MODELING STUDENT BEHAVIORS IN A VIRTUAL CLASSROOM USING BELIEF DESIRE INTENTION MODEL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRE CANBAZOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF MODELING AND SIMULATION

FEBRUARY 2014

Approval of the thesis:

MODELING STUDENT BEHAVIORS IN A VIRTUAL CLASSROOM USING BELIEF DESIRE INTENTION MODEL

submitted by **EMRE CANBAZOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Game Technologies, Middle East Technical University** by,

Prof.Dr. Nazife Baykal
Director, Informatics Institute _____

Assist.Prof.Dr. Hüseyin Hacıhabiboglu
Head of Department, MODSIM, METU, **Game Technologies** _____

Prof. Dr. Veysi İşler
Supervisor, **Department of Computer Engineering** _____

Examining Committee Members:

Prof. Dr. Faruk Polat
Computer Engineering, METU _____

Prof. Dr. Veysi İşler
Computer Engineering, METU _____

Prof. Dr. Kürşat Çağıltay
Computer Education and Instructional Technology , METU _____

Assoc. Prof. Dr. Uğur Güdükbay
Computer Engineering, Bilkent University _____

Dr. Erdal Yılmaz
Argedor _____

Date: 06.02.2014

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: EMRE CANBAZOĞLU

Signature :

ABSTRACT

MODELING STUDENT BEHAVIORS IN A VIRTUAL CLASSROOM USING BELIEF DESIRE INTENTION MODEL

Canbazoğlu, Emre

M.Sc., Department of Modeling and Simulation

Supervisor : Prof. Dr. Veysi İşler

February 2014, 66 pages

Agent and behavior modeling is one of the most important components of computer games and virtual environments that make these products more realistic and attractive. Agent and behavior modeling can also be used for serious games which is designed for training people in virtual interactive environments instead of real life training with less cost and close effectiveness. Belief-Desire-Intention(BDI) model is one of the software models that is used to model intelligent agents. In this thesis, we used BDI architecture to build an interactive virtual training environment, that is full of intelligent agents, for people who want to train themselves before facing real-world situations. With this approach, we show that it is possible to train and evaluate people with fully developer generated virtual systems adapted to specific scenerios without needing any living individual. We also integrated the system with Unity3D, one of the most popular game engine in the world and modeled a virtual training classroom with various intelligent student agents. These agents differentiate from each other with characteristics, behaviors and the responses to the user actions. Thus, the environment can become more realistic

and unexpectable so that trainees can learn how to deal with variety kind of situations.

Keywords: BDI, Virtual Environment, Training, Artificial Intelligence, Virtual Classroom

ÖZ

SANAL BİR SINIFTA ÖĞRENCİ DAVRANIŞLARININ BELIEF DESIRE INTENTION MODELİ İLE MODELLENMESİ

Canbazoğlu, Emre
Yüksek Lisans, Modelleme ve Simulasyon
Tez Yöneticisi : Prof. Dr. Veysi İşler

Şubat 2014, 66 sayfa

Üstlenici ve davranış modellemesi, bilgisayar oyunlarının ve sanal ortamların gerçekçi ve çekici olmasını sağlayan en önemli bileşenlerden biridir. Üstlenici ve davranış modellemesi ayrıca, gerçek dünya eğitimlerine yakın bir verimde ve ondan daha az maliyetli olacak şekilde, insanları sanal etkileşimli ortamlarda eğitmek üzere dizayn edilmiş ciddi oyunlarda da kullanılabilir. Belief-Desire-Intention(BDI) modeli, akıllı üstleniciler yaratmak için kullanılan bir yazılım modelidir. Bu tezde, gerçek dünya olaylarıyla karşılaşmadan önce kendilerini eğitmek isteyen insanların kullanabileceği, akıllı üstleniciler ile dolu etkileşimli bir sanal çevreyi BDI modelini kullanarak oluşturduk. Bu yaklaşımla birlikte, insanları tamamen geliştiriciler tarafından oluşturulmuş ve belirli senaryolara adapte edilmiş sanal sistemlerde hiç bir gerçek bireye ihtiyaç duymadan eğitmenin ve değerlendirmenin mümkün olabileceğini göstermeye çalıştık. Bununla beraber, sistemi çeşitli akıllı üstlenicilerle dolu bir eğitsel sınıf modellemek üzere günümüzün en popüler oyun motorlarından biri olan Unity 3D ile entegrasyonunu gerçekleştirdik. Bu üstleniciler birbirlerinden gözlemleri, arzuları ve davranışları bağlamında birbirlerinden ayrılmaktadırlar. Bu ayrılma ile beraber sanal çevre daha gerçekçi ve beklenmeyen

hale gelmiş olup, eğitilen insanların farklı durumlarla nasıl başa çıkabileceği konusunda eğitilmelerine yardımcı olacaktır.

Anahtar Kelimeler: BDI, Sanal Ortam, Eğitim, Yapay Zeka, Sanal Sınıf

To people who believes that computer is a miracle..

ACKNOWLEDGMENTS

I would to express my deepest appreciation to my advisor Prof.Dr. Veysi İşler for his recommendation about the subject of the research and guidance, advices and encouragement throughout the research.

Also I am very grateful to TÜBİTAK for supporting me financially through my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Agent and behavior modeling is one of the key aspects of realism in computer games and simulations. Without well designed agents, the immersion of the users is lost in many ways. Users expect the agents to think and act similarly as in real-life. Modeling a realistic agent is a complex progress. All the possibilities should have been taken into consideration by the developers. This can be really difficult in complicated games and simulations. One big problem is, such a high level realistic modeling architecture has also a high computational cost. In recent years, as the IPS of the computers increases, agent and behavior modeling started to took more important place in solving real-time problems without concerning the performance cost deeply. More complicated algorithms introduced in computer games and intelligent agents gained more importance in real-time simulations. One other key area that intelligent agents take place is virtual training simulations. With virtual training, users can be trained in many various conditions that are hard to generate in real-life without needing living beings. This reduces the cost financially, covers wide range of scenerios and increases the variaty of agent types, brings opportunity to train without being affected by the results and also opportunity to practise many times as wanted.

Virtual training simulations such as military simulations, emergency planning simulations and serious games that are used for training users, need well designed modeling architecture for getting more effective results. These kind of applications must involve intelligent agents so that the user can interact with them and have the capability to effect the environment and the agents behaviors. In this thesis, virtual students with different characteristics will be modeled in a virtual classroom that will be used for training teacher candidates before they start to teach in real world. Virtual classroom differentiate from other applications like military training or emergency planning with an important difference. In military like training applications, users generally order other agents to do specific actions. On the other hand, there should be a two way of interaction in a classroom. The agents act, user reacts; user acts and the agents react. Considering this kind of interaction, an agent should observe the current situation of the environment and act according to these observations. For this purpose, BDI is selected as the architecture of the intelligent agents. In BDI, an agent has perceptions about the environment,

goals that he wants to achieve and plans to make these goals successful. BDI architecture is also suitable for the key component of interaction, communication. Agents can request and send information about the environment so that their perceptions are not limited with what they see and think.

There are some implementations of virtual training classrooms with variaty of approaches. Some of them uses real people as students. They control an avatar in the virtual world and behave as a student in the classroom. With the proposed method, the aim of the thesis is to bring virtual training classrooms one step further.

1.1 Scope

The main focus of this thesis is implementing BDI architecture to model intelligent agents , more specifically virtual students with different characteristics in a virtual classroom, where teacher candidates can train themselves by dealing student issues without being responsible for the results. Our main contribution is using BDI agents to model various virtual students in a real-time, interactive, scenario based training environment so that realism of the environment will be kept at high level. We implemented this approach in a 3D modeled virtual classroom which is developed with a well-known game engine, Unity3D, and build the agent modeling architecture as a service using JASON which can be used in a server-client architecture in the future. We also tried to make it possible to generate BDI agents in the Unity3D environment which makes generating the scenarios easier.

1.2 Outline

The thesis has been organized into the following chapters:

- Chapter 2 provides an overview of the concepts ”intelligent agents”, ”multi-agent systems”, ”agent-oriented programming”, discusses the related works and state of the art about the virtual training environments, explaines the BDI architecture which is used during the thesis, gives examples of applications that uses this architecture, gives brief information about BDI interpreters and Unity3D.
- Chapter 3 describes the proposed virtual student modeling method, conceptual design of the virtual classroom, the architecture of the system; explains how we map student behaviors to BDI format in details.
- Chapter 4 covers the case studies that demonstrates the accuracy of the system and the differences between various user behaviors in virtual classroom. This chapter also discusses the results we got from these studies.
- Chapter 5 concludes the study with a summary and potential future works

CHAPTER 2

BACKGROUND AND RELATED WORKS

The main focus of this thesis is creating an interactive virtual training classroom which is full of various characterized intelligent agents. In this chapter, we first define what an intelligent agent is. We also define agent characteristics and their properties. We explain multi-agent systems, give examples of multi-agent applications, introduce agent-oriented programming and compare it with object-oriented programming to give an overview about the rest of the thesis. Then we extend the definition of the concepts; ”virtual environment” and more specifically, ”virtual training environment” and give information about recent researches and implementations about these concepts. We discuss the capabilities, limitations of these implementations and go through their underlying main architecture. Next, we dive into BDI model which we used in the scope of this thesis and mention the theory behind BDI model. We explain the architecture of the model and introduce the key aspects of this architecture. Also we express the main steps of the BDI cycle and give some details about them. Next, we go through the related works and researches about BDI models. After that, we introduce the mostly-used BDI interpreters and compare them with giving more details. We represent JASON, an implementation of AgentSpeak which is used for the thesis, and explain the reasons why it is chosen.

2.1 INTELLIGENT AGENT

2.1.1 Definition

Before going deeper in multi-agent systems and virtual environments with agents, we must understand the meaning and the content of the word ”agent”. First, we must clarify the definition of ”agent” vary as the set of examples of agents differ in the definers mind. According to Jennings and Wooldridge [3], agents are computer systems which are designed to achieve their objectives that are appointed by the system developers with autonomous actions in an environment they situated in. By autonomousity, we mean that the agents can act according to the environment conditions without the control of any human. Jennings and Wooldridge [3] express that the difference between an object in a traditional system and an agent is, an abject can invoke a method in another object so that the object that is invoked is not autonomous and

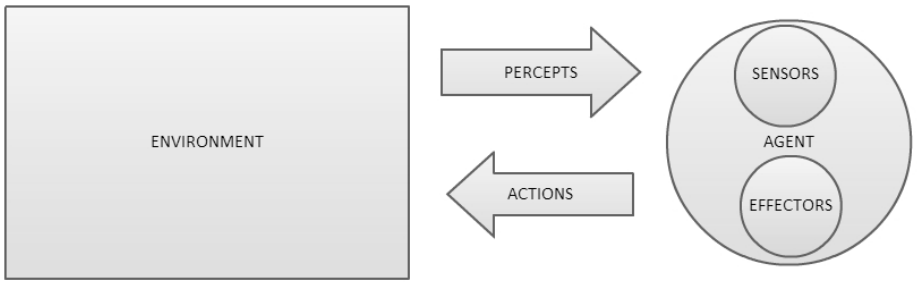


Figure 2.1: Agent and Environment

has no control over its own actions. Bordini et. al. [2] think agents that agents are active action producers. Agents in the environment, have goals to achieve for us figuring out how to accomplish best rather than having to be told what to do. Bordini et. al. gives ”booking a holiday for us”, ”bidding on our behalf in an online auction” or ”cleaning our office space for us” as examples of the agent tasks. According to Hayes and Roth [4], there are three main functions that the agents continuously perform. First one is percepting the dynamic conditions in the environment. Agents continuously check whether the conditions of the environment changed or not to decide what to do and how to do. Second one is, acting to affect and changing the conditions in the environment. In every action of an agent, the conditions in the environment change somehow. The final function that agents continuously perform is reasoning to interpret perceptions, solve problems and determine actions.

2.1.2 Properties and Characteristics of Agents

Lin and Michael [5] state that two basic property of the agents are being autonomous and being situated in an environment. Being autonomous means that each agent make their own decisions and that is what differs agents from objects. On the other hand, situatedness means that agents sense their environments and have a set of actions to perform that can change the environment. Figure 2.1 shows the situatedness of an agent.

Other properties that agents should have are proactiveness, reactivity and social ability [3, 2]. They should exhibit a goal-directed behaviour which means, if an agent has appointed to a goal, then it should try to achieve this goal. This property shows agents' difference from regular objects. An agent does not require to be invoked to perform some action in the environment. This property is called proactiveness. The other property that an agent should have is reactiveness or responsiveness. This means, an agent should perceive the environment they situated in and respond to the changes occurred. Balancing the reactiveness and proactiveness is one of the main challenges in agent-systems [2, 5]. An agent's plans and action must be influenced by both environmental changes and agent's own goals. If an agent is too reactive it will not be able to perform a decided plan because of adjusting it all the time. On the other hand, if the

agent is too proactive, it will stick to his plan redundantly even if that plan is not able to be achieved because of the changes in the environment. One other important property of agents is having social ability. The agents should interact and communicate with other agents and humans to cooperate and coordinate their activities. They should share their knowledge about the environment with each other. Additionally, according to Ralf [6] subjective rationality, robustness, coherence, personalizability are the other characteristics that an agent should have. Some of these are less central characteristics that not all agent applications use these characteristics. Subjective rationality means, an agent must be equipped with intelligence that makes the agent capable of solving problems and making decisions. Also an agent should learn from its failures and adjust their plans. This is called robustness.

We should mention about agent environment since all software can be considered to be situated in an environment [5]. Lin and Michael state that the environments where the agents are situated in are dynamic because they can change rapidly; unpredictable because it is impossible to predict the future and unreliable because not all the time agents can achieve their goals. According to Russell and Norvig [7] there are four classification of environment properties. These are;

- Accessible versus inaccessible : States if all the information in the environment is accessible or not.
- Deterministic versus non-deterministic : States if an action has only one effect or not.
- Static versus dynamic : States if the conditions of the environment changes or not.
- Discrete versus continuous : States if the environment has finite number of actions and percepts.

2.2 MULTI-AGENT SYSTEMS

We talked about agents and their characteristics. Now, we will discuss the agent systems that are used in applications. In practice, single-agent systems are rare [2]. In many applications, there are more than one agent in the environment and it is called ”multi-agent system”. Multi-agent systems are composed of multiple interacting agents [8]. Agents interact with each other by sending and receiving messages. Generally, these agents have different goals and motivations. According to Syscara [9] there are some motivational subjects that leads people to make researches about multi-agent systems. First, some problems can be too complex for a single agent to solve. Second, supporting interconnection of legacy systems. Third, solving problems that can decompose into interacting agents. Fourth, using distributed spatial information efficiently. Fifth, solving problems that expertise is not centeralized. Sixth and last, making the system more reliable, flexible, extensible, maintable and responsive.

Wooldridge [10] states that the agents require the ability to cooperate, coordinate and negotiate with other agents in a similar way as we do with other people in real life. When we look at the

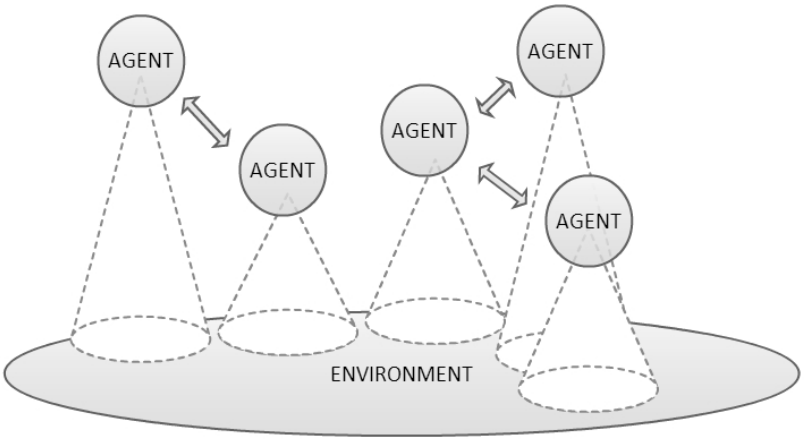


Figure 2.2: Structure of MAS

overview of the multi-agent systems, we see that all agents has an influence area that they have fully or partially control over it. These areas can overlap so more than one agent have control over the same area. These overlaps make the system more complicated because agents have to take into account other agents actions and desires. According to Syscara [9], there are four main characteristics of multi-agent systems. These are;

- Each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint
- There is no system global control
- Data are decentralized
- Computation is asynchronous

2.3 MULTI-AGENT APPLICATIONS

There are several areas of agent applications. These areas vary from industrial applications to commercial applications, medical applications to entertainment applications. If we extend these topics, we can see process control [11], air traffic control, information management, mail filtering, internet news filtering [12], electronic market-place [13], business process management [14], patient monitoring [15], many computer games, interactive theatre and cinema and many other applications. These are the first agent applications in their area and they have been evolving day by day. Considering the scope of this thesis, we will dwell on the area of computer games and simulations.

(a) Snapshot from Strategy Game, AOE III

(b) Snapshot from Simulation Game, The Sims

(c) Snapshot from FPS Game, Rainbow Six

Figure 2.3: Examples of MAS Applications

2.4 AGENT-ORIENTED PROGRAMMING

We can think agent-oriented programming as an extension of object-oriented programming [10]. Agent-oriented programming requires different programming properties respect to object-oriented programming. Bordini et al. [2] describes these requirements as follows:

- We should able to delegate goals to the agents.
- We should be able to describe our goals without dealing with low-level programming and not concerning about the way they will achieve these goals.
- The language should provide support for goal-directed problem solving.
- We should be able to produce systems that are responsive to dynamic environments.
- We should be able to use goal-directed and responsive behaviours together clearly.
- The language should support knowledge sharing and cooperation between the agents.

There are differences between the agents and the objects. Unlike objects, agents are autonomous, heterogeneous and social. Objects are passive and have no control over method invocation, designed for a common goal and typically integrated into a single thread. In contrast, in agent-oriented programming, agents are autonomous, they can have diverging goals and have own thread of control. In procedural programming, developers should think the problem in detail so it is difficult to solve a complex problem with procedural programming. On the other hand, in agent-oriented programming, we only state a goal and let the system handle how to achieve this goal. Thus, it becomes easier to decompose a complex problem with agent-oriented programming [1].

Multi-agent systems can be categorized according to their architecture. There are four main architectures [16]. These are:

- Logic-based architectures: In this architecture, environment is symbolically represented. As the human knowledge is also symbolic, we can understand the logic easier. On the other hand, it is harder to represent the world symbolically.
- Reactive architectures: This architecture implements decision-making as a direct mapping of state to action and based on observe-react mechanism. In this architecture there is a real-time information flow between the finite state machines and sensors of the agents.
- BDI architectures : There are four key structures that are used to represent data. These are beliefs, desires, intentions and plans. BDI is the most popular agent architecture and has a wide range of applications. The model's validity is testified with these applications. We will discuss BDI model more deeply in the next sections as this architecture is used for the implementation of virtual training classroom.

Table 2.1: OOP vs AOP [1]

	OOP	AOP
Primitive Unit	Object	Agent
How to Define Unit	unconstrained	beliefs, capabilities, commitments, choices...
Computation	message passing and response methods	message passing and response method
Message Types	unconstrained	inform, request, offer, promise, decline...
Constraints on methods	none	honesty, consistency...

- Layered (hybrid) architectures: This kind of architecture allows both reactive and deliberative agent behaviour. To achieve this, subsystems are arranged.

2.5 BDI AGENTS

2.5.1 BDI Theory

BDI (Belief-Desire-Intention) model is based on human behaviours in terms of folk-psychology. This model was developed by philosophers. BDI model was first introduced by Bratman [17, 18] in the mid-1980s. The main idea behind the model is about practical reasoning which focuses on intentions, special desires that agents are committed to. The most important component that Bratman introduced in the theory was intention. The importance of other components of the BDI model (belief and desire) was stated in the previous researches such as "intentional stance" [19]. According to intentional stance, the agents decide their actions depending on their beliefs and desires. The most important statement Bratman explained was the decomposable structure of the intentions into beliefs and desires. The biggest difference between an intention and a desire is, agents do not give their intentions up easily when they are committed to it like a mere desire. Bratman defines intentions' properties as; conduct-controlling pro-attitudes, having inertia and being input for the further actions. A pro-attitude is an agent's mental attitude directed toward an action under a certain description. We expect the agent to make some attempt to achieve the intention. Intentions are conduct-controlling pro-attitudes as they bring a commitment to the actions. Inertia property represents the stability of the intentions. Once an

agent is committed to an action, normally it stays still until the agent does the action. Intentions can be input for future so that they influence further decision makings. Because of these properties of intentions, Bratman describes the characteristics of the intentions as:

- An intention is a high-level plan.
- An intention shapes further planning
- An intention has the agent's commitment
- An agent abandons an intention if:
    1. he achieved the intentions;
    2. he believes current intention is impossible to achieve;
    3. another intention is more sensible to commit

As a result, Bratman [17, 18] distinguished intentions from beliefs and desires and came up with the belief-desire-intention model instead of belief-desire model.

2.5.2 BDI Architecture

We referred to the significance of intentions in the previous section. In this section we will explain the architecture and the key components of BDI model in detail. First, we will start with beliefs, desires, intentions and some other definitions. We continue with practical reasoning, cycle of the architecture, procedural reasoning system and finish the section with the communication between agents in the BDI model.

Beliefs are the information that the agent has about the world [2]. These beliefs can change in time. Therefore all agents have to update their beliefs in periods. Other key component of the model , desire, represents all the states of actions that the agent would like to achieve. They can be considered as the options for the agent. That means, they are potential influencer for the agent [2]. It might be useful to walk through the definition of intention briefly. Intentions can be counted as "selected" desires for the agents. Agents are committed to intentions and try to achieve it until the intention is done or impossible to be done. There are some other sub-definitions as goals and plans. Goal set is the subset of the desire set. They are realistic and consistent. This subset is formed depending on the beliefs. And plans are the intentions which are broken into list of actions.

The agent which has beliefs, desires and intentions must do actions. Thus, the agent should decide what to do. The decision making model that BDI depends on is practical reasoning. We can shortly define practical reasoning as "the process of figuring out what to do". Bratman defines practical reasoning as follows:

”Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.” [20]

Human practical reasoning consists two activities; deliberation which means deciding what to achieve and means-ends reasoning which means how to achieve it. Agents generally do not decide on intentions that they believe conflict with their current intentions [21]. Also they believe their intentions are possible. The outputs of deliberation activity are intentions whereas the outputs of means-ends reasoning activity are plans. It might be useful to dwell upon means-ends reasoning a bit because of the scope of this thesis. Means-ends reasoning is deciding how to achieve the intention that the agent is committed to. It is also known as planning. This system takes inputs into account and generates a plan. A plan is a sequence of actions, the smallest structure that an agent can do. The agents' current intention, beliefs about the environment and the action set of the agent are the inputs of the system. If the agent follows the plan that the system generates and the environment conditions remain same, he can achieve his intention. In the scope of the thesis, we will focus on decision making depending on offline-designed partial plans. The partial plans of an agent are developed by the programmer during the design progress and used when corresponding goal is selected as an intention. In this pre-written approach, the plans are like recipes that carries the agent to success. We will explain the plans in details in the following sections.

At first look, the cycle of the BDI model looks simple. We can list the main steps of the agents in the cycle as follows:

- Observes the environment and updates its beliefs.
- Deliberates its desires to decide which one to commit and take it as an intention.
- Generates a plan by using means-ends reasoning to achieve the intention.
- Executes the plan.

The cycle looks simple at first however there are many problems with this cycle like commitment balancing. Cohen and Levasque [21] defines commitment balancing as rational balance. Rational balance means, the agent should not be too strongly or too weak committed to their intentions. Bordini et al [2] came up with a control loop that covers rational balance. The agent remains committed to the intention until it achieves it, believes it is impossible to achieve or it another intention is more sensible to commit. Figure 2.4 shows the pseudo-code for an agent's BDI cycle.

This pseudo-code covers the main steps we listed above and also come up with solution to rational balance.

However, this pseudo-code is still way different from the implementation [2]. One of the most stable approach to implementation of BDI architecture is the procedural reasoning system

```
1.  B ← B0;        /* B0 are initial beliefs */
2.  I ← I0;        /* I0 are initial intentions */
3.  while true do
4.    get next percept ρ via sensors;
5.    B ← brf(B, ρ);
6.    D ← options(B, I);
7.    I ← filter(B, D, I);
8.    π ← plan(B, I, Ac); /* Ac is the set of actions */
9.    while not (empty(π) or succeeded(I, B) or impossible(I, B)) do
10.     α ← first element of π;
11.     execute(α);
12.     π ← tail of π;
13.     observe environment to get next percept ρ;
14.     B ← brf(B, ρ);
15.     if reconsider(I, B) then
16.       D ← options(B, I);
17.       I ← filter(B, D, I);
18.     end-if
19.     if not sound(π, I, B) then
20.       π ← plan(B, I, Ac)
21.     end-if
22.   end-while
23. end-while
```

Figure 2.4: BDI Cycle [2]

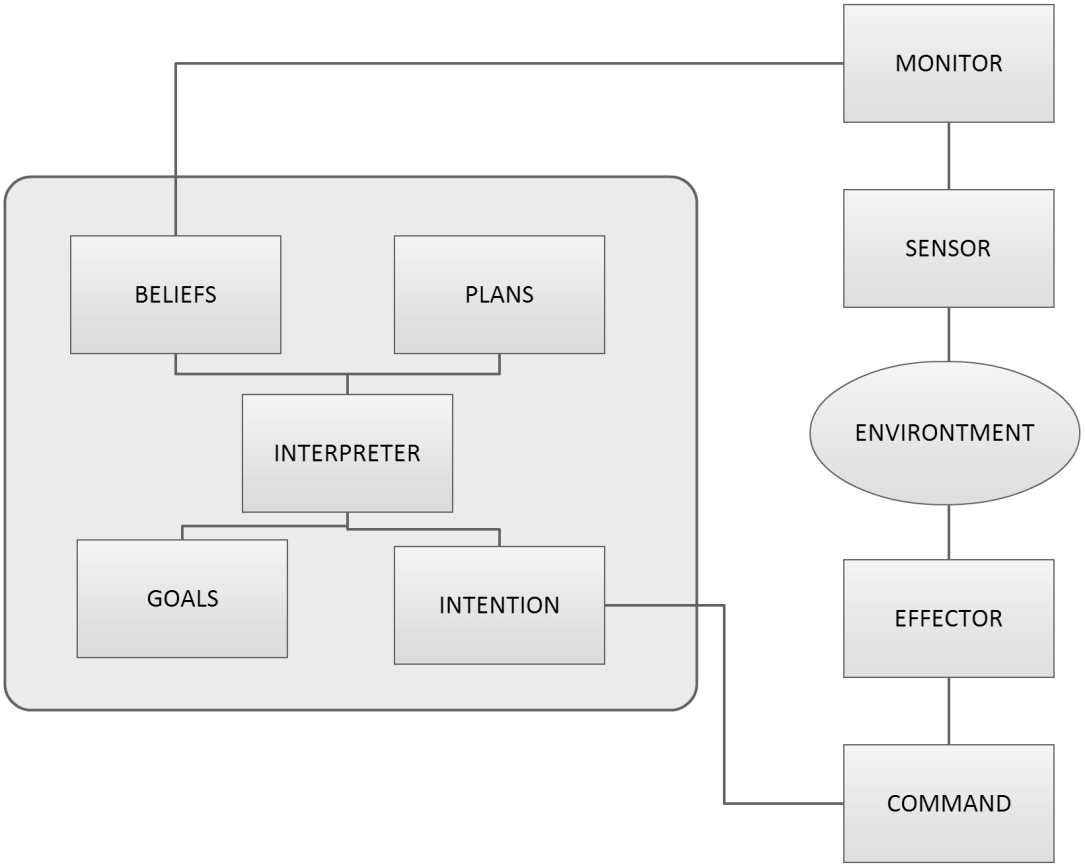


Figure 2.5: PRS System Structure

(PRS). The procedural reasoning system developed at Stanford Research Institute by Michael Georgeff and been used for BDI architectures since. PRS does not use primitive actions to generate plans. PRS is a plan execution system [22]. It assumes there are already plans ready to be executed. As we mentioned before these plans are programmed by the developer and used as ”receipts” in the PRS. One other important difference of PRS from other planners is being more dynamic and is being able to adapt easily to changes in the environment. In traditional planners, the entire course of action is planned before execution of the plan and does not take the changes in the environment into account. On the other hand PRS continuously checks the changes in the environment and reconsider its choices.

As seen in Figure 2.5, PRS has a database of beliefs, set of goals, set of plans and an intention structure. The interpreter uses all of the components and decides which plan to execute according to the beliefs and and goal set. The agent observes the environment with its sensors and effects it with its effectors.

Communication is one of the essential components of multi-agent systems. Communication in BDI model is typically based on speech-act theory [23]. There are various types of speech act [24]. These are listed below:

- Representatives
- Directives
- Commissives
- Expressives
- Declarations

Agents in the environment communicate with each other by using one of these speech acts.

2.5.3 Related Researches

In this section we will go through the researches about BDI modeling in virtual training applications.

Cap et al. [25] states that Finite State Machines (FSM) are generally used for developing agents. However, it can be impossible to create a set that spans all possible states for complex problems. Instead they chose BDI modeling to develop agents which is a more promising approach. With this research [25] Cap et al. present the design, development and implementation of training simulator which is based on complex real-world task. Within the scope of the paper, they developed an agent based training simulation for the Netherlands Defence Organization. The domain of the simulation is fire fighting where the users are trained as commanding officer, Officer of the Watch (OW). Users communicate with other officers, command them, develop and adjust plans, monitor the events, generally organize the team to deal with the situation. Cap et al.[25] refer to the importance of the realism of the simulation. So they also represent other agents in the simulation as virtual characters. Moreover, they think all the information and communication equipment should be simulated in order to make trainee practice the task as similar as in real-life. The environment should be modeled realistic too. In this application it is called machinery control room (MCR). As it is a training simulation, it is also a must to evaluate the performance of the trainee and the proper feedbacks should be given. Some of the functional requirements that training applications should have according to Cap et al. [25] are listed as below:

- Virtual characters represent the trainee and other team members.
- Users can interact with other virtual characters in some degrees of freedom.

- Virtual characters should be able to autonomously perform their tasks
- They need to have knowledge and reason about what has happened in the past.
- What a team member can perceive should be determined by its location.
- System should be able to compare trainee actions with expert actions.
- Only relevant information is transferred in order to keep the system fast.
- Systems should be easily extendible with new scenarios.

See [22] for technical solutions in the scope of that specific application to address these function requirements. These functional requirements were considered during the development of the virtual training classroom which will be explained in the following chapter.

The system of the fire fighting simulation architecture consists of two major parts: First one is a 3D visualization system and second one is the intelligent agent system that build on BDI model. We will not dwell on visualization part which is not in the scope of this thesis completely.

The agents in this research are called role-playing agents. Role playing agents can act because of their own task knowledge (using its own beliefs and commit to a desire) or because of requests, orders given by the trainee or other role-playing agents. The priority rules determine what will the agent do.

Other than role-playing agents, there are functional agents like connection manager which connects the 3D world with agents; world manager which simulates the non-visualized part and blends it with the visualized part; OW which is the user model of trainee; broker which is responsible of determining which messages will be transmitted to each of the agents and scenario manager which controls the course of events based on a pre-defined scenario. The agent-system is implemented using Jadex platform which will be explained in the following sections.

Norling and Sonenberg [26] also argue that BDI paradigm is well-suited to developing virtual characters that use similar underlying reasoning system and have similar level of intelligence to people. They developed Quake 2 bots with this model that could play against other players to demonstrate their case. They state that with BDI model, they do not have to fully specify every possible course of action to generate complex behaviours. They used JACK programming language [27] for implementation.

According to Norling and Sonenberg [26], gathering the knowledge that the characters will need to operate in their world is big challenge. And the difficulty of the challenge depends on two main factors: the complexity of the environment and the complexity of the interactions between agents.

The method they used to gather the knowledge data is making interviews to the expert players of Quake 2 and that data was used to build models of these players. In the scope of paper [26],

they focused on two players with different styles of game planing. They did three different interviews; first one to get a general idea and the second one to get detailed data and the last one to fill the gaps in the knowledge. With these interviews they tried to understand how the players perceive the environment, and the goal they focus on.

One of the most important results of the interviews was the way that players perceives the world. So they came up with an argument of "beliefs of the character are largely determined by the role". They also differ in strategies they perform. For example; as one player takes more risks while seeking other players, other choose to stay safe. The aim of the paper is to implement these differences by using the advantage of the abstraction provided by the BDI model.

2.5.4 BDI Interpreters

In this section, we will give brief information about some of the BDI frameworks and languages.

2.5.4.1 JACK

According to Winikoff [28] an agent platform needs to contain at least the following components:

- Agents should be written by using agent components like plans, beliefs, goals rather than non-agent oriented languages
- Library providing communication between agents.

JACK is an agent platform that includes these components and more.

2.5.4.2 JADE

JADE [29] is a software environment to build multi-agent systems through the predefined programmable and extensible agent model. It is one of the most used agent development framework which has more than two thousands active members. It is fully developed in Java and supports Java libraries. Main principles of JADE are listed below:

- Interoperability by using FIPA communication standard.
- Uniformity and portability. It provides same API for J2EE, J2SE and J2ME.

- Easy to use
- Pay as you go philosophy.

2.5.4.3 JADEX

JADEX [30] provides development of multi-agent systems by using object-oriented concepts and technologies such as JAVA and XML. Also, JADEX tries to overcome the limitations of BDI systems by explicit representations of goals.

2.5.4.4 JAM

JAM [31] is an intelligent agent architecture that combines BDI, PRS and Structures Circuit Semantics (SCS) [32]. It was not a complete architecture by the time paper published but provides rich, expressive procedural representations, useful goal semantics, meta-level reasoning and more while sticking to the BDI theoretic.

2.5.4.5 JASON

Jason [2, 33] is an extension of AgentSpeak(L), one of the most sharp programming languages in the literature which is introduced by A. Rao [34] and the purpose of the language is implementing reactive planning systems. It is built on BDI agent architecture. It also provides speech-act based communication which we mentioned in previous sections. Some of the features of Jason are [33]:

- Speech-act based communication
- Annotations are provided
- The possibility to run a multi-agent system over a network.
- Fully customizable in Java
- Extensibility by user-defined internal actions
- Multi-agent environments can be implemented in Java

These features are the main reasons why we choose Jason as the BDI interpreter and framework within the scope of this thesis. Another advantage of Jason is, it can be integrated to Eclipse which supports Java IDE for Jason language and that makes easier to program for a developer.

It also has a debugger, which is an essential component of developing and generating large and complex systems.

One of the most important thing to know about AgentSpeak(L) is, the agents are represented by a set of beliefs that gives the initial state of the agent's belief base and with a set of plans called plan library.

The main components of Jason are: Beliefs, Goals and Plans. We explained beliefs in details in previous sections. However, Jason differs from others with some representations like annotations which give specific details about the beliefs and rules which represents complex beliefs. Goals are fundamental in agent programming. In AgentSpeak(L) there are two types of goals: achievement goals which represent a state that an agent want to achieve and test goals which represent whether a goal is associated with the agent's belief base. Plans are the agents' "know-how" [2]. The changes in the agents' beliefs and goals, trigger the execution of the plans. A plan in AgentSpeak(L) has three components: triggering event, context and the body. Triggering event represents what will be done when the plan is achieved. There are six types of triggering events that plans can have. These are: belief addition, belief deletion, achievement-goal addition, achievement-goal deletion, test-goal addition and test-goal deletion. Context part of the plan defines when a plan should be considerable. It can be count as the conditions of the plan. And lastly body of the plan represents the plan steps which should be done to achieve the plan. It is the recipe of the plan.

These are the main concepts about AgentSpeak and the extension of it, Jason, which will be used during the implementation of the virtual training classroom.

2.6 VIRTUAL ENVIRONMENTS

As the scope of this thesis covers a virtual environment with intelligent agents, we should define the concept of virtual environment and virtual environment-based training first. In this section we give detailed information about these concepts and show the researches done among this field.

Virtual environments are computer-generated environments which are used to simulate the real world [35]. These environments can be either simple as a semi-immersive environments or completely immersive environments. By completely immersive environments we mean, hardware-based, three-dimensional interactive experiences with sound and force feedbacks, accurate as possible [36]. There are two categories of virtual environments when we classify them according to their rendering. One is image-based rendering method which is less interactive and the other one is model-based method which is more interactive. Image based method is generally used in game applications, while model based method is used in technical applications.

We should also mention what does virtual reality mean because it is a key component for

```
skill(plasticBomb).
skill(bioBomb).
-skill(nuclearBomb).
safetyArea(field1).

@p1
+bomb(Terminal, Gate, BombType) : skill(BombType)
<- !go(Terminal, Gate);
disarm(BombType).

@p2
+bomb(Terminal, Gate, BombType) : ~skill(BombType)
<- !moveSafeArea(Terminal, Gate, BombType).


@p3
+bomb(Terminal, Gate, BombType) : not skill(BombType) &
not ~skill(BombType)
<- .broadcast(tell, alter).


@p4
+!moveSafeArea(T,G,Bomb) : true
<- ?safeArea(Place);
!discoverFreeCPH(FreeCPH);
.send(FreeCPH, achieve,
carryToSafePlace(T,G,Place,Bomb)).
```

Figure 2.6: Examples of AgentSpeak Plans

virtual training environments. Virtual reality provides users the feeling of being inside the virtual environment. It is called presence [37] The more realistic environment is, the more presence user feels. Presence can be provided by stimulating the senses of users. This can be done by continuous visual, audio or haptic streams [35].

After giving brief information about virtual environment and virtual reality, we can focus on the main concept that the rest of the thesis based on. As we mentioned above, virtual environments are used to simulate the real world. So why can not we train people with these virtual environments without needing real environments? Generally trainees take certification tests after the learning progress which is provided by an expert of that field. This kind of training has been used successfully in the past. However, it is clear that the cost of this training method is very high because of the resources it requires [35]. The other problems of real-world trainings are; mistakes that the trainees done can lead to personal or property damage and not finding the opportunity to perform some process on a frequent basis so that the trainee can not master that process. Virtual environment-based training has advantages as we consider these issues. There are no risks of personal or property damage. Also trainees can perform the same tasks as frequent as they want because the system has the ability to repeat the same process. General advantages of virtual environment-based training is listed below [35]:

- It can occur anytime without needing any assistance.
- Cost savings can be realized by reducing assistance and component usage during the practice.
- It is safe.
- It can be repeated multiple times.
- Gives the trainee an opportunity to analyze the process from different perspectives.

According to Gupta et al.[35] there are also disadvantages of virtual environment-based trainings. These are:

- Users may not be able to transfer what they learn in virtual environment to the real world with a full success.
- Some users can experience motion sickness.
- The equipment needed for the virtual environment can be high.
- Special software must be developed
- Tutorials are time-consuming to create.

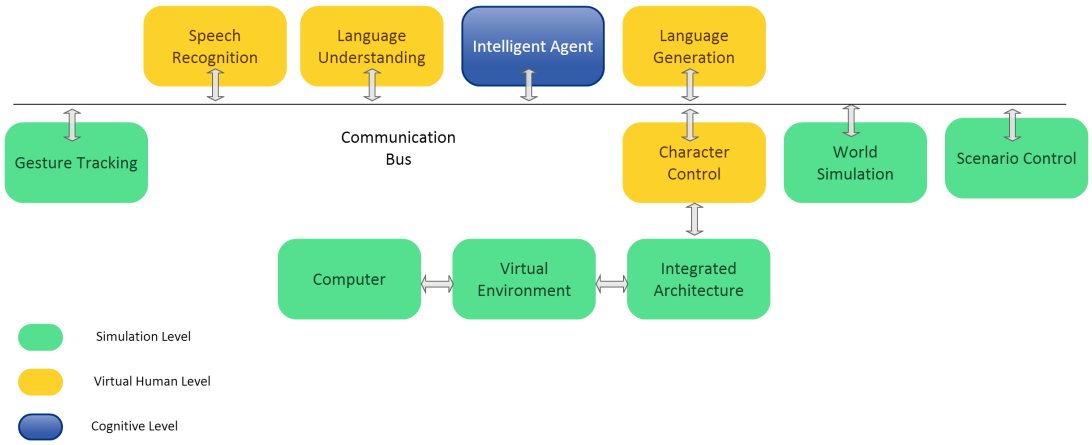


Figure 2.7: Virtual Human Architecture Overview

There are many researches and implementations of virtual training environments with intelligent agents. Virtual Human project [38] aims to fill the gap of training leadership, negotiation, cultural awareness and interviewing skills. Kenny et al. [38] states that, existing virtual environments like simulations and computer games offers intelligent agents that provide training for physical skills or strategy but lack of previously mentioned skills. It is also stated that virtual worlds can be used for various training applications if the agents has these three main characteristics; believable, responsive and interpretable. According to Kenny et al. [38] virtual humans should include three layers; cognitive layer which makes decisions, virtual human layer which includes input and output processing and simulation layer which does everything else that has to do with the environment that virtual human situated. Virtual humans have the same structure with BDI systems. They both implement sense-think-act cycle. The aim of the research is integrating these three layers and find ways to use these layers in virtual training environments effectively. They remark that virtual training environments can only be successful if various technologies are integrated together. Figure 2.7 shows the overview of the Virtual Human integrated architecture.

The research includes cognition and emotion modeling, natural language processing, knowledge representation, speech recognition and many key areas that make an virtual human useful for a virtual training environment.

Virtual Human implemented to wide range of applications such as military training to virtual patients. Kenny et al. [39] evaluated the systems with the criterias of performance, interactivity, believability and feedback from users or trainees. As the result they state that there are many trade off for these systems but these systems are definitely a story of the future. The issues these systems have are; quality of artwork, difficulty of designing the dialogue, lack of procedural system and the hard process of building the whole domain.

In another research, Barot et al. [39] present the outcomes of the V3S project. The aim of the research is to train learners in virtual environment for high risk industries as risk management training has become a major issue. Virtual environments for risk-management bring similar advantages we mentioned before such as risk reducement and specific situation reproduction. Different from other training purposes, in high risk industries, learners should be practised in damaged work conditions or difficult situations. According to V3S project, virtual environment efficiency increases when trainees have to chance of seeing the results of the decisions they make.

The autonomous character architecture in V3S is based on BDI model. They [39] proposed a multi-agent framework which takes into account several characteristics that real human has, like tiredness or stress. This multi-agent framework uses some models that allow virtual characters to deviate from the ideal actions and display errors. Thus the virtual environment becomes more realistic.

The V3S project has two kinds of application prototypes. The first one is substitution of pipes in chemical-processing sites. The user plays as the manager and manages a team of virtual characters. Second one is loading hazardous material on oil depots where the user is responsible for his own actions. The environments were in 3D and the second prototype has two options of interactions. One is with keyboard and mouse whereas the second one is more immersive with motion capture system and stereoscopic visualization. One of the goals of the research is measuring the degree of usability of these two kinds of interactions. Other main goals of the research are; to examine the suitability of the system with the real conditions of training with professional trainers and to assess the acceptability of the system.

They used System Usability Scale questionnaire [40] to evaluate the efficiency of the system. The results showed that the system can be used in real training sessions. Barot et al. [39] state that the system already met many needs of training in risk management however there are many components need to be improved like interactions without markers attached on the users, the reactions of the virtual characters to the user actions and so on.

Buche et al. [41] proposed MASCARET model that provides social, cognitive and reactive abilities to agents to simulate an environment. This model also organizes the interactions between agents. These interactions make the environment social. One another important capability that the model offers is providing the ability to evolve of the autonomous agents. The users participate in the environment with their avatars. They used the model in a fire-fighters training environment to demonstrate the validity of the model.

Buche et al. [41] defines the key aspects of virtual environments as being heterogeneous and open. They also think that users of the virtual environments should be thought as other autonomous agents because they interact with other agents and environment in the same way as autonomous agents do. The concepts of the proposed organizational model are organisation, roles, behavioural features and agents. The model is derived into the organisations; physical and social environment that each of them is a must for a virtual training environment. The physical environments should be realistic, interactive and real-time. In the research, they de-

fine source-target interactions with a privileged direction. They also define a recruiting role which maintains the knowledge of agents.

As mentioned above, the users in the MASCARET model are also rational agents that are reactive and has autonomous actions. The link between the autonomous action module and the decision making module for the user is limited by the model so that the user takes the control of the behaviours of the agents. The avatar is still in interaction with other agents and updates other agents with the actions it does and updates its own knowledge with the other agents actions. They integrate the MASCARET with Intelligent Tutoring System (ITS) that aims to tutor students. ITS has four components; domain model, learner model, pedagogical model and interface model. But generally it uses domain and learner model. The aim of the domain model is to consult roles responsibilities. On the other hand the aim of the learner model is to provide the state of the student's knowledge. ITS also represents additional models as pedagogical model, error model and the interface model. Pedagogical model has the strategies about how to teach knowledge. The error model contains the information about the pedagogical errors that students generally do. The interface model presents information to the student.

In ITS, avatars play a pedagogical role. There are four actions used; Explain, Suggest, Show and Disturb. Explain action gives information about the current goal. Suggest action informs the user about the next action. Show action demonstrates the action and lastly disturb action modifies the behaviour of the agents playing.

Lastly, they present a civil security application called SECUREVI [41]. SECUREVI is developed to demonstrate the MASCARET model. The goal of the application is to train the fire officers for operational management. The user controls an avatar and perform the pedagogical actions that mentioned above. Users can explain how something is done during the training session, suggest what to do, show how to do and also disturbs the the physical environment.

Syscara [9] also lists the challenges in the design and the implementation of multi-agent systems.

- Describe and decompose problems among a group of agents
- Communication of the agents
- Interaction of the agents
- Consistent decision making
- Stable behaviours
- Agents' state representation of other agents
- Resolving conflicts between agents
- Constrain in practical distributed AI systems

CHAPTER 3

PROPOSED METHOD

Virtual training applications can be used as an alternative to real-world trainings. These applications also bring some advantages with respect to real-world trainings as mentioned in the previous sections. The suitability of BDI model for virtual training applications which contain multi-agent systems is shown in many research and implementation. Within the scope of the thesis, we focus on a specific kind of virtual training environment, virtual classroom. We modeled the environment and all the autonomous students, which have different characteristics, using the BDI model.

In this section we present our proposed method to train teacher canditates in an interactive virtual classroom which is developed using Unity3D and contains BDI agents developed with Jason multi-agent programming language. First we will explained the conceptual design of the classroom, then we will discuss the architecture that we used to build the environment and why we chose Agent Oriented Programming and BDI model for this research.

3.1 Conceptual Design of the Virtual Classroom

The very first thing that we have to mention before going into detail of the system architecture is the ”interactable object” concept. Every object can be interacted is marked as ”interactable object” so that, the interactions can be handled easier. The agents, teacher and all other environmental objects like door, desks, books are marked as interactable objects.

Students are modeled according to BDI architecture. They have beliefs, desires and intentions. We can define beliefs as ”obervations” and ”mental and physical factors” for students. They observe the environment, the classroom in our scenario, updates its previous observations and also calculate new mental and physical factor levels depending on these observarions. Students have desires which can be defined as goals or options that they can perform during the simulation. Finally, they are committed to one of these desires which can be defined as student’s intention. These three component of the conceptual design for virtual classroom will be explained in details.

Table 3.1: Mental Factors

Mental Factor	Abbreviation
Attention	Att
Desire to Talk	Dtt
Desire to Ask Question	Dtaq
Teacher's Treating Hardness	Th
Tension of the Classroom	Ten
Teacher's Control Over Class	Coc
Respect Teacher	Res
Understand Lesson	Ul

Table 3.2: Physical Factors

Physical Factor	Abbreviation
Energy	En
Noise	N

We proposed a heuristic model to represent the student characteristics and behaviors according to an ongoing research [42]. Agents that represent students have 8 different mental factors and 2 physical factors which change over time. The behavior of the students are determined depending on these mental and physical factor levels. Mental and physical factor levels are calculated every cycle and updated in the students' belief base. These factors also can change as the lesson proceeds. These mental factors are shown in Table 3.1. Physical Factors are shown in Table 3.2.

Attention: Attention factor indicates how much the student is intented to focus lesson and listen to the teacher. This factor is also effected by the duration of the lesson. As the lesson proceeds, the attention level of the student decreases faster (Equation 3.1).

Desire to Talk: Desire to talk factor indicates how much the student is intented to start or be included in a conversation. As this mental factor increases, the possibility of the student to talk with other student increases (Equation 3.2).

Desire to Ask Question: Desire to ask question factor indicates how much the student is intented to ask a question to teacher. This question can be either about the lesson or about asking permission. As the lesson proceeds, the possibility of student asking question increases (Equation 3.3).

Treating Hardness: Treating hardness factor indicates how the teacher acts to class according to that student. This mental factor level is effected by all the actions that the teacher performs. If the teacher gets tough with the class this factor level will increase (Equation 3.4).

Tension of the Classroom: Tension of the classroom factor indicates how much tension is in the classroom. Again the actions of the teacher and noise in the environment effects this mental factor level (Equation 3.5).

Understand Lesson: Understand lesson factor indicates how much student understands the lesson. This mental factor is effected by the characteristic of the students as well as other mental factors and observations (Equation 3.6).

Teacher's Control Over Class: Teacher's control over class factor indicates the level of teacher's control over class according to the student. All the actions which teacher performs effect this factor level. The goal of the users are trying to keep this mental factor level high because this factor shows how they perform in the virtual classroom (Equation 3.7).

Respect Teacher: Respect teacher factor indicates how much the student respects the teacher and appriciate his behaviors. This mental factor is effected by the characteristic of the students as well as other mental factors and observations (Equation 3.8).

Noise: Noise indicates the number of conversation in the classroom.This physical factor effects the students mental factors like tension of the classroom or attention of students.

Energy: Each student has a total amount of energy to perform actions. When a student performs actions like walking, talking or focusing lesson the energy level decreases. In contrast, if a student sit down or daydream in lesson, his energy level increases.

Mental and physical factor levels can not be calculated just by using other mental and physical factors. There are some other changes happening in the environment and the students have to observe these changes in order to make a proper determination of action to perform. These are called observations. Observations defined for students are listed in Table 3.3.

Another critical component of agents are states. There are various states for students defined which effect mental and physical factor levels. Student states are also used while determining the next behavior as in finite-state-machines. These states are listed in Table 3.4.

In real life, students can have many characteristics and their tendency to perform specific behaviors change according to these characteristics. We had to simplify these complex characteristics into several major basic characteristics for making it possible to implement.

Characteristics of the agents should be a component that effects the decision making and the way they behave. Because of the differences in their characteristics, changes in their mental and physical factors can differ too. Therefore, they intend to do different behaviors. The characteristic factors of the agents are listed in Table 3.5.

Table 3.3: Observations

Observations	Abbreviation
All Conversations Count	Acc
Walking Students Count	Wsc
Recent Conversation Request Count	Rcrc
Recent Given Permission Count	Rgpc
Recent Rejected Permission Count	Rrpc
Recent Answered Questions by Teacher Count	Ransc
Recent Asked Questions by Teacher Count	Raqc
Recent Asked Questions to Teacher Count	Raqsc
Recent Unable to Answer Question Count	Ruansc
Recent Answered Questions Count	Raqsc
Recent Given Examples by Teacher Count	Rgetc

Table 3.4: Student States

Student States
Standing
Sitting
Walking
Talking
Has Question
Has Answer
Doesn't Have Answer
Asked Question
Want to Leave Class
Want to Get Inside Class
Raising Hand
Ready For Lesson
Focused Lesson
Daydreaming

Table 3.5: Character Factors

Characteristic Factors	Abbreviation	Effected Mental Factors
Comprehensive Level	Com	Att, Dtaq
Social Ability Level	Sa	Dtaq, Dtt
Talkative Level	Tal	Dtt
Respectfulness Level	Resful	Res, Dtt

Table 3.6: Character Types

Character Types
Balanced
Talkative
Clever
Energetic
Lazy

Comprehensive Level: Comprehensive level of students effects mental factors like ”attention”, ”desire to ask question to teacher” or ”understanding lesson”.

Social Ability Level: Social ability level of students effects mental factors like ”desire to ask question to teacher” or ”desire to talk with other students”.

Talkative Level: Talkative level of students effects ”desire to talk with other students”.

Respectfulness Level: Respectfulness level of students effects mental factors like, ”respect teacher” or ”desire to talk with other students”.

These characteristic factors provide a basis for the defined characteristics of students. In [43], there are more than 10 different characteristics just for problematic students like, underachiever, passive aggresive, hyperactive or distractible. We had to reduce these characteristics and generalize them to make it feasable to implement. By using the characteristic factors, we proposed five different characteristics. These characteristics are shown in Table 3.6.

Each student characteristic has different coefficients for each characteristic factor. These coefficients are shown in Table 3.7.

As mentioned before, mental and physical factors are calculated using other factors and also observations. Every effecting factor has a level percentage that indicates how much it effects

Table 3.7: Characteristic Coefficients

CharacteristicType	Comprehensive %	Social Ability %	Talkative %	Respectfulness %
Balanced	50	50	50	50
Talkative	40	70	70	40
Clever	90	60	30	80
Energetic	50	80	60	50
Lazy	25	60	50	30

that factor level. For example; attention level is calculated considering energy and tension levels. Other effecting observations are ”moving student coun” and ”recent conversation request count”. All the formulas of mental factors are listed below.

$$
\begin{aligned}
Att = {} & (En * EnEffectPerc) + (Ten * TenEffectPerc) \\
& + (Com * ComEffectPerc) + (Dtt * DttEffectPerc) \\
& + (Wsc * WscEffectPerc) + (Rcrc * RcrcEffectPerc) \\
& + (Ranssc * RansscEffectPerc) + (Raqc * RaqcEffectPerc)
\end{aligned} \tag{3.1}
$$

$$
\begin{aligned}
Dtt = {} & (En * EnEffectPerc) + (Ten * TenEffectPerc) \\
& + (Th * ThEffectPerc) + (Att * AttEffectPerc) \\
& + (Tal * TalEffectPerc) + (Sa * SaEffectPerc) \\
& + (Resful * ResfulEffectPerc) + (Rcrc * RcrcEffectPerc)
\end{aligned} \tag{3.2}
$$

$$
\begin{aligned}
Dtaq = {} & (Coc * CocEffectPerc) + (Att * AttEffectPerc) \\
& + (Sa * SaEffectPerc) + (Com * ComEffectPerc) \\
& + (Ransc * RanscEffectPerc) + (Rgpc * RgpcEffectPerc) \\
& + (Rrpc * RrpcEffectPerc)
\end{aligned} \tag{3.3}
$$

$$
\begin{aligned}
Th = {} & (Ransc * RanscEffectPerc) + (Rrpc * RrpcEffectPerc) \\
& + (Rgpc * RgpcEffectPerc) + (Wsc * WscEffectPerc)
\end{aligned} \tag{3.4}
$$

$$
\begin{aligned}
Ten = {} & (Th * ThEffectPerc) + (N * NEffectPerc) \\
& + (Rrpc * RrpcEffectPerc)
\end{aligned} \tag{3.5}
$$

$$
\begin{aligned}
Ul = {} & (Att * AttEffectPerc) + (Com * ComEffectPerc) \\
& + (Raqsc * RaqscEffectPerc) + (Ransc * RanscEffectPerc) \\
& + (Rgetc * RgetcEffectPerc)
\end{aligned} \tag{3.6}
$$

$$
\begin{aligned}
Coc = {} & (Th * ThEffectPerc) + (Att * AttEffectPerc) \\
& + (Ten * TenEffectPerc) + (N * NEffectPerc) \\
& + (Wsc * WscEffectPerc)
\end{aligned} \tag{3.7}
$$

$$
Res = (Coc * CocEffectPerc) + (Resful * ResfulEffectPerc) \tag{3.8}
$$

The calculation of mental and physical factor levels are done according to the formulas, characteristic factors and coefficients explained above. After the calculations of the mental and physical factors, the decision making is done according to these factor levels by depending on the ”rule”s for the agent which is defined in implementation stage by designer. Some examples of these rules are shown in Figure 3.1.

```
canWalkDuringLesson :-
            level(energy, En) & level(attention, Att)
            & En >= 80 & En < 90
            & Att < 40
            & lesson(started).

canAskQuestionDuringLesson :-
            not asking(Q)
            & level(desireToAskQuestion, Dtaq) & Dtaq > 50
            & lesson(started)
            & has(question, Q)
            & listen(teacher).

canFocusLesson :-
            not listen(teacher)
            & level(attention, Att) & Att >= 60
            & level(energy, En) & En > 50
            & lesson(started)
            & teacher(teaching).
```

Figure 3.1: Rule Examples

After the decision making progress, agent commits to an action which can be defined as intention that is most proper for him. Then he performs the corresponding plan step-by-step.

Agents can interact with the objects in the environment. For example they can open the door, put their book on to the desk etc. Agents can also interact with each other by communication. They can start a conversation, listen to another agent or the teacher, look at another agent etc. Some actions can change the conditions of the environment and these conditions effect agents' mental and phsical factors. For example, when an agent starts to walk during the lesson, attention of other agents decrease.

The agents have different action queues like ”Conversation Queue” or ”Change Position/State Queue”. This provides an agent to perform concurrent actions that are not related to each other like walking and talking. Thus an agent does not have to wait for unrelated action to finish to perform another action.

As PRS continuously checks the changes in environment and reconsider it's choices, the actions can be interrupted during their performing phase. Therefore, in the multi-agent system we built, the agents can interrupt their current actions if another intention is more suitable to commit.

Users control an avatar which represents a virtual teacher in the class. Users can interact with the students by performing various actions. Every action of the user effects the environment

Table 3.8: User Actions

User Actions	Specific Student Effects	Class Effects
Teach Lesson	-	Att, Dtt
Stop Teaching Lesson	-	Att, Dtt
Warn to Get Seat	-	En, Att, Ten, Th
Warn to Sit Down	En, Att, Ten, Th	Att
Warn to Sit Down Angrily	En, Att, Ten, Th, Dtt, Dtaq	En, Ten, Dtt, Dtaq, Th
Warn to Stop Talking	Att, Ten, Dtt, Dtaq, Th	Ten, Dtt, Dtaq, Th
Warn to Focus Lesson	Att, Ten, Dtt, Dtaq, Th	Att, Ten, Dtt, Dtaq, Th
Call Student's Name	Att, Ten, Dtt	Att
Look at Student	Att, Th	-
Ask Question to Class	-	Att

and consequently the virtual students' behaviors. Users can perform agent specific actions like "Warning a Student to Sit Down" or "Calling Student by Name". Moreover, they can also perform environment based actions like "Teaching Lesson" or "Warning to Be Quite". The user can move through the virtual classroom too. Thus, users have the chance to deal with students by arranging the distance between him and students. The actions done by the user effect the mental and physical factors of the students. Depending on the interventions mentioned in [43], we proposed three levels of actions. They are; soft, default and harsh. The action types user prefers to perform, identify the user's characteristic and how the students percieve him. Each action has an effect on different levels of factors. Student specific actions can effect all the agents in the class with more effect on the agent that is interacted and less effect on the other agents. Environment based actions effects all the agents depending on their coefficients. Some actions and their effects on specific agent or class are shown in Table 3.8.

As we mentioned above, there are various mental factors defined for students. User actions effect these factors. The main aim of the user is coping with each student and keeping the student mental and physical factors in the optimum level. For example, the higher "Understand Level" or "Teacher's Control Over Class" factor is, the better user performed. On the other hand, the user should try to keep "Tension" or "Treating Hardness" level low.

There are also tasks integrated in the system that the user should achieve to perform better in the training simulation. The system guides and trains the trainee by using these tasks. Tasks are not obligations for successing or training. Some of these tasks are listed in Table 3.9.

Table 3.9: Tasks

Tasks
Focusing All Class
Focusing Daydreaming Student
Focusing Late Student
Interacting With Students
Using Sense Of Humour

3.2 Architecture of the System

In the scope of this thesis, we used BDI model with Agent Oriented Programming paradigm to implement the virtual classroom. The main structure of BDI is explained in detail and the suitability of the model for the purpose of the research is shown depending on related works in previous sections. Also the differences between Agent Oriented Programming and Object Oriented Programming is discussed. In this section we will discuss the reasons we used Agent Oriented Programming with BDI model for the proposed method.

Agent Oriented Programming architecture can be defined as an abstracted form of Object Oriented Programming. This abstraction provides us more flexibility while designing intelligent agents with respect to Object Oriented Programming. Also it makes the implementing agent behaviors convenient because of the abstraction. Moreover, we are not renouncing the features that Object Oriented Programming paradigm offers to developers. The working mechanism of Unity3D Game Engine is also similar to what Agent Oriented Programming architecture offers. Each gameobject defined in Unity3D behaves as an agent and it's behaviors are defined within the scripts attached to it. They can interact with each other at the knowledge level . They are autonomous and have specific objectives to achieve.Thus the agents in multi-agent system can be easily matched with the gameobjects defined in Unity3D.

We proposed using BDI model for virtual students. In most of the real-time applications like video games, finite state machines are used to build intelligent agents because of its performance advantages. On the other hand, designing a complex system and solving a complex problem with finite state machines can be too difficult and complicated. Because, the developer has to define all the states of intelligent agents clearly. Thus, we prefered to use BDI model instead of finite state machines to model the student behaviors in a more simple way without defining all the states. Bartish [44] states that FSM implementations become impossible to control with increasing number of agent behaviors. One disadvantage of BDI is the performance issues that can arise from huge number of agents. However, this factor is out of this research's scope as we are using maximum 6 agents in the system.

The system has two main components that communicate between each other. First one is the MAS component that makes the decision making and means-end reasoning. The agents are created and added to the environment in this component. In every cycle, the system reconsider the options that are available for the agents and choose the best option for them to be committed to.

The second one is 3D modeling component that is used to present the classroom in a 3D way and perform the actions which are ordered from the MAS component. For this component, we used Unity3D game engine as mentioned before. The representation of virtual students, teacher and other environmental items are done within this component. Each agent defined in MAS component has a matching gameobject in Unity3D. These gameobjects represent the students visually. This component can be counted as the front-end of the system. Thus, users do all the interaction with this component.

There is a local socket networking system between these two components which makes the system work as a whole. The steps of the main cycle of the system are listed below.

- MAS decides which action will be performed by which agent.
- The information is transferred to modeling component.
- Modeling component performs the action and updates the environment.
- The changes in the environment are transferred to MAS.
- MAS updates the beliefs of the agents.

As the system uses BDI agents, it should be implemented in BDI programming language, in the scope of this thesis, Jason. Logics of the agents are programmed using Jason. In regular Jason projects, the actions of the agents and the environment are implemented using Java. However, as we use Unity3D in our implementation, the actions of the agents and also the environment should be implemented in Unity3D. So they are programmed using the scripting language of Unity3D, C#. Therefore, agents have logical code which represents the initial beliefs, initial goals, all plans; also scripting code which represents the actions of agents that are done in the environment.

Users or designer of the system can define how many students will be included in the virtual classroom by arranging the number of students from the editor. The student count of the system is limited with the number of desks placed in the classroom. We designed the case studies with a maximum student number of 6. We did not try to increase the number as it is not in the scope of this thesis. Users can define the characteristics of the students by selecting from the editor so that they can deal with various situations in each time they run the simulation. The students placed in the environment are informed to the MAS component and the properties of the students are transferred to MAS to generate the agent on the other side. Another feature of the system is adding and removing students to the classroom in runtime. Thus, users or

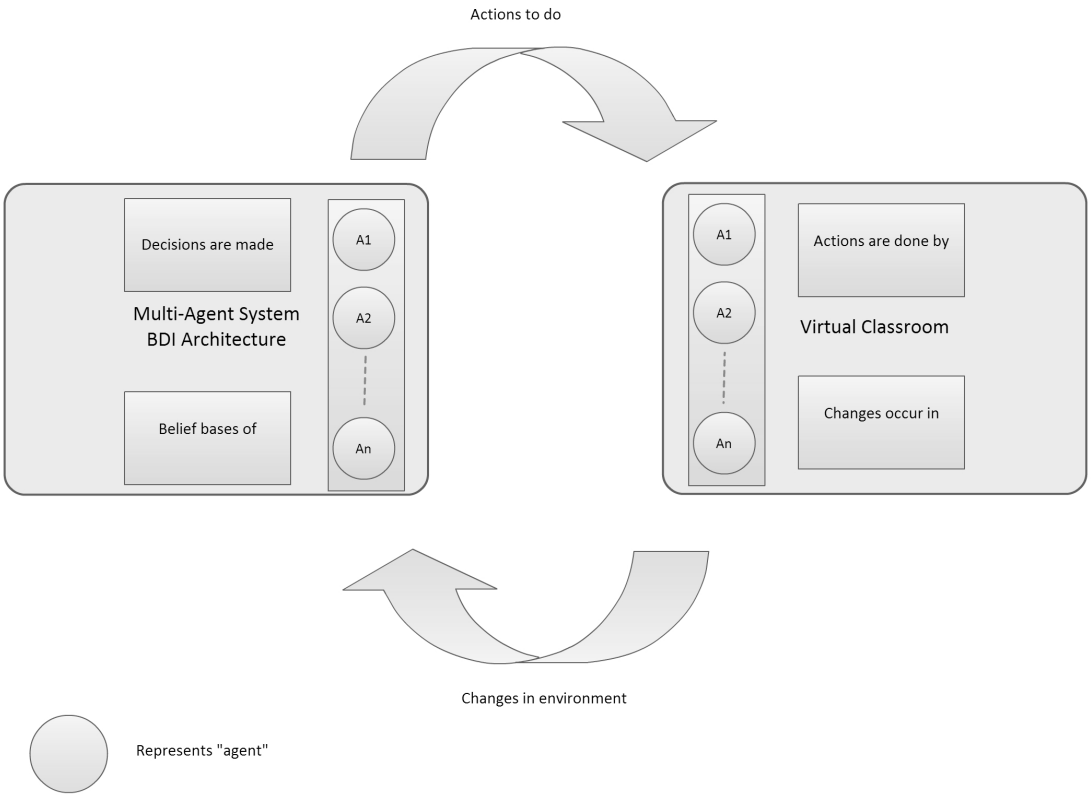


Figure 3.2: Arctihecture of Proposed Method

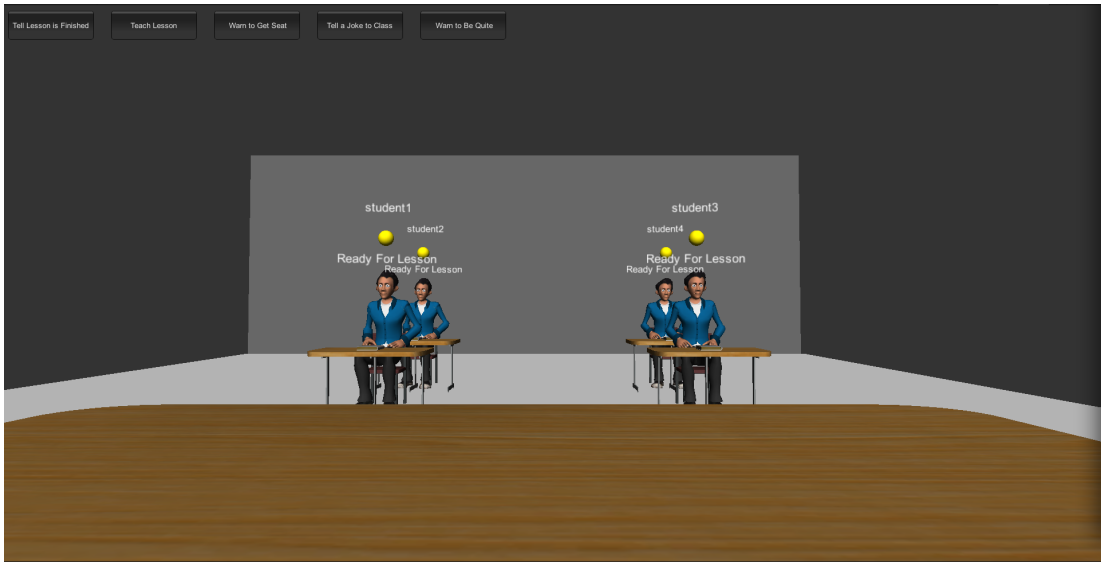


Figure 3.3: User Interface

designers can add new students which can be considered as late students and the trainee should deal with that kind of situation.

There is a user interface designed for users to interact and perform actions to deal with the students. Users can choose actions to perform by clicking the relevant button on screen. The action selection section of user interface is shown in Figure 3.3. Users can also track the mental and physical states of the students during the simulation. The mental and physical factors are shown as level bars in the user interface. Moreover, student states like ”Talking” or ”Daydreaming” is represented above the their 3D model so that the user can act directly to those students to interact. When the user clicks on a specific student these level bars will appear in the user interface. It is shown in Figure 3.4.

As mentioned before, there are tasks integrated into the system that guide teacher to deal with the class. Thus, users should see and track these tasks to check if they have completed them or not. We designed a simple menu that shows all the tasks and the tasks which are completed by the user. This menu is shown in Figure 3.5.

To analyze the data we designed another window that shows the real-time graph of students' mental and physical factor changes in time. We are also collecting each student's factor data and saving it in a XML document so that it can be possible to analyze and evaluate overall performance of the user later. We also developed a tool to generate graphs of each student from saved data. We will be using this tool when discussing the results of the system in the next section.

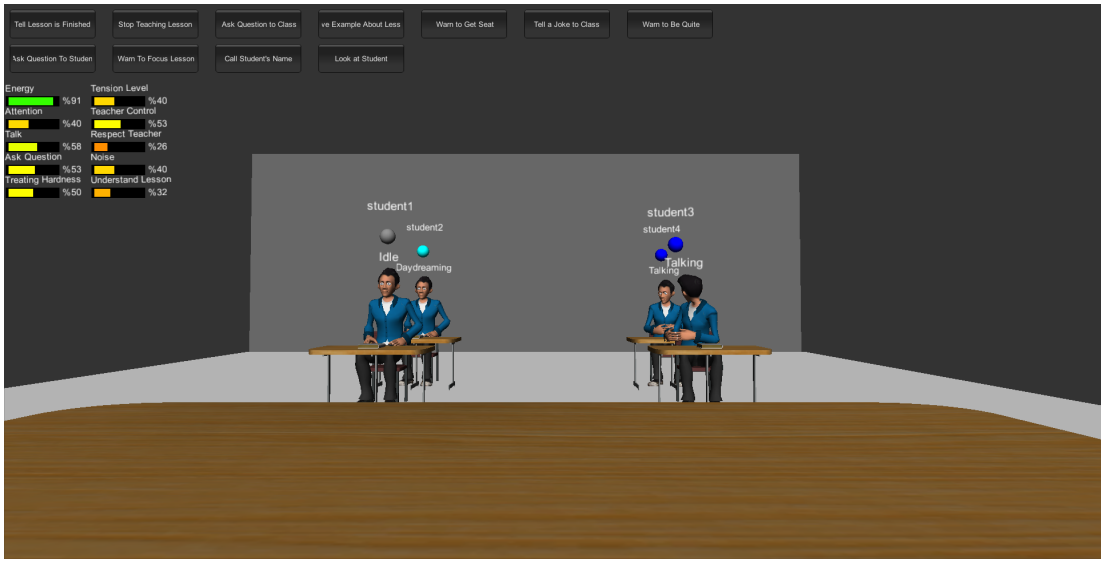


Figure 3.4: Level Bars

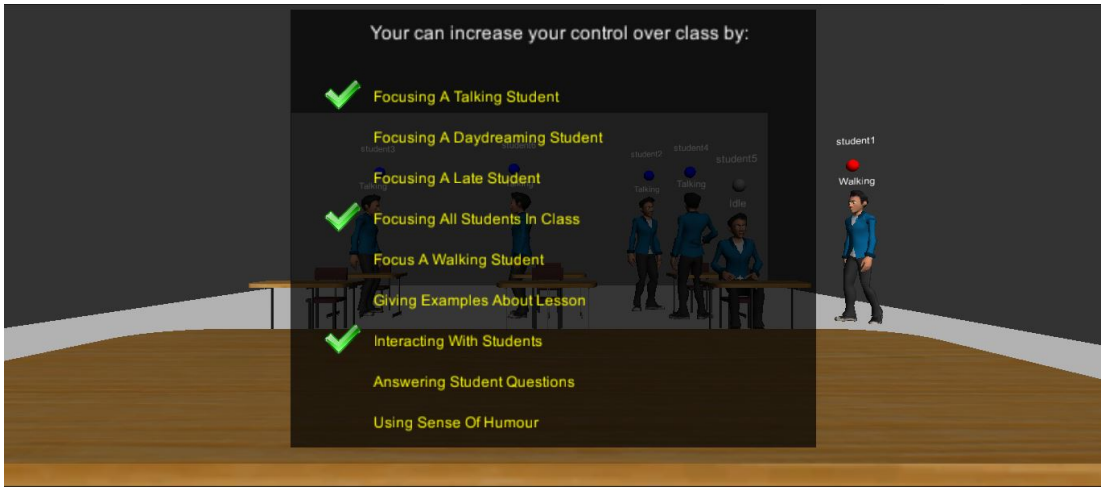


Figure 3.5: Tasks Window

3.3 Implementation Detatils

Two most important components of the implementation are Jason interpreter which we mentioned in previous sections and Unity3D game engine.

Unity3D [45] is one of the most popular 3D game engines in the industry. It has reached

2.000.000 registered users with thousands of game published. Unity makes visualization, handling script execution, organizing the game scene a lot easier. It is built on Mono framework [46] which is an alternative to .NET Framework but open-source and allows developers to build cross-platform applications. One of the main reasons we chose Unity3D is because it supports C#, one of the most used languages today. Unity3D also allows to use .NET libraries which makes programming convenient.

CHAPTER 4

DISCUSSION AND RESULTS

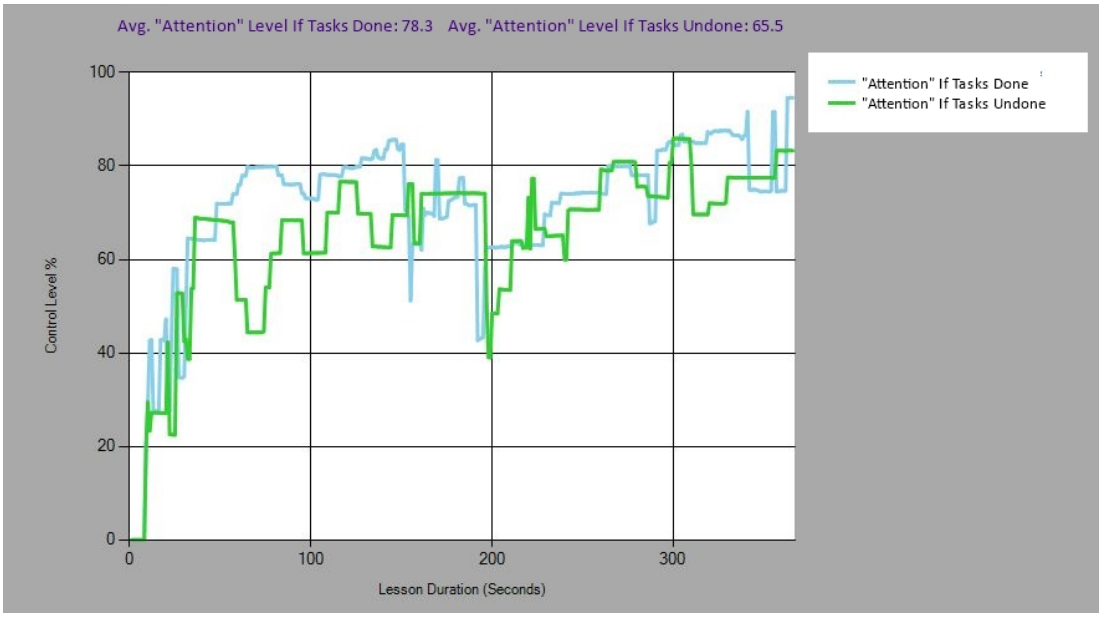
We designed two different case studies to evaluate the proposed method. Our aim was to examine the realism and the accuracy of the implemented system by using BDI model. First, we will describe the case studies and how we compare the results we got from our approaches. Then we will discuss the results and accuracy of the system according to the results. All the case studies were done with 5 users who were all university students with the age of between 22 and 25 with no teaching experience and the discussions are made according to the avarage scores of the users.

4.1 Handling A Balanced and Problematic Class - Guiding The User

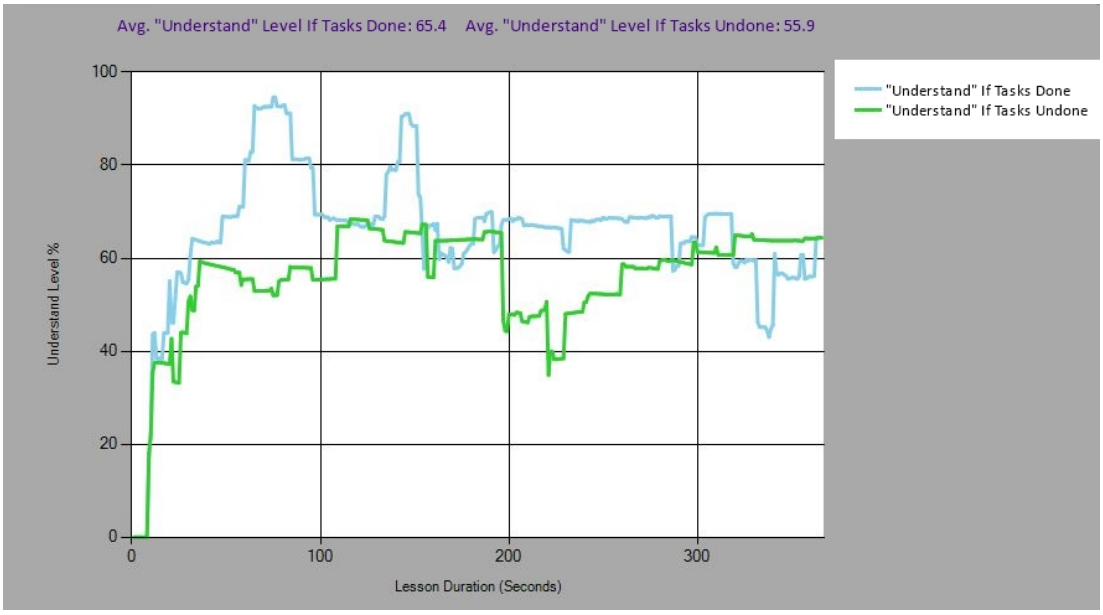
As we mentioned in the previous secions, we designed several tasks for the users to direct them through the simulation. Thus, they can perform better with more ”control over class level” or ”understand lesson level”. In this case study, we measured the accuracy of the system and the effects of the tasks. The simulation was played in two different ways to get compareable results. In the first turn, users tried to complete the given tasks as possible as they could and in the second turn, they tried not to complete the tasks.

For the first part of this case study, we built up a classroom with six students. The distribution of the characteristics of studens were as follows: 2 talkative students, 1 clever student, 1 lazy student, 1 energetic student and 1 balanced student added to the classroom after a while. We tried to form the classroom in a balanced way that all the characteristics appear almost at the same rate. The reason we put 2 talkative students in the class was, increasing the possibility of conversation that can be started. The results of the experiment are shown in Figure 4.1, Figure 4.2, Figure 4.3.

In first approach, users completed all the tasks through the simulation to see the effects of tasks we designed. Contrarily, in the second approach they tried not to complete any tasks but some of the tasks were completed unintentedly. The results showed that tasks have good effects on students' mental factors. As seen in Figure 4.1(a) and 4.1(b) students are more disposed to pay

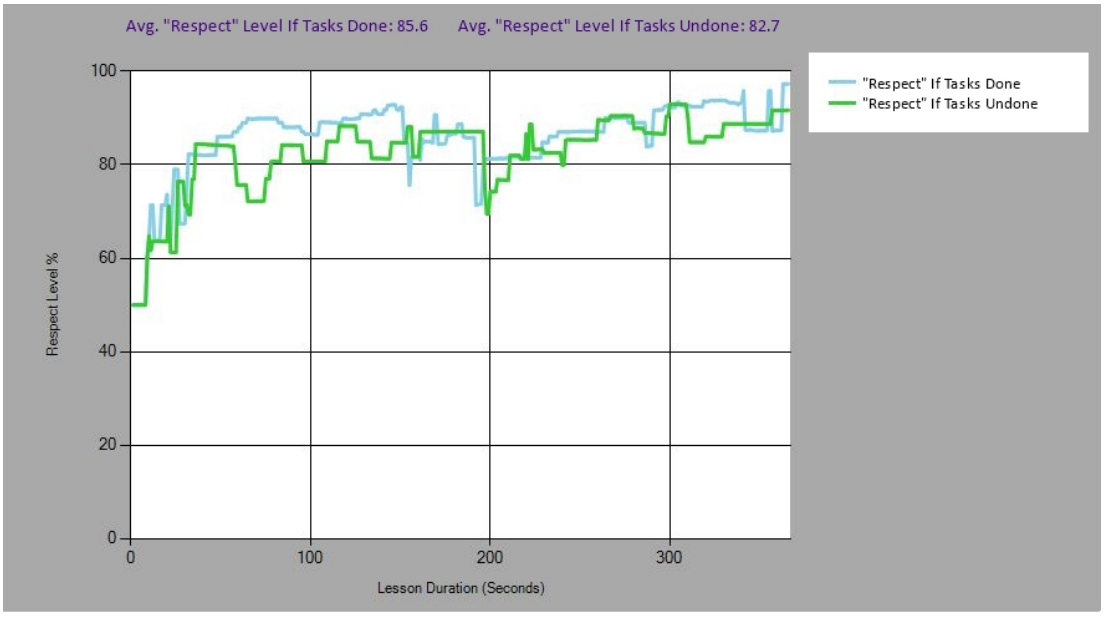


(a)

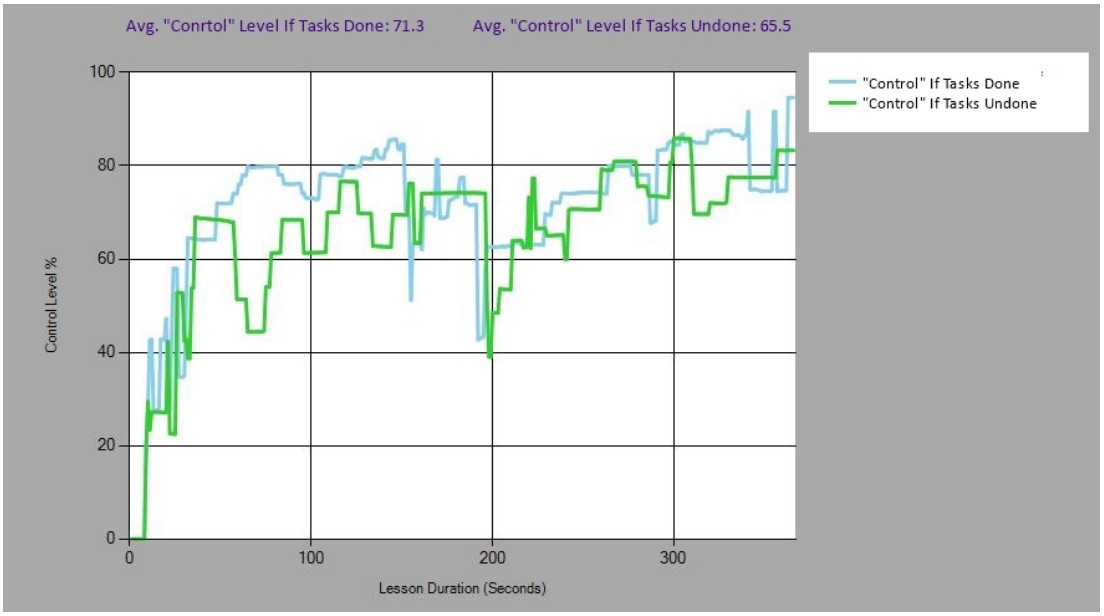


(b)

Figure 4.1: (a) Student's "Attention" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Understand Lesson" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green



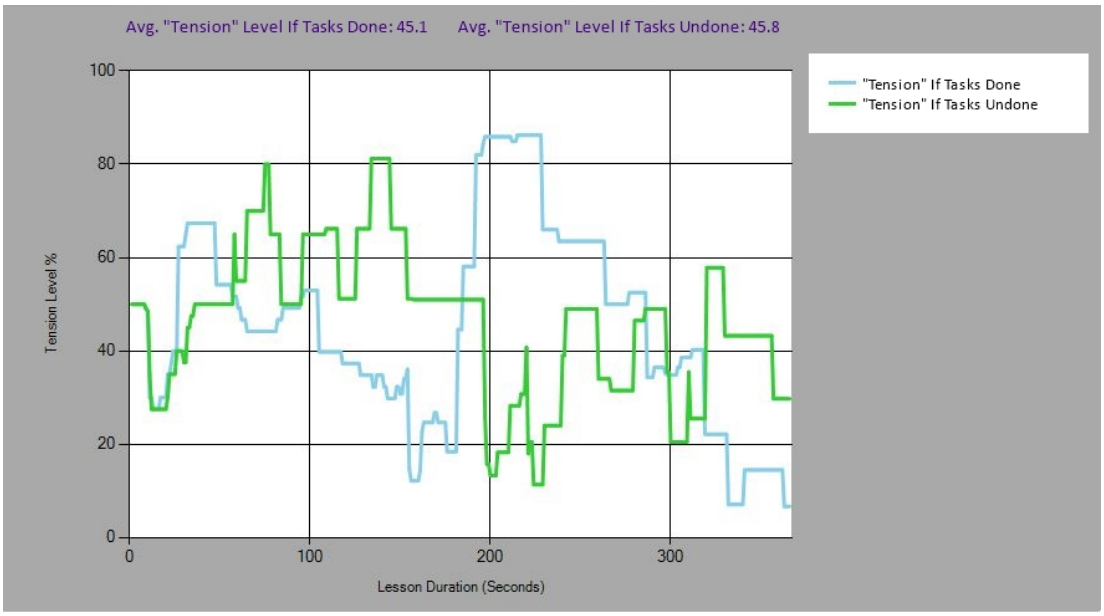
(a)



(b)

Figure 4.2: (a) Students' "Respect Teacher" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Teacher's Control Over Class" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green

(a)



(b)

Figure 4.3: (a) Students' "Teacher's Treating Hardness" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Environment's Tension" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green

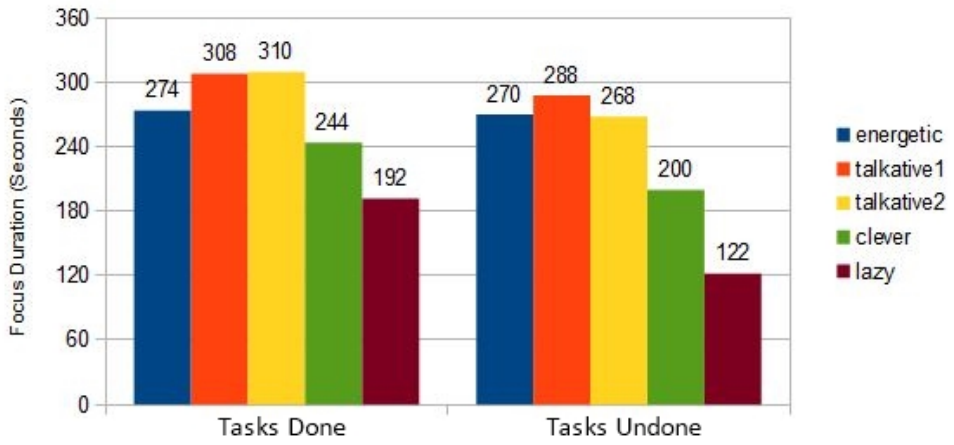
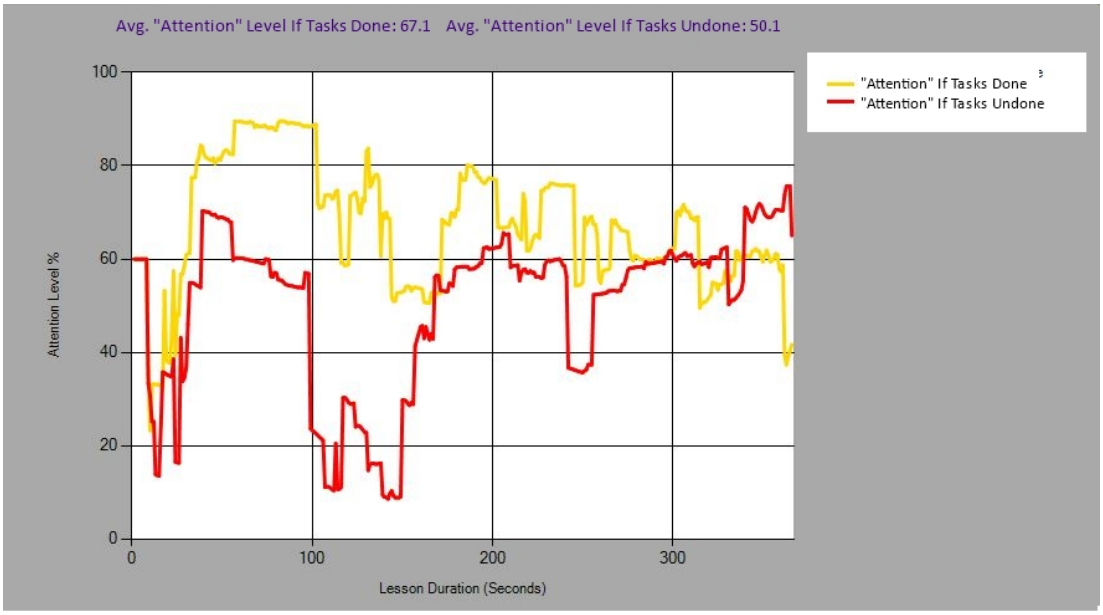


Figure 4.4: Students' focus durations for a balanced distributed class if user tries to complete and not complete given tasks

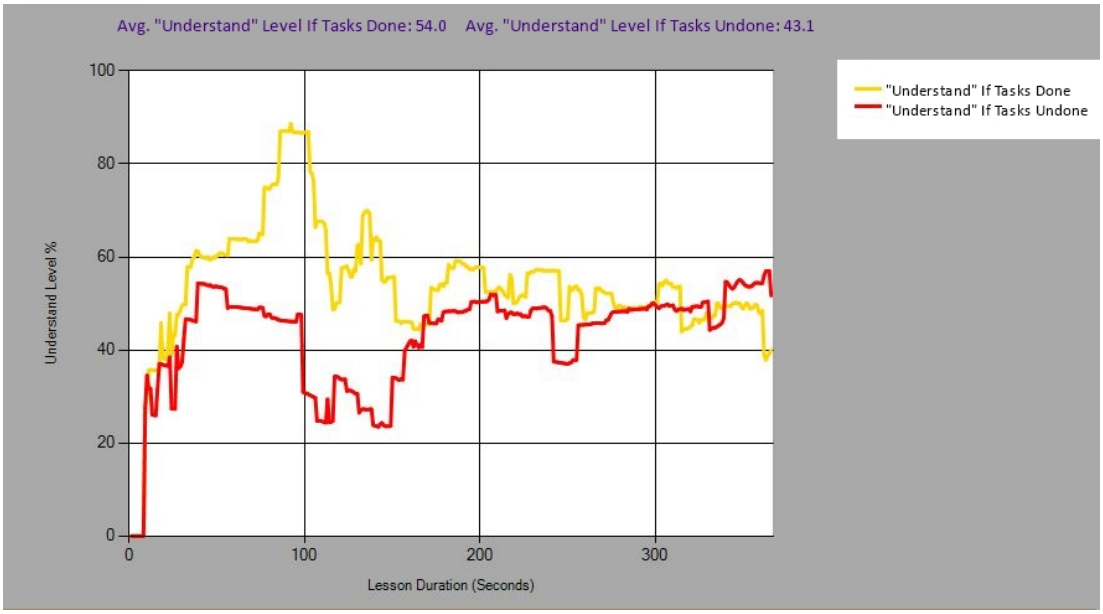
attention to lesson and more tended to understand the subject. Similarly, students think that the teacher has the control of the class more when he completes tasks as seen in Figure 4.2(a). On the other hand, we couldn't see too much difference in "respect teacher" factor in Figure 4.2(b), however still there is a minor increase if the user completes tasks. When the user completes tasks, students observe less harsh treatment as seen in Figure 4.3(a). The tension of the class is almost the same in both approaches.

The focus duration of the students for both approaches for this case are shown in Figure 4.4. The results show that every student can focus lesson for longer time when the user is guided by the tasks.

For the second part of this case study, a more problematic classroom was built up so that the users have to deal with more misbehaviors. The distribution of the characteristics of students were as follows: 2 talkative students, 2 lazy students and 2 energetic students. With this part sceneraio, we aimed to push the users to be more active and try to control more problematic students. The results of the experiment are shown in Figure 4.5, Figure 4.6, Figure 4.7.

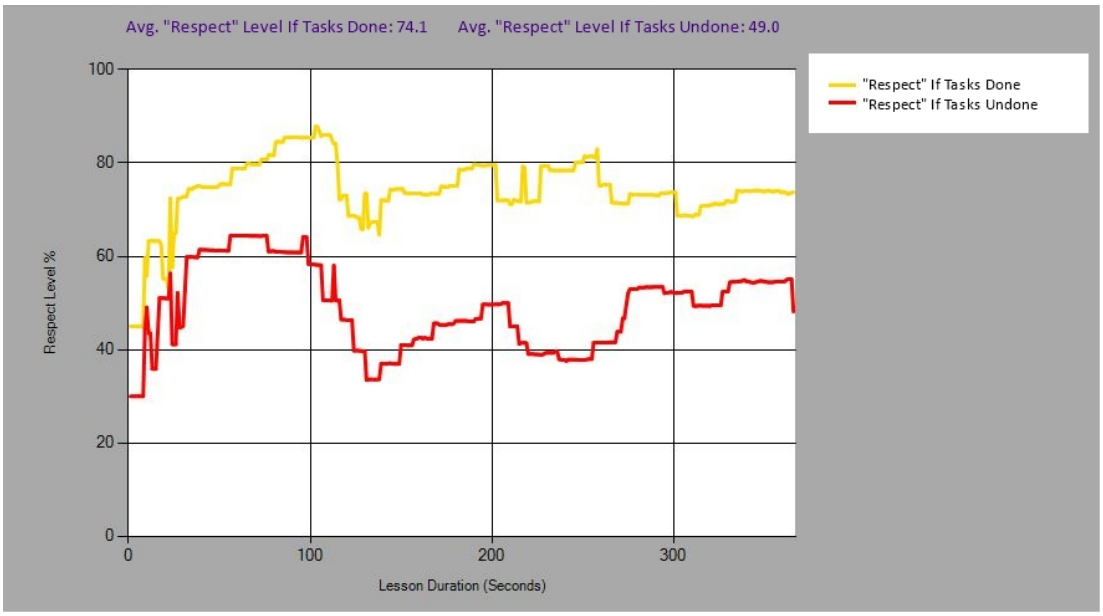


(a)

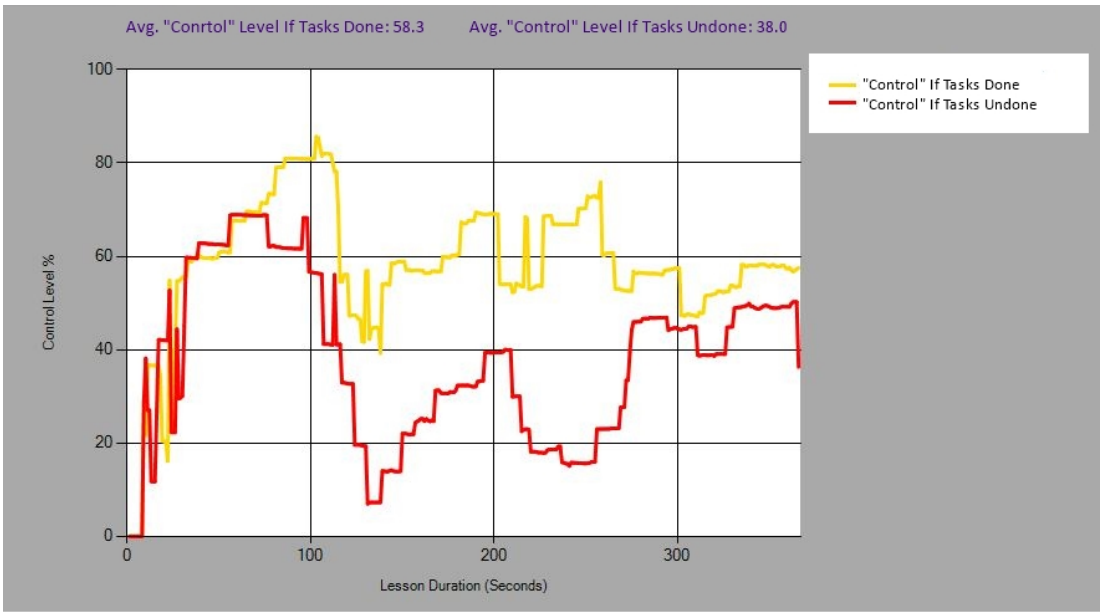


(b)

Figure 4.5: (a) Student's "Attention" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Understand Lesson" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green



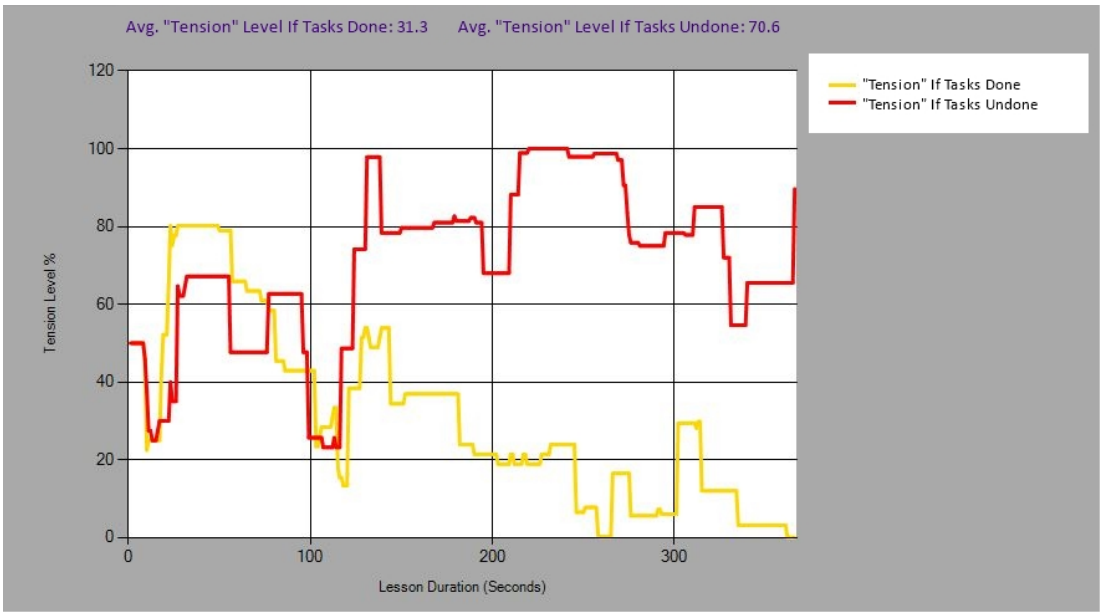
(a)



(b)

Figure 4.6: (a) Students' "Respect Teacher" Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Teacher's Control Over Class" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green

(a)



(b)

Figure 4.7: (a) Students' "Teacher's Treating Hardness" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green (b) Students' "Environment's Tension" perception Level if user tries to complete given tasks shown with blue and if user tries not to complete given tasks shown with green

The results of the second part of this case study show similarty in some values with the first part. The first inference we made is the decrement of all the mental factors of students. The reason can be explained by the characteristics of the students. The distribution of student characteristics was different with respect to the first part of the case study. It becomes difficult as the students in the classroom are more problematic. In Figure 4.5(a), there is a huge difference between the attention values. Students' attention increases magnificently if the user satisfies the needs of tasks. Likewise, students can understand the lesson better with this approach as seen in Figure 4.5(b). "Respect teacher" factor results differ from the first part. The effect of tasks can be seen better with a more problematic class. We can also say that users can control the class better, treat the students more gently and create less tension in the environment by adhering the guidence of the tasks.

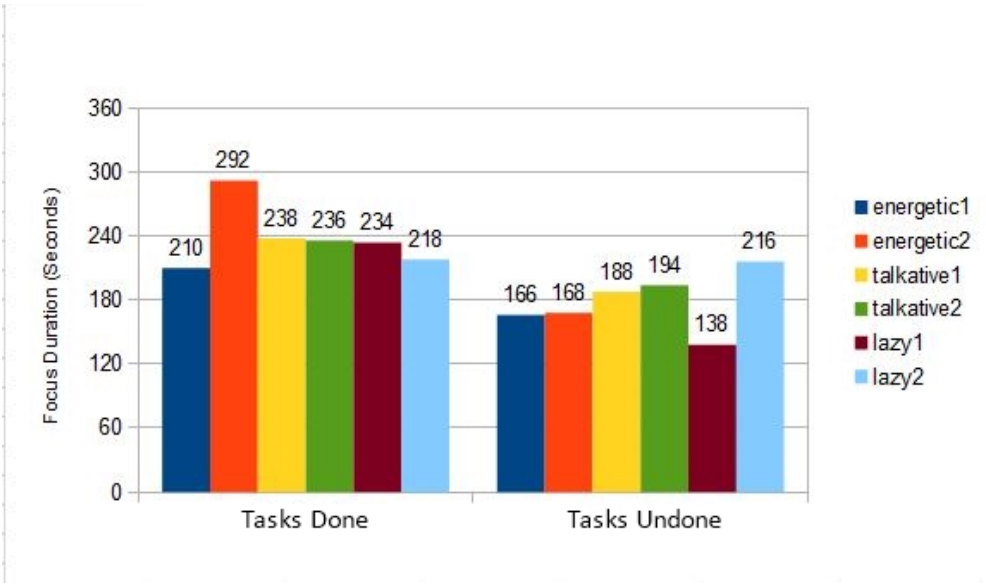


Figure 4.8: Students' focus durations for a more problematic class if user tries to complete and not complete given tasks

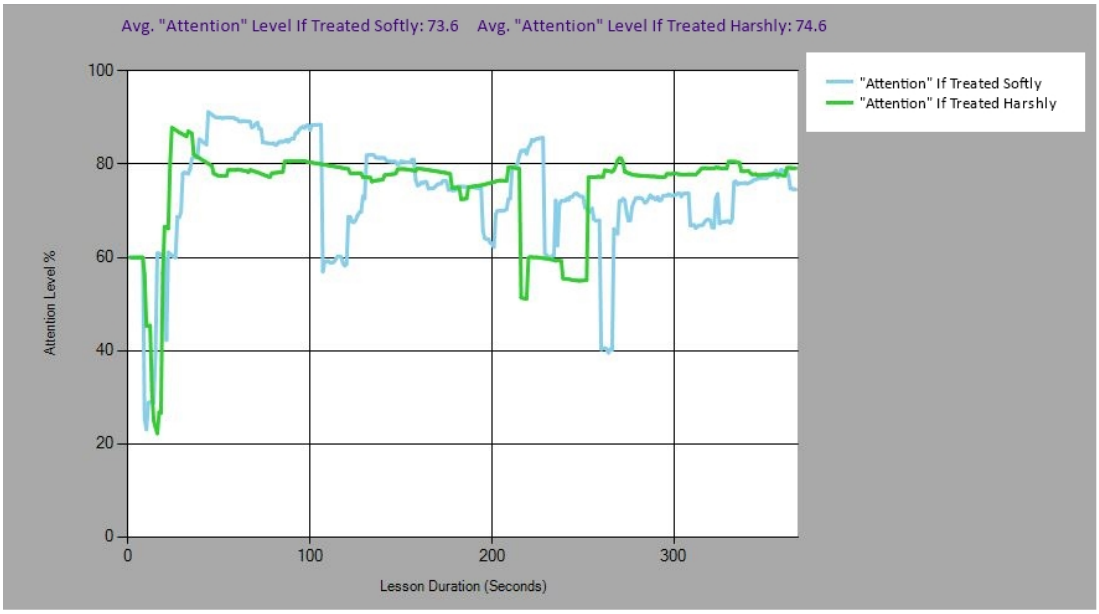
The focus duration of the students for both approaches for this experiment are shown in Figure 4.8. Similar to the first part of the case study, students can focus lesson more if the tasks are completed.

4.2 Handling A Balanced and Problematic Class - Treating Effect

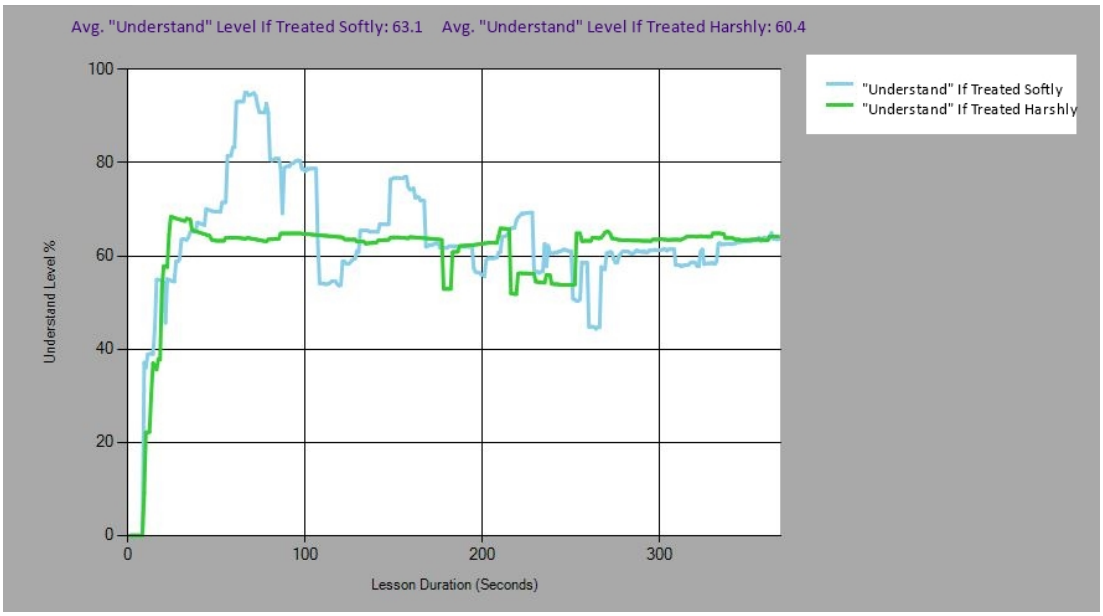
Another important approach to measure the accuracy and the realism of the proposed method is testing the way users act through the simulation. As we explained in previous section, there are some levels of treating hardness for actions to be performed. Some of these actions are harsher for the students and can effect them in different ways. For example "looking at a student"

and ”warning to focus lesson” actions have different treating hardness level where they have similar effects on students like increasing attention. The simulation was played in two different ways. In first turn, students were treated more softly in the beginning with increasing treating hardness in time if they do not respond in the expected way. For example; trainee first looks at a student and expect him to focus lesson but if the student does not focus lesson, next action will be harsher and the trainee will warn the student to focus lesson. In second turn, the trainee treated the students harshly from the beginning and performed actions that effect students' treating hardness levels more.

We built up the same classrooms as we did for the previous case study and collected data for those two different student groups. The results are shown in Figure 4.9, Figure 4.10, Figure 4.11, Figure 4.13, Figure 4.14, Figure 4.15.

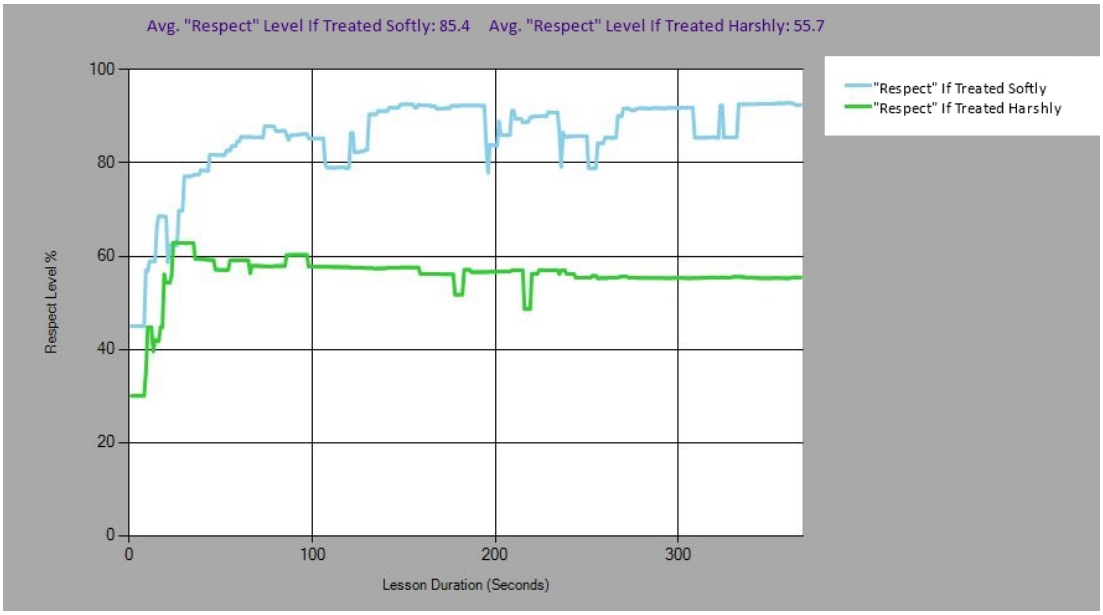


(a)

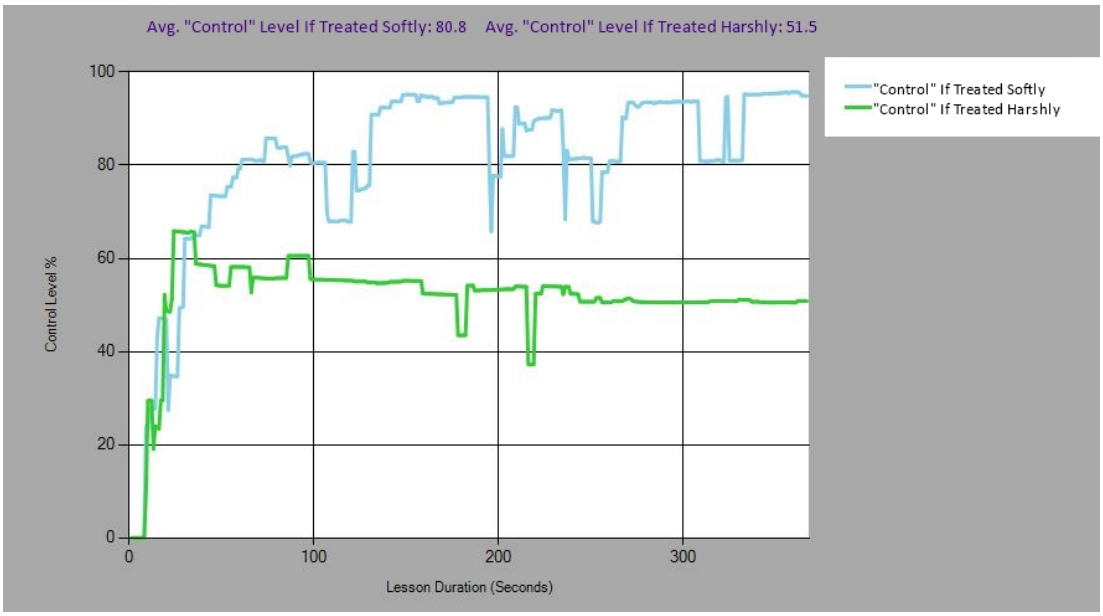


(b)

Figure 4.9: (a) Student's "Attention" Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Understand Lesson" Level if user treats students harshly shown with blue and if user treats students softly shown with green

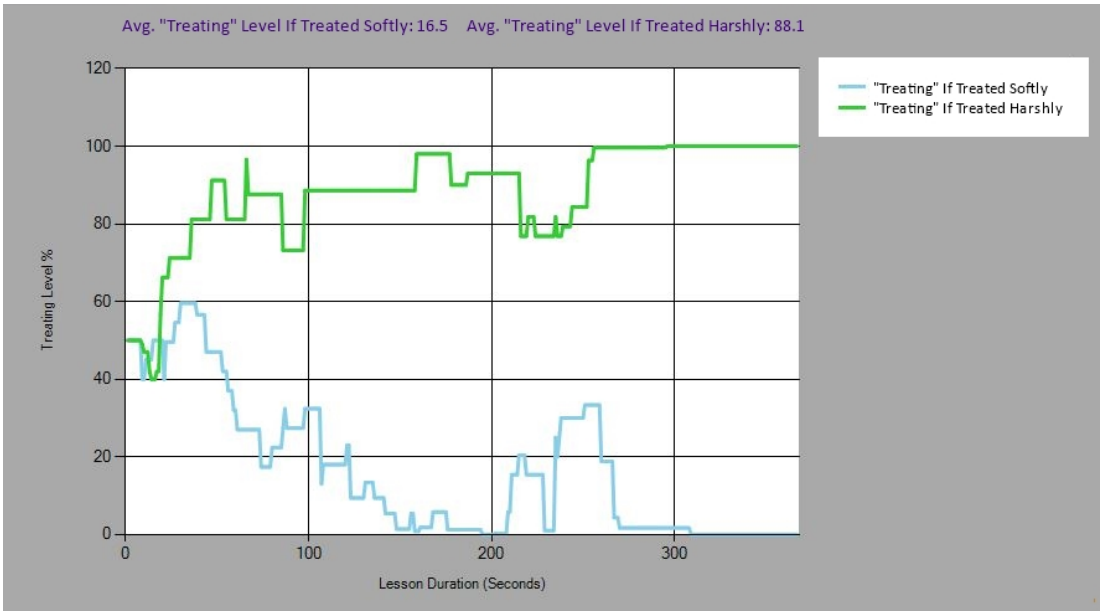


(a)

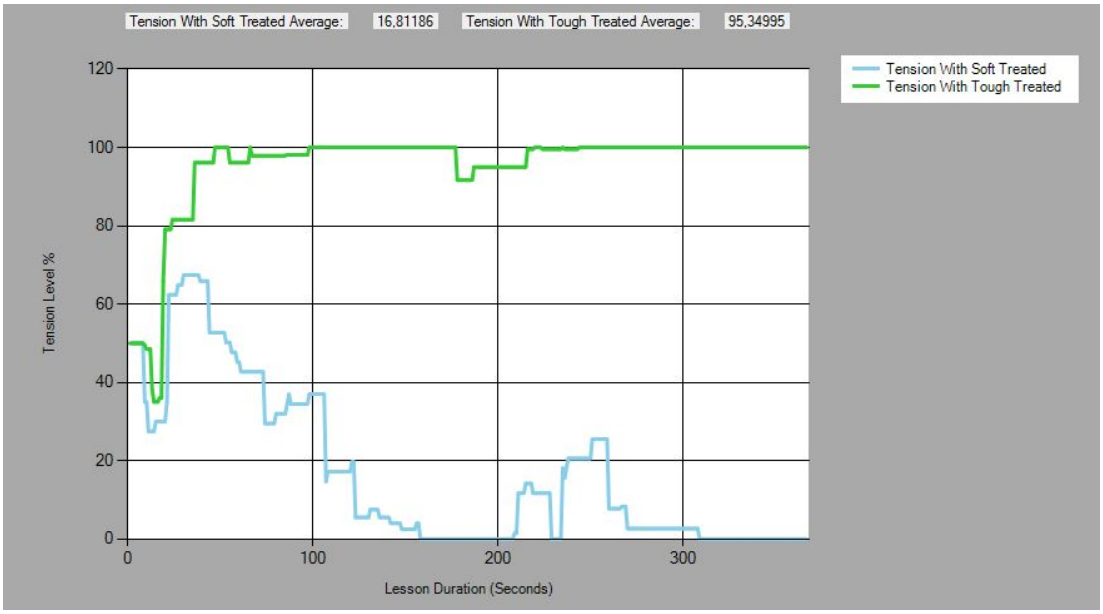


(b)

Figure 4.10: (a) Students' "Respect Teacher" Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Teacher's Control Over Class" perception Level if usertreats students harshly shown with blue and if user treats students softly shown with green



(a)



(b)

Figure 4.11: (a) Students' "Teacher's Treating Hardness" perception Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Environment's Tension" perception Level if user treats students harshly shown with blue and if user treats students softly shown with green

Data we collected gave us some interesting results that we should dwell on. First thing we noticed was the ”attention” and ”understand lesson” factor values. The results are almost the same with both approach as seen in Figure 4.9(a) and Figure 4.9(b). Students can pay attention if the user treats harshly or softly. Moreover, they can understand the lesson in either way. The difference between these two treatment approaches show up in ”respect teacher” and ”control over class” factors. The results show that the user can control better by treating the students more gently instead of treating harsly. Also they respect the trainee more if they aviod harsh actions. In addition, as expected, there is a gap between two approaches in ”treating hardness” and ”tension” mental factors. With soft treatment, there will be less tension in the environment as seen in Figure 4.11(b)

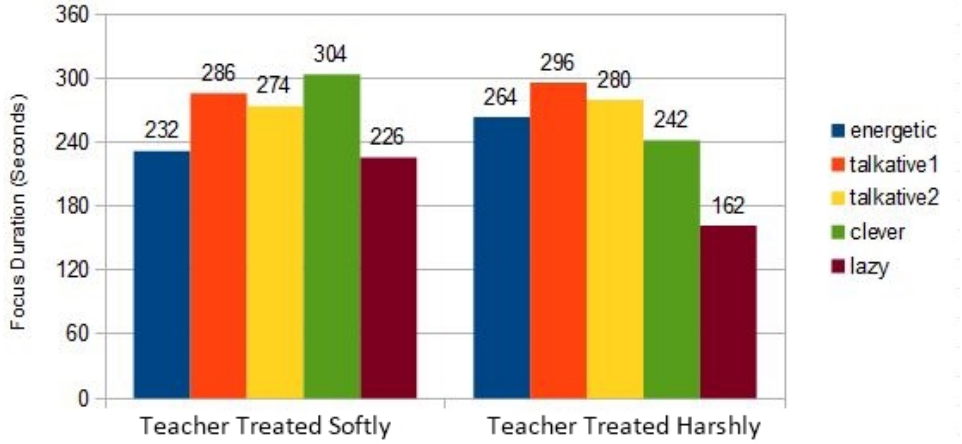
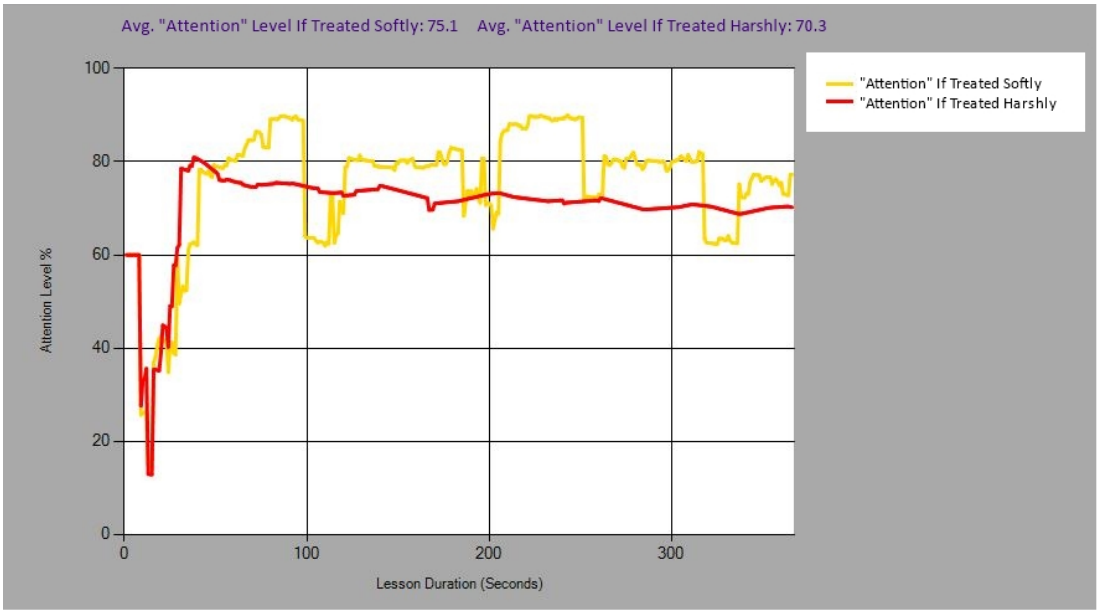
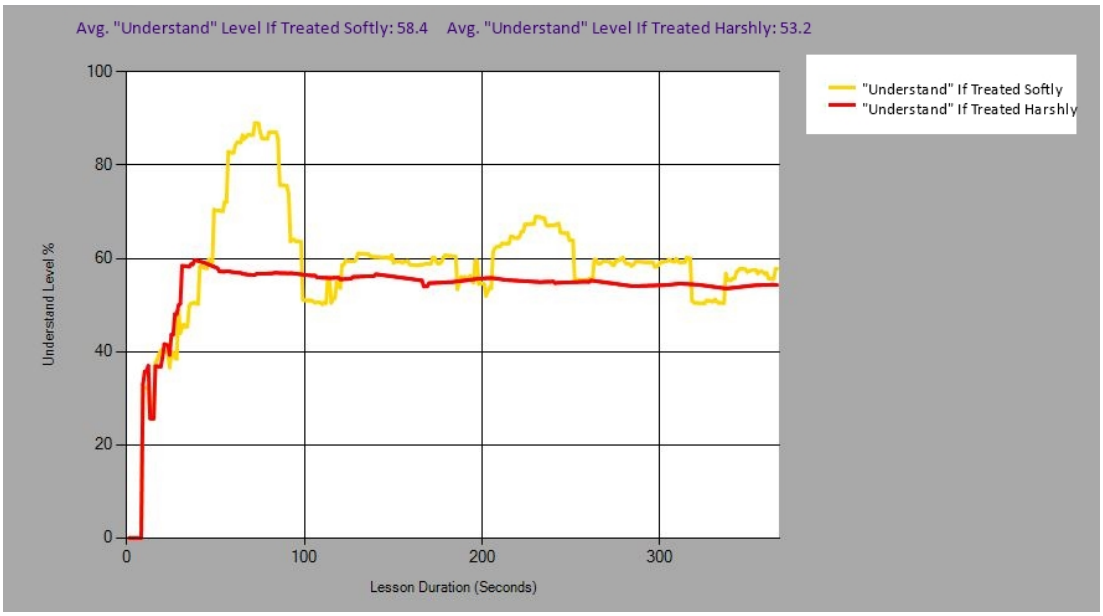


Figure 4.12: Students’ focus durations for a balanced distributed class if user treats students softly and harshly

It can be seen in Figure 4.12, some students can focus longer with soft treatment while others can focus longer with harsh treatment. We can state that treating way does not effect the focus duration or understanding lesson much but keeping the control of the class is much more difficult with harsh treatment.

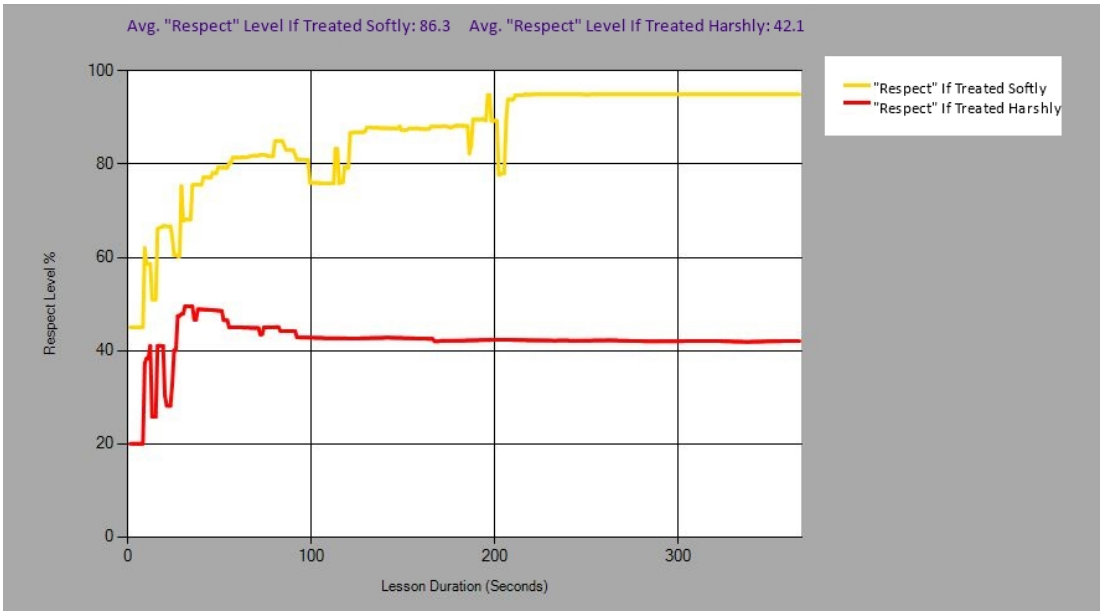


(a)



(b)

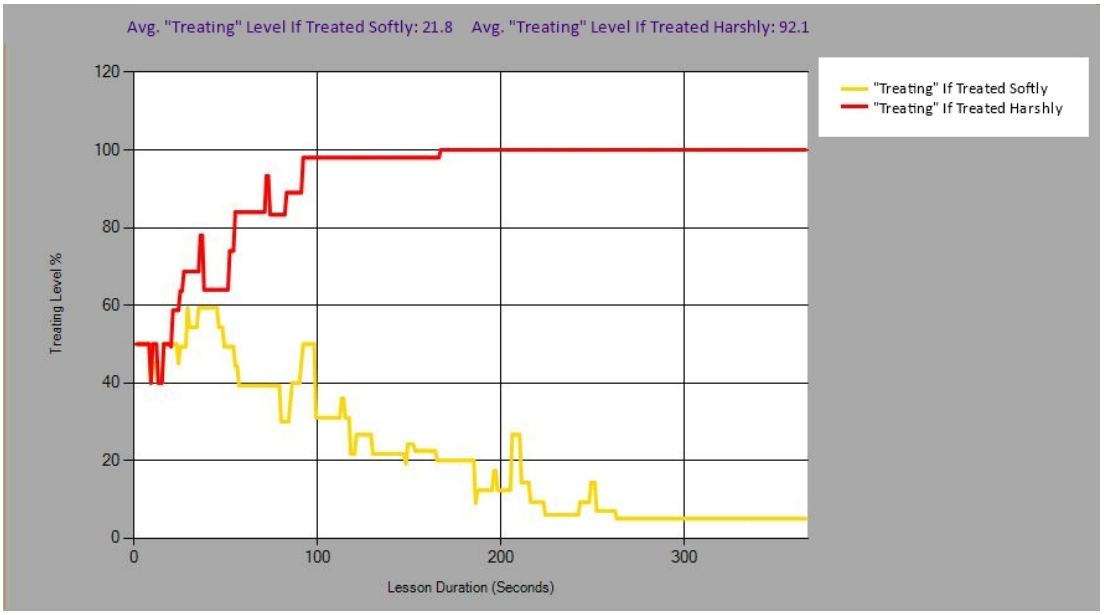
Figure 4.13: (a) Student's "Attention" Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Understand Lesson" Level if user treats students harshly shown with blue and if user treats students softly shown with green



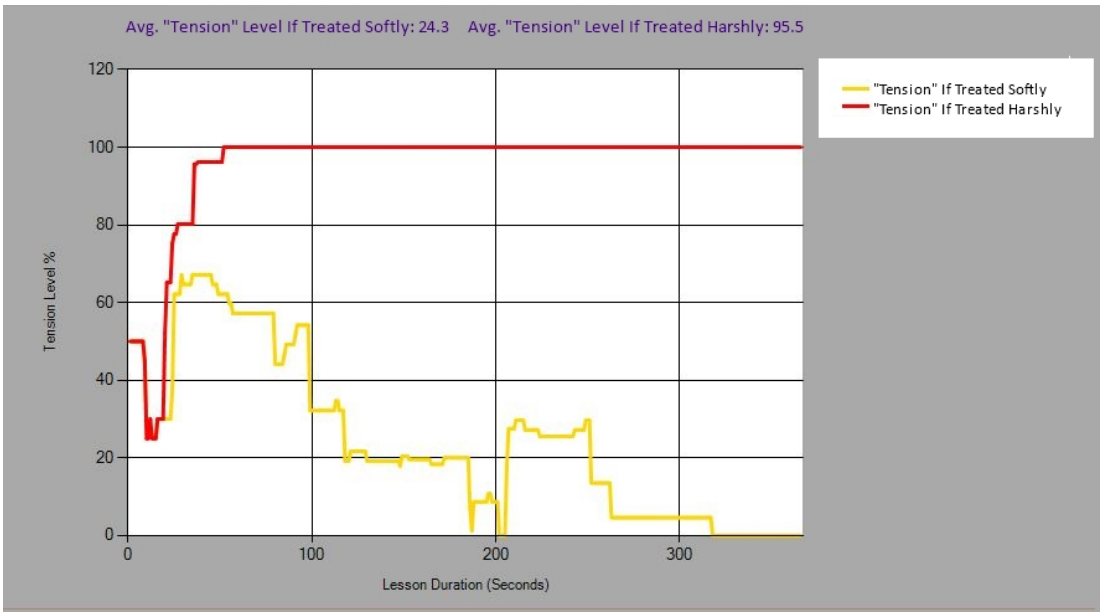
(a)

(b)

Figure 4.14: (a) Students' "Respect Teacher" Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Teacher's Control Over Class" perception Level if usertreats students harshly shown with blue and if user treats students softly shown with green



(a)



(b)

Figure 4.15: (a) Students' "Teacher's Treating Hardness" perception Level if user treats students harshly shown with blue and if user treats students softly shown with green (b) Students' "Environment's Tension" perception Level if user treats students harshly shown with blue and if user treats students softly shown with green

The results of the second part of this case study show significantly similarity with very little distinctions in ”attention” and ”understand level” mental factors. The results show that these factors are slightly better if the user performs soft and gentle actions. For other mental factors soft treatment brings the trainee advantage to perform better in the simulatiom with better control over class.

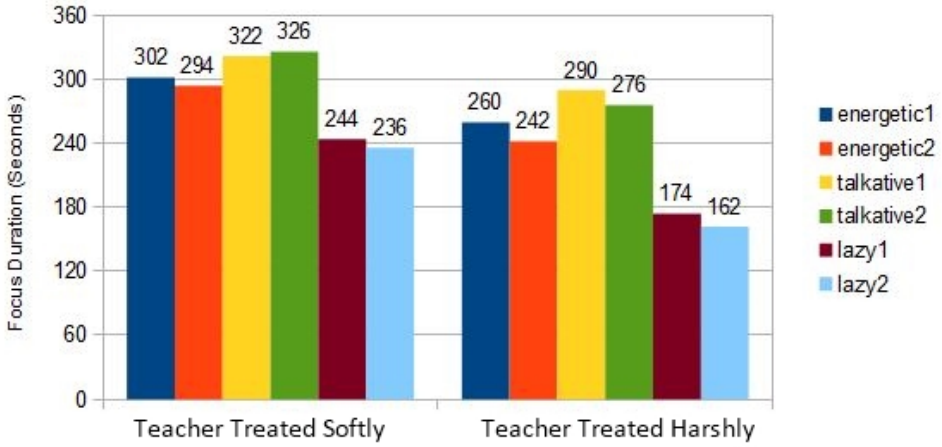


Figure 4.16: Students’ focus durations for a more problematic class if user treats students softly and harshly

The focus duration of the students are slightly better in soft treatment approach with respect to harsh treatment approach. There is a significant difference for only lazy students so that we can state that soft treatment works better with lazy students.

Both case studies showed us that the method we proposed gave rational results about the accuracy of the system. Students act and react as expected in real-life. Moreover agents suppose to behave more unpredictable with respect to FSM model. This is a benefit that BDI model brings as we mentioned in previous sections. The tasks we designed for guiding the user helps trainees to perform better in simulation according to the results we got from first case study. Also we showed that treating softly is better for controlling the class without creating tension in the environment with the second case study. The students can still focus and understand lesson with harsh treatment but we can not say that this is a preferred way to choose for trainees because of low respect and control levels. The rapid drops in the graphs indicates that there are some misbehaviors of students at that time frame like talking to each other, walking in classroom or reactions to user actions like warning to be quiet or warning to sit down.

As a conclusion, we can state that the system we designed and developed can be a robust prototype for future works. The system needs to be improved in some ways to be ready for using in real-life training sessions. These possible improvements are discussed in the next chapter.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

With this research, we have shown that an interactive virtual environment for training can be developed by using one of the most popular and human-like agent architecture in literature. With the advantages of Agent Oriented Programming and BDI model offers, the application we developed was easier to implement with respect to FSM or similar architecures because their design complexity can increase extremely as the number of behaviors increase.

We proposed a method to use BDI model to design and implement a virtual classroom. The architecture has two main components. The first one is the Multi Agent System and the second one is Visual Representation System. With this research, we built a prototype of virtual classroom which is developed using one of the most popular game engine Unity3D.

The students in the system have different characteristics and set of actions which they can choose according to their mental factors. We tried to measure the accuracy of the student behaviors by using various case studies with various student characteristics. The results showed that students' actions are rational and can be accepted as realistic.

The classroom we built needs to be improved in many ways to be more usable for training in real-life but the method we proposed presents a robust architecture and the system we developed is an open-to-improvement prototype for future works. Firstly, both the 3D object models and student models used for the virtual classroom should be more realistic and well designed as visuality is one of the most important components for the realism of virtual environments. The students should have more various animations that can be used during performing actions. For example; currently we are using only one kind of walking, sitting or laughing animations and this restriction decreases the realism of the system. Another important component that would make the virtual classroom more realistic is sounds. Currently there are no sounds implemented which limits the immersion for trainees.

In the scope of this research, we used limited kind of student actions like walking, talking, focusing or daydreaming as our system represents a prototype. However, these actions should be diversified to make the application usable in real-life. Also the conversations between the students should be extended as the infrastructure of the prototype we developed supports this

kind of extention. Moreover, students' mental factors should be changed according to the type of conversation they have.

As described in the previous sections, we offered a list of teacher actions to users that they can perform while handling the situations of the students and trying to control the class. And some of these actions are grouped according to their harshness level. These action set should be extended in the future so that the users can have more options while dealing with the situations. This can also reduce the repeatitiveness of user interactions.

Another feature that we must work on is the characteristics of the students. For now, the students have static characteristics with no background story. These characteristic should be implemented in a more generic way so that they can behave according to their dynamically changing characteristics. Also the background story for students is very important too. The family structure, social status or economical condition play a crucial role on the characteristics of students.

REFERENCES

[1] Yoav Shoham. Agent-oriented programming. *Artificial intelligence*, 60(1):51–92, 1993.

[2] Rafael H Bordini, Jomi Fred Hübner, and Michael Wooldridge. Programming multi-agent systems in AgentSpeak using Jason. *Wiley Series in Agent Technology*, 8, 2007.

[3] Nicholas R Jennings and Michael Wooldridge. Applications of intelligent agents. *Agent Technology: Foundations, Applications, and Markets*, pages 3–28, 1998.

[4] Barbara Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence*, 72(1):329–365, 1995.

[5] Lin Padgham and Michael Winikoff. *Developing intelligent agent systems: A practical guide*, volume 13. Wiley, Chichester, 2005.

[6] Ralf Schleiffer. An intelligent agent model. *European Journal of Operational Research*, 166(3):666–693, 2005.

[7] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995.

[8] Michael Wooldridge. An introduction to multiagent systems. pages 1–42, 2008.

[9] Katia P Sycara. Multiagent systems. *AI magazine*, 19(2):79, 1998.

[10] Leon S Sterling and Kuldar Taveter. *The art of agent-oriented modeling*. The MIT Press, Cambridge, MA, London, England, 2009.

[11] Nicholas R Jennings, Jose Manuel Corera, and Iñaki Laresgoiti. Developing Industrial Multi-Agent Systems. In *ICMAS*, pages 423–430, 1995.

[12] Pattie Maes et al. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

[13] Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, volume 31, page 40. London, UK, 1996.

[14] Nicholas R. Jennings, Peyman Faratin, MJ Johnson, Timothy J. Norman, P O'brien, and ME Wiegand. Agent-based business process management. *International Journal of Cooperative Information Systems*, 5(02n03):105–130, 1996.

[15] B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett, and A. Seiver. Distributing Intelligence within an Individual. Technical report, Stanford, CA, USA, 1988.

[16] Fabio Bellifemine, Giovanni Caire, and Dominic Greenwood. Developing Multi-Agent Systems with JADE. pages 3–27, 2007.

[17] Michael E Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, 1999.

[18] Michael E Bratman, David J Israel, and Martha E Pollack. Plans and Resource-Bounded Practical Reasoning. *Computational intelligence*, 4(3):349–355, 1988.

[19] Daniel C Dennett. The Intentional Stance. pages 1–37, 1989.

[20] Michael E Bratman. What Is Intention. *Intentions in communication*, pages 15–32, 1990.

[21] Philip R Cohen and Hector J Levesque. Intention Is Choice With Commitment. *Artificial intelligence*, 42(2):213–261, 1990.

[22] Jaeho Lee, Marcus J Huber, Edmund H Durfee, and Patrick G Kenny. UM-PRS: An Implementation Of The Procedural Reasoning System For Multirobot Applications. In *NASA CONFERENCE PUBLICATION*, pages 842–842. Citeseer, 1994.

[23] John Langshaw Austin. How To Do Things With Words. pages 1–53, 1975.

[24] John R Searle. Expression and meaning: Studies in the theory of speech acts. pages 1–50, 1985.

[25] Michal Cap, Annerieke Heuvelink, Karel Van Den Bosch, and Willem Van Doesburg. Using Agent Technology To Build A Real-World Training Application. In *Agents for games and simulations II*, pages 132–147. Springer, 2011.

[26] Emma Norling and Liz Sonenberg. Creating Interactive Characters With BDI Agents. In *Proceedings of the Australian Workshop on Interactive Entertainment (IE'04)*, pages 69–76, 2004.

[27] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. JACK Intelligent Agents-Summary Of An Agent Infrastructure. In *5th International conference on autonomous agents*, 2001.

[28] Michael Winikoff. JACK Intelligent Agents: An Industrial Strength Platform. In *Multi-Agent Programming*, pages 175–193. Springer, 2005.

[29] Fabio Bellifemine, Federico Bergenti, Giovanni Caire, and Agostino Poggi. JADE A Java Agent Development Framework. In *Multi-Agent Programming*, pages 125–147. Springer, 2005.

[30] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex: A BDI Reasoning Engine. In *Multi-agent programming*, pages 149–174. Springer, 2005.

[31] Marcus J Huber. JAM: A BDI-theoretic mobile agent architecture. In *Proceedings of the third annual conference on Autonomous Agents*, pages 236–243. ACM, 1999.

[32] Lee J., Huber M. J., Durfee E. H., and Kenny P. G. UM-PRS: An Implementation Of The Procedural Reasoning System For Multirobot Applications. In *Conference on Intelligent Robotics in Field, Factory, Service, and Space (CiRFFSS'94)*, pages 842–849, 1994.

[33] Rafael H Bordini, Jomi F Hübner, and Renata Vieira. Jason And The Golden Fleece Of Agent-Oriented Programming. In *Multi-agent programming*, pages 3–37. Springer, 2005.

[34] Anand S Rao. AgentSpeak (L): BDI Agents Speak Out In A Logical Computable Language. In *Agents Breaking Away*, pages 42–55. Springer, 1996.

[35] Gupta, Satyandra K and Anand, Davinder K and Brough, J and Schwartz, Maxim and Kavetsky, R. Training in Virtual Environments. *A Safe, cost-effective, and engaging approach to training. University of Maryland*, 2008.

[36] Soh K Ong and Andrew YC Nee. Virtual Reality and Augmented Reality Applications in Manufacturing. pages 1–11, 2004.

[37] Sarah Nichols, Clovissa Haldane, and John R Wilson. Measurement of presence and its consequences in virtual environments. *International Journal of Human-Computer Studies*, 52(3):471–491, 2000.

[38] Patrick Kenny, Arno Hartholt, Jonathan Gratch, William Swartout, David Traum, Stacy Marsella, and Diane Piepol. Building interactive virtual humans for training environments. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, number 1. NTSA, 2007.

[39] S Coquillart, A Steed, and G Welch. V3S, a virtual environment for risk management training. In *JVRC11: Joint Virtual Reality Conference of EGVE - EuroVR*, pages 95–102, 2011.

[40] J. Brooke. SUS: A "quick and dirty" usability scale. *Usability evaluation in industry*, pages 189–194, 1996.

[41] Cédric Buche, Ronan Querrec, Pierre De Loor, and Pierre Chevaillier. MASCARET: pedagogical multi-agents systems for virtual environment for training. In *Cyberworlds, 2003. Proceedings. 2003 International Conference on*, pages 423–430. IEEE, 2003.

[42] Sanam Dehghan. A Serious Game To Learn Student Teacher How To Control Students' Misbehaviours. Master's thesis, METU, 2013.

[43] Jere H Brophy and Mary McCaslin. Teachers' Reports of How They Perceive and Cope with Problem Students. *The Elementary School Journal*, 8:3–68, 1992.

[44] Arran Bartish and Charles Thevathayan. BDI Agents For Game Development. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2*, pages 668–669. 2002.

[45] Unity3d. http://www.unity3d.com. Accessed: 2013-08-02.

[46] Mono project. http://www.mono-project.com/What_is_Mono. Accessed: 2013-08-02.