscience that the agent will only proceed whenever he wants. So, the fact of the agent only showing another step after the user's will, may not always be positive, since the distraction that the user may suffer can also make him forget the steps he had just seen until that moment.

Another possible explanation for these results was that some users, by knowing they would interact with the agent to perform a recipe, became anxious and wanted to end the demonstration as quick as possible. Therefore, the users weren't paying much attention to agent demonstration and when they got to the Try-it! Mode couldn't remember most of the steps.

Concerning the user's enjoyment, as explained before, It was assessed by two different methods: through the questionnaire and by observation during the experiment.

Only questions 1 to 5 of the questionnaire intends to evaluate the user's enjoyment, by taking into account the opinion of the users concerning the interface and usability of the Virtual Chef.

Questions 1 and 5 tried to evaluate if the system is easy to interact with. Figure 32 illustrates the results achieved. As we can see, most of the participants thought that, besides being easy to use, the type of interaction (the buttons in the lateral panel) was appropriate.

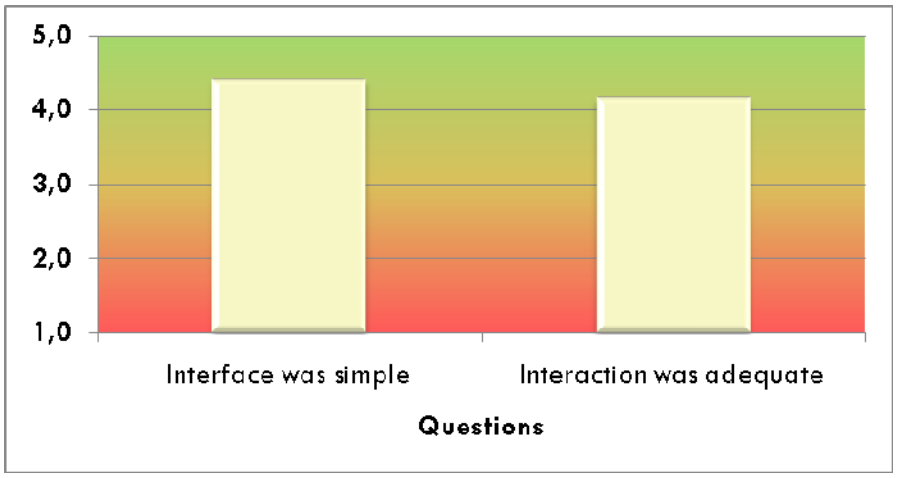


Figure 32- Mean values obtained from the user questionnaire from Questions 1 and 5

On the other hand, questions 3 and 4 tried to assess if the user thought the agent was pleasant. Observing Figure 32, is possible to verify that the majority of the users though the agent wasn't boring at all and liked to interact with him.

The other way to evaluate the user's enjoyment was by observation. During the entire recipe the experimenter was monitoring every step of the user paying attention to each reaction.

The participants tended to be more excited in the second phase. It was much more challenger than just watching a demonstration. Moreover, the users tend to react quite well when the cook congratulated him (using sounds) whenever he made a correct choice. The fact of the demonstration mode doesn't have any sound at all may be a possible cause for the preference in using the Try It mode.

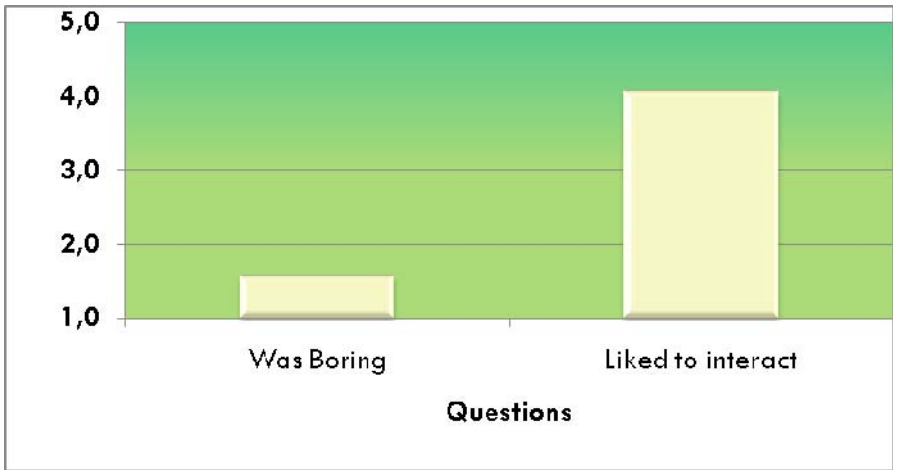


Figure 33 - Mean values obtained from the user questionnaire from Questions 3 and 4

5.3. Concluding Remarks

The main question behind this experiment was to see if the strategies used by the Virtual Chef were appropriate and could result in a better learning process, more pleasant and enjoyable.

In this chapter we started by introducing our research question for this experiment. Afterwards, the methodology used was described as well as the measures that would be taken into account. Finally, the preliminary results were presented and discussed.

The results can be organized in two different groups: the first one tries to asses if the tutoring strategies chosen and implemented were adequate to the Virtual Chef, while the second one evaluates the user's enjoyment during the interaction with the system.

Concerning the first aspect, we observed that the participants that were submitted to the continuous demonstration mode presented better results than the ones that interacted with the step-by-step one. Although some considerable mistakes appeared on both modes, this cannot be a negative aspect of the Virtual Chef since the system was new to all the users and the recipe had some considerable steps. Mistakes were expected, but the point to prove was that the users could recover their mistakes with the help provided by the agent and therefore, complete the task they began. With the help of the questionnaire we could conclude that the assistance provided by the agent turned out to be adequate to the difficulties the users were facing while performing the recipe.

Another aspect that was possible to conclude was that the step by step mode turned out to be a “two way road” because, in one way the fact of the speed of the explanations being according to the user’s will, can help them to better understand each step but, in the other hand, the possibility of distraction is much higher than in the other mode. Despite this, the inclusion of this mode in the system still seems appropriate because these distractions didn’t affect all the users and some of them even provided good results.

However, some tests can still be made. These were preliminary tests and, in order to conclude if the presence of the Virtual Chef really makes a positive influence on the user’s learning process, it could be performed a similar test to the ones made, but wouldn’t have a demonstration phase. The user only had to read the recipe in a paper and, then, he had to execute it interacting with the Try-It Mode. By assessing the number of mistakes and all the metrics used and comparing with the results achieved in this experiment, it was possible to conclude if the presence of the agent had really made a difference.

Concerning the other variable, the user’s enjoyment, it was possible to conclude that users thought the agent was pleasant and in, some cases, even funny. The reaction was quite good. However the introduction of voice and/or sounds would provide an even better experience to most of the users.

Chapter 6

Conclusion

The work presented in this document pretends to provide a contribution in animated pedagogical agents. They are becoming more a more an important tool in learning environments. They can provide proper advices and, in some cases, perform practical demonstrations; they can even replace a human teacher. These embodied pedagogical agents, when lifelike and believable, can turn the learning process more effective, motivating and enjoyable.

This dissertation addressed the following research question: *What are the main requirements for building an animated pedagogical agent, in particular, a virtual chef?* The research was focused on pedagogical strategies for animated agents, starting from the hypothesis: *By analyzing the existent animated pedagogical agents and its tutoring strategies, if the virtual chef employs the strategies that best adequate to the cooking domain, then the user may experience a better learning process, more motivating and pleasant.*

Trying to answer the research question, several literatures of animated pedagogical agents were studied. The most conceived examples of pedagogical agents were analyzed as well as their tutoring strategies. Afterwards, and having these strategies in mind, it was built a model of a pedagogical agent, in this case a virtual chef. Several domains in pedagogical agents are still unexplored and cooking is one of them. There are emerging lots of new cooking programs as well as several sites on the Internet with millions of recipes. This can be a great opportunity to produce something new in the area of animated pedagogical agents.

Then, after the tutoring strategies were analyzed, it was idealized a model of a virtual chef that could teach recipes, just like the ones on those TV programs. The strategies and elements that aid learning that we thought could better fulfill the virtual chef's demands were: Adaptive Pedagogical Interaction, Interactive Demonstration, Attentional Guides, Feedback and Conveying and Eliciting Emotion.

After implementing a system that could support the model idealized, a preliminary evaluation was performed. The first measure to evaluate was the user's learning process and second one was the user's enjoyment. Although the number of tests were limited, it was possible to take some positive conclusions. The majority of the participants liked to interact with the agent and even declared that they had learned how to perform the recipe with the agent's help. However, some other experiments could be done so it was possible to assume that this virtual chef can really be an asset to the user's learning process.

With this thesis it was possible to extract the best tutoring strategies to follow when conceiving an animated pedagogical agent, in particular, a cook, and the importance they may have on learning environments. Although the preliminary results were limited, it is possible to

conclude that our hypothesis may be correct and some future work in this field can reveal to be very relevant in learning environments.

6.1. Future Work

This document described the several steps for building an animated pedagogical agent, in this case a cook. The strategies a cook needs to support were all described and, based on that, an agent was built. Although all strategies were implemented, some are very limited. If well explored, these features may enhance a lot the system's acceptability and results.

As described before on Section 3.3.1, it is essential that a cook performs interactive demonstrations. A kitchen may be considered a complex environment with all the utensils and ingredients present in it. It was interesting to add some movement to the Virtual Chef besides the gestures. The cook should be able to walk through the environment during his explanations, such as moving towards the oven whenever is necessary to melt or heat something. This feature can increase the agent's believability and therefore can enhance the user's learning and enjoyment.

Feedback is one of the agent's strongest aspects. However there is one important feature that could be introduced: facial expression. Virtual Chef only uses nonverbal feedback through gestures that represent his agreement or disagreement. By entering facials expressions, the feedback would become much more complete and less intrusive. For instance, the agent could prevent some mistakes but just making a sign with his eyes showing the user he might be following the wrong way.

Another aspect that should be taken in consideration is the synthesized speech. Some users explained it was hard to follow the gestures and read the sentences at the same time. Therefore, people tend to focus only in one of them, generally the utterances. If the cook could speak, users would limit to listen to him and could pay all attention to the gestures which are an important guide. Besides, the fact of the application having sound would increase the enthusiasm of the user, instead of just looking at a silent demonstration.

Finally, it was interesting if the agent could actually manipulate the ingredients. This would give a more realistic impact of the system as well as turning it much more interesting and pleasant to interact with.

This dissertation tried to provide a contribution to the area of pedagogical agents, in particular, pedagogical agents in the cooking domain. This work meant to be innovative, since it doesn't exist any other work using this domain. Moreover, by trying to discover what where the strategies that best suit a virtual chef, a great study on tutoring strategies was made. This can be useful not only to a virtual chef. By knowing and understanding each of the strategies it is easier to apply them correctly in any domain.

Although this Virtual Chef may not be as complete as desired, it can be a great point to start entering some new areas in pedagogical agents.

References

[1] Cassell, J., et al. “Animated conversation: rule-based generation of facial expression, gesture and spoken intonation for multiple conversational agents.” *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques.* ACM, 1994. 413-420.

[2] Graesser, A. C., K. Wiemer-Hastings, P. Wiemer-Hastings, e R. Kreuz. “AutoTutor: A simulation of a human tutor.” *Journal of Cognitive Systems Research*, 1999: 35-51.

[3] Graesser, A. C., P. Chipman, B. C. Haynes, e A. Olney. “AutoTutor: An intelligent tutoring system with mixed-initiative dialogue.” *IEEE Transactions in Education.* 2005. 612-618.

[4] Gulz, A., e M. Haake. “Design of animated pedagogical agents - A look at their look.” *International Journal of Man-Machine Studies* 64 (2006): 322-339.

[5] Johnson, W. L., E. Shaw, A. Marshall, e C. LaBore. “Evolution of user interaction: the case of agent adele.” *Proceedings of the 2003 International Conference on Intelligent User Interfaces.* Miami: ACM, 2003. 93-100.

[6] Johnson, W. L., J. W. Rickel, e J. C. Lester. “Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments.” *International Journal of Artificial Intelligence in Education*, 2000: 47-78.

[7] Lester, J. C., e B. A. Stone. “Increasing Believability in Animated Pedagogical Agents.” 1997. 16-21.

[8] Lester, J. C., J. L. Voerman, S. G. Towns, e C. B. Callaway. “Cosmo: A life-like animated pedagogical agent with deictic believability.” *IJCAI97 Workshop on Animated Interface Agents: Making them Intelligent.* Nagoya, 1997.

[9] Lester, J. C., J. L. Voerman, S. G. Towns, e C. B. Callaway. “Deictic Believability: Coordinated Gesture, Locomotion, and Speech in Lifelike Pedagogical Agents.” *Applied Artificial Intelligence*, 1999: 383-414.

[10] Lester, J. C., S. A. Converse, B. A. Stone, e S. E. Kahler. “Animated Pedagogical Agents and Problem-Solving Effectiveness: A Large-Scale Empirical Evaluation.” *Proceedings of the Eighth World Conference on Artificial Intelligence in Education.* Kobe, 1997. 23-30.

[11] Person, N. K., e A. C. Graesser. “Pedagogical Agents and Tutors.” In *Encyclopedia of Education*, de J. W. Guthrie. New York: Macmillan.

[12] Rickel, J., e W. L. Johnson. “Integrating Pedagogical Capabilities in a Virtual Environment Agents.” *Proc. of First International Conference on Autonomus Agents.* ACM, 1997. 30-38.

[13] Rickel, J., e W. L. Johnson. “Task-Oriented Collaboration with Embodied Agents in Virtual Worlds.” In *Embodied Conversational Agents*, de J. Cassell, J. Sullivan e S. Prevost. Boston: MIT Press, 2000.

[14] Ruttkay, Z., J. Zwiers, H. Welbergen, e D. Reidsma. “Towards a Reactive Virtual Trainer.” In *Intelligent Virtual Agents*, de J. Gratch, M. Young, R. Aylett, D. Ballin e P. Olivier, 292-303. Spinger, 2006.

[15] Shaw, E., W. L. Johnson, e R. Ganeshan. “Pedagogical Agents on the Web.” 1999. 283-290.

[16] Rist, T.; André, E.; Müller, J.; “Adding Animated Presentation Agents to the Interface.” In Intelligent User Interfaces, 1997. 79-86.

[17] Ribeiro, R., Baptista, F., Pardal, J., Mamede, N., & Pinto, H. (2006). "Cooking an Ontology". AIMSA, (pp. 213-221).

[18] Melo, C. "Gesticulation Expression in Virtual Humans", Master Thesis, Instituto Superior Técnico, Lisbon, 2006.

[19] Dias, J., Ho, S., Azenha, B., Prada, R., Enz, S., Zoll, C., Kriegel, M., Paiva, A., Dautenhahn, K., Ayllet R., Gomes, M. R., Ferreira, C.. "D4.1.1-D6.2.1 Actor with Attitude and Component Specification for Computational Autobiographic Memory", (2007), ST-4-027656-STP eCircus

Appendix A

A. Example of an XML recipe

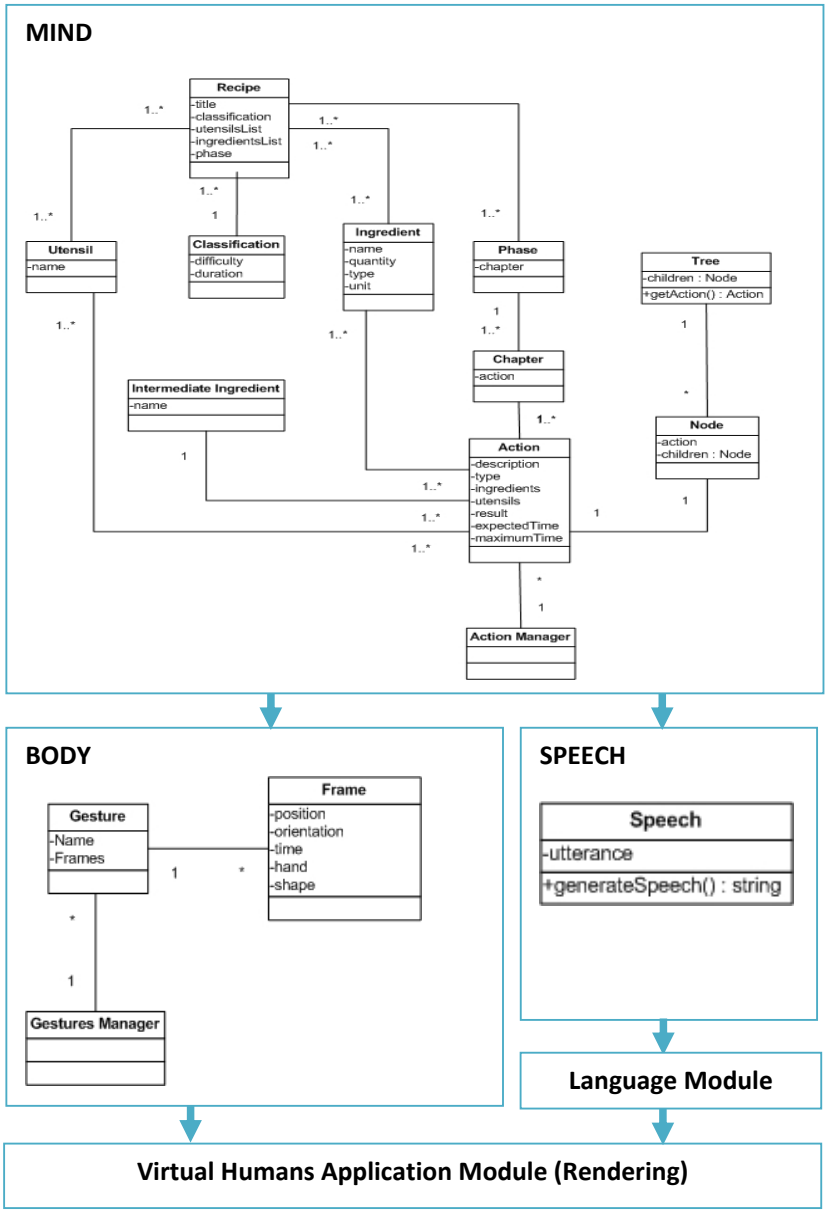
```
<recipe xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:recipe recipe.xsd" xmlns="urn:recipe">
  <title>Chocolate Mousse</title>
  <classification>
    <duration>30 min</duration>
    <difficulty>easy</difficulty>
  </classification>
  <ingredientsList>
    <ingredient>
      <name>chocolate</name>
      <quantity>250</quantity>
      <unit>gram</unit>
      <type>sweets</type>
    </ingredient>
    <ingredient>
      <name>butter</name>
      <quantity>125</quantity>
      <unit>gram</unit>
      <type>fats</type>
    </ingredient>
(…)
</ingredientsList>
<utensilsList>
    <utensil>
      <name>spoon</name>
    </utensil>
    <utensil>
      <name>bowl</name>
    </utensil>
(…)
</utensilsList>
<confection>
    <chapter>
      <action>
        <description>Put the chocolate and butter in a pan</description>
        <type>join</type>
        <what>chocolate</what>
        <what>butter</what>
        <where>pan</where>
```

```
        <how></how>
        <result>chocolate and butter</result>
        <expectedTime>
          <time>00:00:30</time>
        </expectedTime>
        <maxTime>
          <time>00:01:20</time>
        </maxTime>
      </action>
      <action>
          <description>You need to let the chocolate and butter melt in a pan
       </description>
        <type>melt</type>
        <what>chocolate and butter</what>
        <where>stove</where>
        <how></how>
        <result>chocolate and butter melted</result>
        <expectedTime>
          <time>00:00:20</time>
        </expectedTime>
        <maxTime>
          <time>00:01:30</time>
        </maxTime>
      </action>
(…)
</chapter>
  </confection>
</recipe>
```

Appendix B

B. Class Diagram

The following example illustrates how the main classes of the system are related between then and with the external modules.



Appendix C

C. – Example of a Gestures XML File

```
<gestures xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:gestures gestures.xsd" xmlns="urn:gestures">
<gesture>
    <name>greeting-right</name>
       <frame>
          <position>-32,6;90;-5</position>
          <orientation>-90;25;90</orientation>
          <hand>0</hand>
          <time>1</time>
          <shape></shape>
       </frame>
   </gesture>
   <gesture>
    <name>greeting-left</name>
       <frame>
          <position>-32,6;90;-5</position>
          <orientation>-90;-25;90</orientation>
          <hand>0</hand>
          <time>1</time>
          <shape></shape>
       </frame>
   </gesture>
(…)
</gestures>
```

Appendix D

D. User Questionnaire

At the end of the experiment, the following questionnaire was delivered to all participants. The questions were made in Portuguese. However, it was made a translation of all questions to English and is presented along with the Portuguese questionnaire.

Idade:
Age:

Habilitações Académicas:
Education:

Sexo:
Sex:

Este questionário destina-se à apreciação da aplicação do ponto de vista do utilizador. Por favor responda às questões utilizando a escala lateral onde:

This questionnaire tries to evaluate the user's interaction with this application. Please answer the questions using the following scale where:

1 – Discordo totalmente
Totally disagree

3 – Não concordo nem discordo
Neither agree nor disagree

5 – Concordo totalmente
Totally agree

Pergunta *Question*	**1**	**2**	**3**	**4**	**5**
1. Achei a interface simples e acessível *I found the interface simple and accessible to use*					
2. Achei o sistema útil *I found the system useful*					
3. Achei o sistema aborrecido *I found the system boring*					
4. Gostei de interagir com o sistema *I liked to interact with the system*					
5. Achei adequado o método de interacção com o sistema *The interaction with the system was adequate*					
6. Consegui acompanhar o agente na sua explicação *I was able to keep up the agent's explanations*					
7. Consegui compreender a receita *I understood the recipe*					
8. Os gestos do agente eram de fácil compreensão *The gestures of the agent were easily understood*					
9. Os gestos do agente eram adequados às acções *The agent's gestures were in conformity with the action*					
10. As explicações do agente eram de fácil compreensão *The agent's explanations were easily understood*					
11. O agente ajudou-me quando estava em dificuldades *The agent helped me when I needed*					
12. O agente deveria auxiliar mais vezes *The agent should assist more often*					
13. Achei o agente muito intrusivo *I found the agent very intruding*					
14. Achei as ajudas adequadas *I found the tips given were adequate*					
15. Consegui aprender a receita apresentada *I was able to learn the recipe given*					

<u>Comentários/Observações</u>

Comments

_____

_____

_____

_____

Obrigado pela sua participação!

Thank you!