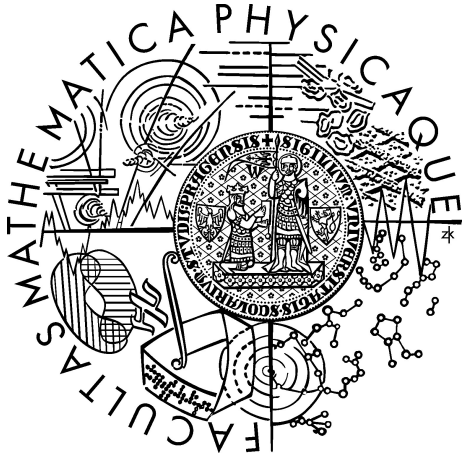
Charles University
Faculty of Mathematics and Physics

DOCTORAL THESIS



Michal Bída

Artificial Emotions in Virtual Storytelling

Department of Software and Computer Science Education

Supervisor of the doctoral thesis: Cyril Brom
Study programme: Computer Science
Specialization: Theoretical Computer Science

Prague 2017

I would like to dedicate this thesis to all storytellers, writers and dreamers.

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Umělé emoce ve virtuálním vyprávění

Autor: Michal Bída

Katedra: Katedra software a výuky informatiky, Matematicko-fyzikální fakulta, Univerzita Karlova

Vedoucí disertační práce: Mgr. Cyril Brom, PhD., Katedra software a výuky informatiky, Matematicko-fyzikální fakulta, Univerzita Karlova

Abstrakt: Práce se zabývá problematikou interaktivního digitálního vyprávění (IDS) ze dvou perspektiv. První část práce studuje přístupy k vytváření IDS systémů a specifikuje jaké chování musí splňovat agenti pohybující se v takových prostředích. První část je zakončena návhrem minimalistické architektury modulované afekty, která umožňuje vytváření agentů fungujících ve středně velkých IDS systémech. Druhá část práce představuje dva IDS systémy, které byly vytvořeny pomocí této architektury. Tyto systémy jsou dále použity pro výzkum v oblasti automatické analýzy stavových prostorů generovaných IDS systémy. Je představena obecná metodologie analýzy, která je implementována a otestována na třech doménách různých IDS systémů.

Klíčová slova: interaktivní digitální vyprávění, agentí architektura, uvěřitelnost, emoce, analýza dramatu

Title: Artificial Emotions in Virtual Storytelling

Author: Michal Bída

Department: Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University

Supervisor: MSc. Cyril Brom, PhD., Department of Software and Computer Science Education, Faculty of Mathematics and Physics, Charles University

Abstract: This thesis is concerned with Interactive Digital Storytelling (IDS) from two perspectives. Firstly, it analyses the problem of the development of IDS systems and defines behavioral requirements for the agents in those systems. The first part is concluded with a proposal of a minimalistic affect-modulated architecture that can be used to develop agents for medium-sized IDS systems. Secondly, two working IDS systems built on top of the architecture are introduced and used in the following part of the thesis that researches the problem of an automatic analysis of story spaces generated by IDS systems. A general methodology of analysis is introduced, implemented and tested on the domains of three working IDS systems.

Keywords: interactive digital storytelling, agent architecture, believability, emotions, drama analysis

Contents

Introduction **4**

1 Theoretical Background **9**
1.1 Interactive Digital Storytelling . . . 9
1.1.1 The *Wicked Problem* . . . 10
1.1.2 Theoretical Perspective . . . 11
1.1.3 Modern Approaches . . . 14
1.1.4 Summary . . . 16
1.2 Believable human-like Characters . . . 16
1.2.1 Believability . . . 16
1.3 Affect Simulation . . . 18
1.3.1 Theories of Emotions . . . 19
1.4 Agent Architectures . . . 21
1.4.1 The Warhorse AI System . . . 21
1.4.2 FAtiMA . . . 22
1.4.3 A Behavior Language (ABL) . . . 23
1.4.4 Summary . . . 24
1.5 Overview of Current IDS Systems and Games . . . 25
1.5.1 Firewatch (PC game) . . . 25
1.5.2 Façade . . . 26
1.5.3 FearNOT! . . . 28
1.5.4 The Prom Week . . . 30
1.5.5 Other Notable Examples . . . 31
1.6 Summary . . . 33

2 Emohawk Agent Architecture (EWA) **35**
2.1 Problem Analysis . . . 37
2.2 Related Work . . . 38
2.2.1 General Approaches . . . 38
2.2.2 Selected Agent Architecture and Languages . . . 41
2.3 The Proposal . . . 43
2.4 Basic Overview . . . 43
2.4.1 Representing the Agent Behavior . . . 45
2.4.2 Summary . . . 48
2.5 Decision Making System . . . 49
2.5.1 Reactive Factories . . . 50
2.5.2 Story Controller (Drama Manager) . . . 51
2.6 Reasoning in EWA . . . 52
2.7 Emotion Model . . . 52
2.7.1 OCC Theory . . . 53
2.7.2 ALMA Overview . . . 54
2.7.3 EWA and ALMA Integration . . . 56
2.7.4 Other Computational Models of Emotions . . . 58
2.8 Navigation Manager . . . 59
2.8.1 Motivation and Related Work . . . 59

2.8.2 Steerings . . . 60
2.8.3 Steerings and EWA integration . . . 61
2.9 Summary . . . 62

3 Applications **63**
3.1 Pogamut . . . 63
3.2 StoryFactory . . . 64
3.3 SimDate3D Level One (SD One) . . . 67
3.3.1 Game Overview . . . 67
3.3.2 Implementation . . . 69
3.3.3 Discussion . . . 76
3.4 SimDate3D Level Two (SD Two) . . . 77
3.4.1 Game Overview . . . 77
3.4.2 Implementation . . . 81
3.4.3 Discussion . . . 84
3.5 Debugging Support . . . 86
3.5.1 Run-time Debugging Tool . . . 86
3.5.2 Graphical Tools For Log Analysis . . . 87
3.6 Lessons Learned . . . 90

4 Drama Analysis **93**
4.1 Related Work . . . 93
4.1.1 Theoretical Approach . . . 94
4.1.2 Technical Work . . . 96
4.2 Problem Analysis . . . 98
4.3 The Proposed Solution . . . 100
4.3.1 Abstracting The Stories . . . 100
4.3.2 Clustering . . . 103
4.3.3 Story Space Exploration . . . 105
4.3.4 Discussion . . . 106
4.4 Implementation . . . 107
4.4.1 Action Strings Representation . . . 107
4.4.2 Sub-scenes Extraction . . . 108
4.4.3 Tension Curve Extraction . . . 109
4.4.4 String Metrics Implementation . . . 109
4.4.5 Artificial Player Exploration . . . 109
4.4.6 Clustering . . . 111
4.5 Evaluation . . . 111
4.5.1 Overview . . . 112
4.5.2 Experiment 0 . . . 115
4.5.3 Experiment 1 . . . 120
4.5.4 Experiment 2 . . . 123
4.5.5 Experiment 3 . . . 124
4.5.6 Summary . . . 126
4.6 Discussion . . . 127

Conclusion **129**

Bibliography **132**

List of Figures **147**

List of Tables **149**

Acronyms **150**

Appendices **152**

A ALMA Emotions and Moods **153**

B Debugging Tool Windows Overview **155**

C SimDate3D Level One Game Controls **158**

D SimDate3D Level Two Game Controls **159**

E Example Stories **161**
E.1 SimDate3D Domain 161
E.2 MOSS Domain 161

F Content of the DVD **163**

Introduction

We live in fascinating times. One could call it a renaissance of computers. Computing power increases every year [Moore, 1965], our data storage is getting bigger and the networks are getting faster. Moreover, in the last twenty years there has been a couple of important breakthroughs in the field of artificial intelligence. In 1997, the world chess champion Gary Kasparov was beaten by IBM's computer Deep Blue [Campbell et al., 2002]. In 2011, IBM super computer Watson won the game of Jeopardy in United States beating two of the best human champions [Ferrucci et al., 2010]. In 2016, Google AlphaGo won against Lee Se-dol 4-1 in the game of Go [Silver et al., 2016]. Other remarkable achievements are the recent advances in deep neural networks that can now solve problems in many interesting areas of computer science. There are self driving cars from Google [Hunter, 2015] and for example DeepMind's solver for some of the Atari games [Mnih et al., 2015]. The great thing about the latter is that the solver uses only the images of the game screen to learn how to play. Despite the fact there is no supervision the deep neural networks are able to learn some of these games so well that they are able to beat experienced human players [Mnih et al., 2015]. What comes next? The author believes it is an optimal solution of no limit heads up poker [Moravčík et al., 2017].

It is not hard to assume that these advances will lead to some changes – in our industries and in our society. Computers are penetrating basically every part of the human endeavor, be it medicine, bioinformatics, physics or even philosophy[1]. Writers are now using text editors with spell checking that helps them write and there are endeavors to automate writing even further [Roemmele and Gordon, 2015]. The world of media is also changing rapidly. Gone are the times when the computer games were just for children. In recent years the industry of computer games development established itself as a major media for digital entertainment among the film and music industry[2]. So how are the computer games different?

Watching a movie or listening to music can lead to many positive emotions in humans. However, people usually cannot participate in this process directly[3]. A person is just a passive receiver of the beautiful music or an intriguing movie storyline. With computer games this is different. Computer games introduced an interactive style of entertainment. In computer games a person is a part of the story or at least they are made to believe it. And most of the time that person has the power to change the course of the game, to influence the story and the characters in the game and perhaps to cause a very different game ending. This poses some major challenges on Artificial Intelligence (AI) and algorithms in the game. Note that computer games are not deserted places (usually) but are living worlds inhabited by virtual characters. These are referred to as either Non-player Characters (NPCs) or Intelligent Virtual Agents (IVAs)[4]. These often

[1]The philosophers recognize the use of computer models as a valid method of studying the world around us (e.g. by making models of the universe).

[2]See for example report [Entertainment Software Association, 2016].

[3]Kinoautomat [Činčera et al., 1967] was the first interactive movie presented in the Czechoslovak Pavilion at Expo 67 in Montreal.

[4]In this thesis the terms Intelligent Virtual Agent (IVA), Non-player Character (NPC), agent and character will be used interchangeably.

possess a human like body and they struggle to convince the player that they are living human like characters with a *believable* behavior. More often than not the player interaction with these characters forms and creates the story. The forms of this interaction may differ – in first person shooter for example, the player is expected just to fight and destroy enemies and not to talk to them. On the other hand in social simulation games such as The Sims [Electronic Arts, 2014] – the whole game is about keeping your characters happy by taking care of them and satisfying their needs such as hunger, thirst and the need to socialize. This then creates a completely different set of requirements on the AI of these characters (e.g. requiring also some form of emotion simulation). Moreover, a new sub-genre of computer games is emerging – a virtual drama. This sub-genre is interesting from two perspectives. On one hand a new type of games perceived by some as a form of interactive movies is being created (e.g. the Walking Dead series to name one [Telltale Games, 2012]). These games are – much like movies – based heavily on human emotions, but now the human is a part of the story, so the emotional impact can be much higher. On the other hand, virtual drama is interesting from the scientific perspective as well. There are conferences dedicated solely to this topic, e.g. the ICIDS[5] conference. Why is that? We already have virtual drama games out there, so what is there to research?

Virtual drama focuses on the plot of the game, on the dramatic experience of the player and on the power of the player to influence the plot. In this genre the player often finds herself in complicated, dramatic situations she can somehow influence by performing some actions or communicating with other characters – either by some pre-defined actions or even by natural language. The major difference here is the explicit focus on the plot – the "fun" in virtual drama should come out from the player interacting with the plot. In these games, it is much harder to program a "robust" AI[6]. The AI in these games does more than just control the characters as it needs to assure that the outcome of the interaction between the player and the story will be: a) meaningful and b) entertaining. This is by no means an easy task and there is a whole subfield called Interactive Digital Storytelling (IDS) that is currently working on a solution to this problem which poses as their main research question. Let us delve into this a bit more from a computer game development perspective.

The need to maintain a coherent plot is a requirement that is common for all computer games. Usually, this is approached in as simple way as possible – the story is linear with some detours that in fact lead back to the main story path and do not change the outcome of the story (but the user may experience the feeling that they do). This solution has multiple advantages for the game developers – a) the game developer has a good control over the story, so the game designers and writers can make sure that the story will be interesting and entertaining and b) creating a linear story arc is much less expensive than creating a content for a narrative with a high branching factor.

The thing is, when the developers are investing in the creation of the game content, they would like the player to see the context in full measure when playing the game. Stories with high branching factor that do not let the player see half

[5]International Conference on Interactive Digital Storytelling http://icids.org/

[6]Note that there is a thin line between virtual drama and computer game, most of the systems indeed fall into both categories.

of the game content in her first run are going against that. The linear stories with detours have become a kind of a standard way for the computer game story authoring[7] – most of the games out there right now succumb to this model, virtual drama games inclusive (this will be discussed in depth in Chapter 1). However, some of the game developers and researchers see this as an opportunity to take virtual drama even further by granting the player more freedom to shape the game.

In this regard, we should briefly mention sandbox games. Sandbox games provide the player with high degree of freedom enabling the player to influence the game environment to a great extent. In some of these games – such as Minecraft [Mojang, 2011] – user can build structures, mine ores, shape mountains, create and destroy castles or program a working calculator inside the game world. However, in most of these sandbox games the story itself is not that important[8] – some of these games even lack a story or a game ending completely – player can play infinitely until she is bored.

How is that connected to virtual drama? In virtual drama the holy grail goal is to create a sandbox game, but this time, not with regard to the game world but with regard to the story itself! So now we would like the player to influence the story in any way she pleases and every time the game adapts so the story stays interesting, entertaining and believable. So what if the player lets the hunter in the Little Red Riding Hood story get lost in the woods? Well maybe the girl's mother would think about visiting the granny herself and would fend off the wolf saving both the granny and the girl. But that would require: a) the characters in the game to react intelligently and meaningfully to user actions – they need to be believable, b) the story itself reacting to the player making changes to it by reshaping itself and/or the characters and the game world and c) the game to make sure that the eventual outcome of a) and b) is entertaining for the user.

Thesis Goals

This thesis will tackle questions such as: How to simulate a believable character in dramatic situations from everyday life? What are the key features of believable behavior of NPCs? What is a good story in IDS system? And how could we measure it algorithmically?

In particular, this thesis researches two main problems of IDS:

a) How to control the characters in the IDS system so that their actions are believable but at the same time keep their behavior clearly specified and extensible.
b) How to "dramatically" evaluate stories generated by such a system.

That gives us a set of requirements on what needs to be done as a part of this thesis. To be able to research b), it is advantageous to have a IDS system that contains characters interacting with each other and hence producing stories. To

[7]This is demonstrated for example on a case study of Mass Effect 2 game [Bizzocchi and Tanenbaum, 2012].

[8]There are notable examples of sandbox games that are "generating" also simple forms of narrative, e.g. Red Dead Redemption [Rockstar Games, 2010]. These stories are usually sketchy and do not provide a full fledged dramatic experience for the player.

be able to implement this we need to resolve a). So, how are we going to specify NPC behavior? What architecture are we going to use and in what environment will our virtual characters live?

The above lead to the specification of following goals that were accomplished in this thesis:

G1. Design a minimalistic agent architecture that aims to create a *believable* behavior of the NPCs in computer game-like environments.
G2. Find and use a 3D virtual environment that would allow us to experiment with NPCs engaging in social situations.
G3. Implement an IDS system on top of the agent architecture that will contain NPCs living in the 3D environment and that will produce stories that can be analyzed.
G4. Design and implement a methodology for estimating the quality of virtual drama produced by a game or an IDS system.

In particular, the following pieces were created as a part of the research endeavor of this thesis:

- Emohawk Agent Architecture (EWA) – a minimalistic agent architecture with integrated emotion model and navigation manager allowing for plausible navigation of characters in the virtual environment.
- SimDate3D Level One (SD One) and SimDate3D Level Two (SD Two) – two narrative games in 3D environment using EWA for controlling the NPCs featuring two characters (SD One) and three (SD Two) characters in social situations.
- A methodology for estimating the quality of virtual drama that is based on abstracting the important story features that can be further analyzed, e.g. by methods of clustering.
- Connection of Unreal Engine 2 (UE2) environment to the Pogamut platform, broadening the range of virtual environments that are available in the Pogamut platform.
- StoryFactory – a tool enabling creation of short movies in 3D game environment that can be used to mock up dramatic situations in 3D environment with IVAs and should promote the education in the field of IVA and IDS.

Thesis Structure

The rest of the thesis will be structured as follows. Chapter 1 starts by introducing the IDS field and its main research questions. In particular, it shows that IDS is an instance of the *wicked problem* and define the *narrative paradox*. This Chapter shall also conclude that the characters are an essential building block of every story and continue with introduction to the problematics of *believable* IVAs. To be able to design *believable* IVAs there is a need to tackle the problems of the *affect simulation* and state-of-the-art approaches to the field. This leads to the current state-of-the-art agent architectures that are used to solve various instances of the *wicked problem*. The last section of Chapter 1 presents an overview of current state-of-the-art IDS systems such as computer games or academic simulations.

Chapter 2 introduces a minimalistic solution to the problem of believable simulations of a dramatic behavior in 3D virtual environments. In particular, it

shows that the EWA architecture is a *good enough* solution for a *wicked problem* of specification of the *medium-sized drama* – while simple, it contains all the concepts necessary to design believable behavior in this context. The Chapter 2 presents the high-level overview of the architecture with focus on key aspects that are required for this particular problem, those are: a decision making component, an affect simulation component, a navigation manager allowing for plausible movement of characters in 3D environment, a story controller component managing the consistency of the plot, a component allowing for the debugging of agent's behavior and the integration of all of these components. As far as the author knows, the EWA architecture is the only architecture that is specifically integrating such components in the context of a medium-sized drama.

In the following Chapter 3, the three software pieces that were created as part of the research endeavor of this thesis are introduced. These are the SD One and SD Two games that were implemented by the author of this thesis and the StoryFactory tool that was implemented by a team of people working with the author as a team leader. The SD One and SD Two are implemented on top of EWA and are used to showcase that EWA can be used to solve the domains of the SD One and SD Two games in an effective fashion. The implementation of SD One and SD Two makes possible to move forward to the last part of the thesis.

The last Chapter 4 is tackling the problematic of semi-automatic drama analyses of stories produced by some IDS system. A novel methodology is introduced and analyzed and the results are discussed. The methodology is based on a general story features extraction such as a dramatic *tension curve* that exploits the simulated emotions of the characters in virtual drama, *action strings* and *sub-scenes*. It is shown that the methodology can be applied to three distinct domains of the IDS systems – the SD One, the SD Two and the IDS system Moral Storytelling System (MOSS). The thesis contributions are summarized in the Conclusion.

1. Theoretical Background

This chapter provides the theoretical background for the reader of this thesis. It starts with the IDS field followed by definition of *believable* IVAs. For this, it delves a little bit into the field of affect simulation. Afterwards, state-of-the-art agent architectures used in the IDS and computer games are introduced and summarized. Lastly, several examples of working IDS systems are presented.

1.1 Interactive Digital Storytelling

Imagine a kind of a story sandbox. The author does not write down the exact story in a linear fashion as we are used to in movies or books. Instead, she will only create the basic settings – when and where the story happens, who the characters are and what their personalities and goals are. One plugs this into some IDS system and the result will be a real story according to that setting. For example, imagine that instead of writing down the Little Red Riding Hood story[1], one would just say:

"We have a granny, mother, Red, wolf and a hunter. Red needs to bring some food to the granny. Wolf wants to eat Red and the hunter wants to preserve order in the woods."

With these information the system would be clever enough to play this out not just as a boring story where Red brings the food to her granny and the story ends but with all the common quirks and turns as in the real Little Red Riding Hood story. And let us go ever further. Imagine a player being able to interact with this story somehow – either by playing one of the characters herself or by steering the characters in a way of telling the wolf to go to the granny's house now. It is the holy grail of the IDS research field to be able to effectively develop a system working in that way. Or one can imagine a Star Trek's holodeck – a sci-fi concept where the crew of a space ship spends free time in a virtual reality where they take part in interactive stories playing, e.g. a Sherlock Holmes character.

IDS is a multi-disciplinary field integrating AI, media studies, psychology, computer graphics and narratology. From the computer science perspective IDS is concerned with computer game design, AI and human computer interaction (HCI) in general. HCI researches user interface designs and various other kinds of human computer interaction. From broader perspective the reader can in fact understand IDS as a sub-field of HCI.

IDS is also a form of digital entertainment with the emphasis on the interactivity – the user needs to be able to interact with the system somehow and this interaction should ideally produce a change in the story that will be meaningful from the user perspective. Are the branching pre-scripted stories in computer games a form of IDS? They definitely are but from the author perspective

[1] Little Red Riding Hood story seems to be popular in the IDS community and was modelled as an IDS system for example in [Si et al., 2010].

they would fall into the *lower agency*[2] form of IDS, the higher level would be allowing the player to do *almost* anything, not restricting her to pre-scripted story branches. That is actually one of the grand challenges of IDS field – the *narrative paradox* [Aylett, 2000] – that is closely related with the player *agency* [Tanenbaum and Tanenbaum, 2010].

The *narrative paradox* states that the more freedom is given to the player the bigger the impact she can have on the story, making it harder to produce coherent stories and preserve the authorial intent and vice versa.

How to give such freedom to a player of a computer game? With scripted branching narrative approach used commonly in games that would mean a lot of variants that could be the outcome of such actions and this is infeasible to create effectively – due to economical and time reasons[3]. So how much freedom should the player be allowed in the game? If the player is allowed to lock Red in her house, then how do we create an interesting narrative out of that? We could let her escape through the window or spawn her younger twin, but the *narrative paradox* pattern emerges even in this simple example. The good thing is that the user *usually* does not want a complete freedom per se but wants to pursue actions that are meaningful in the context of the story (unless she wants to break the system).

1.1.1 The *Wicked Problem*

So, let's say we have a scripted story and a user that is willing to cooperate. In that case, our IDS system still needs to make sure that either (1) the story structure is not dependent on what the user does (approach used in many computer games) or (2) that there is some mechanism that steers the narrative towards the intended authorial goal when the user does something outside the plot domain.

Both (1) and (2) essentially limit the player agency but allows IDS system to be built in the first place. Now, the IDS field has been researched for years, but we still do not have the the ultimate solution – the holodeck. Why is this happening? The unfortunate thing here is that the design and implementation of a working IDS system seems to be an instance of the *wicked problem* as defined by [Rittel and Webber, 1973] (originally for the planning problem). The *wicked problem* – applied to IDS – has the following features (cited from [Mateas and Stern, 2005a] based on [Rittel and Webber, 1973]):

- *There is no definitive statement of a wicked problem.* While a problem statement for IDS system might be "create an interactive version of the Little Red Riding Hood story" this is by no means a well defined problem. Often the problem that is solved by IDS system is understood only after the system is built.

- *Wicked problems have no stopping rule.* This is true for IDS systems. Due to their nature there is always a space for improvement – e.g. by authoring

[2]The term *agency* refers to how much freedom the player has in the storytelling system or in a computer game. The higher *agency* the more the player can influence the scenario by her actions. See [Tanenbaum and Tanenbaum, 2010] for more information about player agency.

[3]To develop interactive drama Façade the authors spent three man years to author all the story content in their IDS system [Mateas and Stern, 2005b].

more story branches or giving the player more actions to choose from or agency etc.

- *Solutions to wicked problems are not correct/incorrect but rather better/worse or good enough or not good enough.* There is no holy grail solution for IDS at the moment. There are only approaches that tend to be better at e.g. preserving the story but limiting the player agency and vice versa usually at the cost of extensive authoring of the content.

- *Every wicked problem is essentially unique.* As in every computer game and IDS system is unique. There are many approaches to IDS each focused on a particular problem but often it is hard to generalize common rules.

- *There is no immediate nor ultimate test of a solution to a wicked problem.* Solutions provided by IDS systems can change the player expectations changing the original problem in the first place, e.g. by creating a game with an interesting story the players might want to interact with and be able to further change the story.

The *wicked problem* notion is useful as it points out the difficulties that pop out when designing a working IDS system. In this thesis, the *wicked problem* of IDS is not solved. However, several useful concepts that can help implementing IDS system for medium-sized drama has been pinpointed. Next, the theoretical perspective of storytelling and drama is considered.

1.1.2 Theoretical Perspective

From a theoretical perspective, IDS is concerned with stories and drama – topic that was studied from the times of ancient Greece for example by Aristotle [Halliwell, 1987] or Plato [Plato et al., 1992].

Plato makes a distinction between *mimesis* and *diegesis*. *Mimetic* storytelling has a form of playing out the story in front of the audience as in a movie or a theatre. Here, the story is conveyed directly by the actor's actions and speech on stage. On the other hand, the *diegetic* storytelling is an indirect one. Here the author uses the narrative to convey the story to the audience addressing the reader directly as when reading a book or when listening to a narrator in a movie. These two approaches are often used together in the modern media as the lines between story and discourse are thin.

Both Aristotle's and Plato's point of views are still relevant in IDS today. For example Freytag's Pyramid [Freytag and MacEwan, 1968] can be viewed as an extension to Aristotle theory of drama (Fig. 1.1). The notion of Freytag's pyramid later crystallized into the term *dramatic tension curve* – the author tracked the first use of this term to [Fónagy, 2001] – a concept that is used later in this thesis. Plato is relevant especially when considering the character-oriented or story-oriented forms of IDS systems ("strong autonomy" vs "strong story") which is to be discussed in the next Section.

For more in depth analysis of Aristotle and Plato *diegenis* and *mimesis* principles see [Kirby, 1991]. For even broader introduction to IDS theoretical background, see [Spierling et al., 2010], [Louchart, 2007] or [Swartjes, 2010]. A more practical perspective on current IDS systems can be found in [Rank et al., 2014].

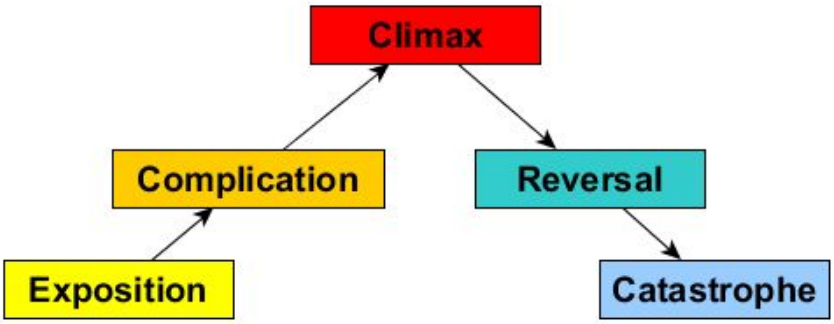


Figure 1.1: Freytag's pyramid. In his work [Freytag and MacEwan, 1968], Freytag layed out a 5-act dramatic structure that has later become known as Freytag's pyramid. This theory proposes five stages of drama that are perceived as universal. First stage is the exposition – the moment for introduction of the settings and the characters. Second stage is the complication, where the actions dramatizing the plot appear and the tension in the drama increases. This then escalates to climax (or conflict or peripetie) where the tension is the highest. After the conflict is resolved the drama moves towards reversal where the outcome of the conflict is portrayed, followed up by the last stage catastrophe, where the closure is provided – usually the downfall of the hero in a tragedy.

From more recent theoretical works that are relevant to IDS, [Propp, 2010] should be mentioned. Vladimir Propp's *Morphology of the Folktale* is a scientific analysis of Russian folktales. Propp analyzed hundreds of Russian folktales and concluded that all of the plots shared a basic structural template. He also identified *character function* as the basic building block of the tale and defined seven character roles: *a hero*, *a villain*, *a mediator* or *a dispatcher*, *a donor*, *a helper*, *a princess* and *a false hero*. These characters carry out one of the 31 functions that can be divided according to [Fairclough, 2004] into five general groups: *preparation*, *complication*, *transference*, *struggle* and *return*. For more in depth analysis of *Morphology of the Folktale* and its relevance to IDS see [Fairclough, 2004] or [Gervás, 2013]. Propp's work was used as a basis for a number of the IDS systems, e.g. [Spierling et al., 2002], [Fairclough, 2004], [Tomaszewski and Binsted, 2007] and [Gervás, 2013].

Beads on a String

[Chatman, 1978] introduces a concept of kernels and satellites of the story where kernels represent the *narrative moments that give rise to cruxes in the direction taken by events* and the satellites represent some minor plot events. From a computer games perspective, this is very similar to *beads on a string* pattern – a common pattern used in games to model narratives [Harrigan and Wardrip-Fruin, 2007].

This model of an interactive narrative works in a way that the story is linear but there are certain beads in the story, where the player can influence the story but only inside this one bead. When the player leaves this bead, the story continues as planned and nothing they did inside this one bead could change how the story will unfold. Why is the beads on a string approach used? One of the

reasons is that if the branching narrative approach would be used, an exponential explosion of branches that would need to be authored for every decision of the user would make the creation of even a small IDS system highly impractical [Stern, 2008], [Crawford, 2012]. The beads on a string approach mitigates this problem by constraining the amount of the content that needs to be authored while maintaining the "feeling" of interactivity for the player [Bizzocchi and Tanenbaum, 2012].

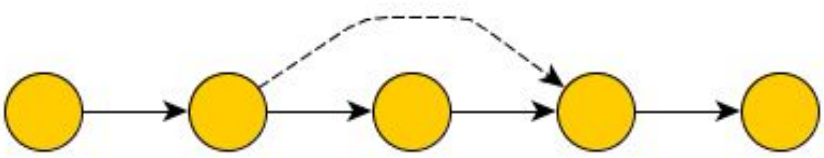


Figure 1.2: Beads on a String. Example of a linear story where the player influence is limited to a particular bead. However, to improve the perception of interactivity, player actions inside one bead can influence future beads as shown by the dashed arrow.

To take the perception of interactivity even further, the *beads on a string* approach is used in a way that while the player cannot change the outcome of the main plot (she needs to follow the beads), she can influence what the future beads will look like by conducting actions in the current bead (Fig. 1.2) – see [Bizzocchi and Tanenbaum, 2012] for a detailed analysis of the interactive narrative in the game of Mass Effect 2. That can be done on multiple levels – on a dialog level, the player can be mean to one of the characters at the beginning of the story. Then, at the end of the story, this character might behave differently than it would if the player was nice to him/her. On the other hand, when the player destroys a vase in one of the rooms during the story, when the player re-enters the same room in the future, the vase will still be destroyed.

[McKee, 1997] states that the smallest element in a scene is a *beat*.

A beat is an exchange of behaviour in action/reaction. "Beat by beat" these changing behaviours shape the turning of a scene.[4]

The notion of a *dramatic beat* as a smallest unit of drama that pushes the story forward was adopted by [Mateas, 2002] when creating IDS system Façade. Also the *beats* resemble the story beads discussed above.

Next, a concept of the *emergent narrative* is defined.

Emergent Narrative and Narrative Paradox

Emergent narrative [Aylett, 1999] is an IDS concept where the narrative *emerges* from the interaction of the human and the IDS system, e.g. a computer game. It is just like in a normal human life, where the "story" also *emerges* from the person's interaction with the world and other humans in it. The *emergent narrative* in an IDS system should be entertaining and purposeful. But this then leads to

[4]Cited from [McKee, 1997], p. 37.

the previously mentioned *narrative paradox* [Aylett, 2000]. The more freedom is given to the user in the virtual environment the harder it is to keep the plot coherent and satisfy authorial intent within this environment. There are two main approaches to solving the *narrative paradox* that are discussed next.

1.1.3 Modern Approaches

There are two main conceptual approaches to IDS systems. The *character-based* approach ("strong autonomy") where the narrative *emerges* from the interaction between player and NPCs, and *plot-based* approach ("strong story") where usually a *drama manager* component is present controlling the story and making sure it stays coherent [Mateas and Stern, 2002], [Louchart, 2007], [Swartjes, 2010], [Spierling et al., 2010], [Rank et al., 2014]. Note that the distinction between these two approaches is not strict. The "strong autonomy" systems can contain at least some rudimentary ways how of steering the characters plotwise and the "strong story" system often contains NPCs that can be autonomous to some extent. Next, advantages and disadvantages of these approaches are to be discussed.

Plot-based approach ("strong story")

The plot-based approach has the obvious advantage of the authors being able to control the plot considering author goals such as the achievement of the ideal dramatic *tension curve*, making sure the events in the story will occur at appropriate times or that the important events will take place in the story in the first place. This comes at a price though. The more coherent the plot the less agency for the user.

The plot-based approach is usually implemented by a *drama manager*. It is the responsibility of the *drama manager* to steer the characters in a way the story is preserved. The "power" of the drama manager to control the characters and steer the plot in the system usually defines whether the system falls more into the "strong story" or the "strong autonomy" category.

Example of a strong story approach is the IN-TALE (Interactive Narrative Tacit Adaptive Leader Experience) system [Riedl and Stern, 2006]. In IN-TALE a drama manager (Automated Story Director) determines the next action the character performs. The story is represented by a branching narrative in the form of partially-ordered plans with *casual links* that are used to mark all casual relationships between the steps in the plan. A plan is not considered complete until all the preconditions of all actions are satisfied by a casual link. This enables to force the planner to adhere to author-defined goals.

Another examples of story-based approaches are Façade [Mateas and Stern, 2003] that are to be discussed later or IDTension [Szilas, 2003], MEXICA [Pérez y Pérez, 2007] and [Fairclough, 2004].

Character-based approach ("strong autonomy")

In general, the advantages and disadvantages of the "strong autonomy" approach contrast with the one of the "strong story". Here, the user has more freedom to influence the story as the characters are driven by their goals and the *drama*

manager – if present at all – has a lower influence over the actions of the characters. The character-based approach benefits from the fact that the developer has now a better control over the actions of a particular character. It is now possible to make the character's personality consistent, to enhance the character with plausible emotions and in general, make the character appear life like and entertaining. The drawback is – having all of the above – the question of how to assure that the story that will *emerge* from the interaction between the user and characters will be good enough? And this is just a tip of the iceberg – the dramatic *tension curve*, pacing, critical story events taking place, synchronization of these autonomous characters – all of this becomes much more complex giving the NPCs bigger autonomy.

Example of "strong autonomy" system is the Thespian [Si and Marsella, 2010]. Thespian implements a two layers system where at the character level the system assures the character's behavior is rich and well motivated and at the plot level, there is a director agent that proactively directs the characters in a way the plot design is reached. If a conflict between character goals and the story goals is detected the character goal will get priority to keep the character's behavior consistent.

Another example is FearNot! [Aylett et al., 2007] that takes the emergent narrative approach and is discussed later in this chapter. For more examples of "strong autnomy" systems see [Swartjes, 2010].

Use of Classical Planning

There is one more important state-of-the-art approach to IDS field – use of the classical planning [Riedl et al., 2011]. The basic idea here is that the story will be represented as a planning domain consisting of actions and their preconditions and effects and some goal state. Going back to the Little Red Riding Hood example, the desired goal state is that the wolf is dead and Red, her granny and hunter are alive. The available actions are then for example: "Red goes to her granny", "wolf goes to granny", "wolf eats Red", "wolf eats granny", "hunter kills wolf" etc. Here, it is very important to design the domain in the right way. The narrative "hunter kills wolf" indeed satisfy the goal state, however, this story is not as interesting as the original Little Red Riding Hood story.

The major problem here is that the planning approach always tries to minimize the number of actions to reach the goal state. But in IDS the goal is not to have the shortest narrative, but to have an interesting narrative. So, how that can be captured in the planning domain? One of the solution is to add more preconditions or constraints to actions, e.g. [Porteous et al., 2010]. Another approach is to use a partial ordering of the actions saying that some actions need to appear before others or it could be taken even further by adding some fixed action to the middle of the plan at the start, forcing the planner to plan around this. With this approach the player is constrained with the actions that are available to him in this planning story domain but there is far more freedom than with the traditional scripted approach and the story is reacting robustly to players' actions by leveraging the planning algorithm. [Cavazza et al., 2007] and [Porteous et al., 2010] are good examples of working systems leveraging the planning approach to model interactive narrative.

1.1.4 Summary

The Section presented theoretical foundations to the problematics of IDS. A broader theoretical introduction to the field can be found in [Spierling et al., 2010] and shorter overview in [Rank et al., 2014].

Next, an introduction to the problematics of *believable* human-like characters is to be presented. This is closely relevant to the character-centric approach where the goal is to have consistent, simulated characters that are perceived as *believable*.

1.2 Believable human-like Characters

This chapter is concerned with the problematics of believable behavior of IVAs or NPCs in games and IDS systems. Before starting, let us review the definition of IVAs according to the main conference concerned with this theme.

Intelligent Virtual Agents (IVAs) are interactive characters that exhibit human-like qualities and communicate with humans or with each other using natural human modalities such as facial expressions, speech and gesture. They are capable of real-time perception, cognition and action that allows them to participate in dynamic social environments.[5]

1.2.1 Believability

How do the IDS and IVAs go together? As probably every writer knows, interesting characters are one of the pillars of a good story. In games and IDS systems, this means that the behavior of these characters in the context of the story and in the context of the action the player has done should be interesting and *believable*. What does it mean to be believable? This means the characters should react "appropriately" in the given context. In other words, their behavior should make sense to the player. The player should be able to make sense of what happened and understand why the character did it in the context of the situation.

What are the requirements on IVAs to be able to achieve the given goals? [Loyall, 1997] wrote entire PhD thesis on this topic. Now to briefly summarize the gist of his work. For IVAs and NPCs to be *believable* it is desirable that the player perceives them as individuals. For this to happen, Loyall states that there is a need to design IVAs in a way they have their own *personality*. Two individual characters might react differently to the same events in the environment. *Personality* is strongly related to character's *emotions* and *emotion perception*. It is not hard to assume that to make our characters *believable* we might need to simulate affective behavior somehow. Now, IVAs and NPCs should have their own *goals* they want to pursue as it is natural for human beings to have personal agendas as well. Moreover, IVAs need to be *social* – they need to be able to react to each other and engage in social situations. And ideally, in the end the behavior produced by IVAs should be *consistent* regarding all of the previous. Simply put, IVAs should be perceived as virtual human beings.

[5] Quoted from description of the Intelligent Virtual Agents 2013 conference, URL: http://www.cstr.ed.ac.uk/iva2013/ [11.11.2016]

Now, how to design characters that produce behavior like the one described? And what does it look like? Let's step back a bit and take a look at how one can design and create scenes with IVAs in general.

Michael was heading home. After 10-hours workday he was completely wasted. He walked slowly with his head down through the park. Suddenly, someone shouted "Michael!". He turned around and saw a girl. He couldn't believe his eyes! It was Magali. They've met in France a couple of years ago. What is she doing here? He ran to her, hugged her and kissed her on her cheeks. Suddenly, he was happy and full of energy. "What are you doing here?" he asked. They continued to walk through the park together immersed in their conversation.

Let us imagine Michael and Magali are IVAs. What is there to do in order to have them play out a scene sketched above? First of all their virtual bodies need to be designed and created. That does not mean just textures and shape, but also the movement and the animations. Creating plausible looking animations for characters in 3D environments is a complex task that needs to be solved. For the best results a motion capture technology is used where the real actors play out all the animations the game or IDS system requires and all is then transformed to the 3D model of the character using advanced 3D editors. In our scene the characters are hugging and kissing each other – these animations need to be captured. For the best realistic outcome, one would want to have facial expressions as well. Facial expressions are important for emotion expressions [Ekman and Friesen, 1971]. To be able to express emotions in a meaningful way is an important pillar of an IDS system – more about this in a while.

The characters are also walking – in the park alone and then together. Being able to walk together in a 3D environment in a believable way with plausible obstacle avoidance is a task that needs to be handled as well, see [Popelová et al., 2011]. The emotions of the characters are changing. The changes need to be apparent and there has to be an emotional reaction. This is one of the key elements of a believable behavior. If the characters will not react with appropriate emotions in the given context they will just look odd. So how to solve all of these problems at once?

One of the solutions that is used in order to circumvent at least some of the hard problems of the task above is scripting. The scene above could be created quite easily in StoryFactory [Bída et al., 2011b] or some game engine such as Unity [Unity Technologies, 2005] or UnrealEngine4 [Epic Games, 2012] by scripting the characters to do exactly what is outlined in the scene. The reader has probably seen a couple of 3D animated movies and the results of this can be truly amazing. The characters are fun, interesting, emotional and beautiful. But from the IDS perspective the scripting is not good. The characters should demonstrate such behavior on their own. As mentioned in the chapter above, a setting needs to be created and the characters should be let to play out the story themselves with the benefit of the player being able to step in the scene and influence it somehow. So, how can the control mechanisms of such characters be designed? And is this even possible? The answer to this question is – to some extent – yes. Let's summarize the basic requirements on the AI of the characters to allow for *believability* and also for everything mentioned above. Our IVAs need to have:

- **Goals.** Goals are natural and common representations of a directed behavior in the AI systems. This is the basic requirement on the characters to be able to solve various tasks with goal oriented behavior, e.g. to get back home from their workplace or to greet their long lost friend.
- **Emotions.** The characters need to be able to express their emotions and even more than that. It makes sense for believable characters not just to express emotions but to have some kind of internal emotion representation that would influence their behavior. E.g. a sad character might want to isolate himself more, while a happy character should react more positively to events.
- **Movement.** This goes beyond the basic path-finding solving graph algorithms such as A*[6]. The characters need to be able to move together in various kinds of social situations such as when a couple walks together or when the characters are arguing. When the requirement of the system is to be able to produce this behavior on a whim this then requires the system to solve this task in an intelligent way.
- **Communication.** The characters need to be able to exchange information in a way they understand each other. Note that using natural language presents many challenges on its own [Roth and Vermeulen, 2013]. That is why in games and IDS systems, there are usually custom abstractions defined – such as list of acts the characters can understand – circumventing the problems of the Natural Language Understanding (NLU).

These requirements are complex and ad hoc implementations will fall apart rather quickly – especially when the complexity of the system increases. This is the reason why many agent architectures are being created, enabling the developer to design and maintain behaviors fulfilling all of the above in a meaningful way. The Emohawk Agent Architecture (EWA) proposed and implemented as a part of this thesis integrates systematic solutions to all of these problems in the domain of medium-sized IDS systems. Before venturing deeper, an introduction of two additional topics important for IVAs, IDS and EWA is needed. These are the topics of *affect simulation* and *agent architectures*.

1.3 Affect Simulation

Emotions are an important factor in human life. They influence decision making, remembering and memory, goals, attention, bodies and basically everything that happens inside people [Minsky, 2007]. Emotions define people. And not only their personality, emotions are essential for intelligence as well [Damasio, 2003]. So is human intelligence built on top of the emotions? Author of the thesis believes that is not the right perspective. Instead, the author thinks that emotions and intelligence are entangled systems where you cannot separate one from the other at all. Or to put it in another words, there is no fine line between emotions and intelligence.

Basic emotions are present in the simplest of animals and their survival de-

[6]Description of A* algorithm can be found in chapter 3.5.2 in [Russell and Norvig, 2010].

pends on the proper functioning of that system. Emotions are simply – old[7]. What happened in humans is that our brains not only used the evolutionary older emotions such as anger and fear but the emotions in humans evolved into a broad range of complex cognitive emotions such as pride, resentment, contempt and more.

Grasping a multifaceted topic such as emotions is not an easy task. There are many theories of emotions [Kleinginna and Kleinginna, 1981]. More often than not, these theories differ even in that what they perceive as emotion. This does not necessarily mean that some of these theories are wrong – seeing the problematic of emotions from e.g. the evolutionary perspective [Plutchik, 2003] might lead to different outcomes than for example looking at the emotions from the social perspective, focusing of facial expressions of humans [Ekman and Friesen, 1971]. And both of these theories can be used practically in computer systems to work with emotions. To give an example – Ekman's theory is used in visual recognition classifiers that are able to recognize the emotion expressions in humans face by analyzing images taken, e.g. from a webcam. When it comes to AI, IVAs and NPCs, there are two main areas where emotions could be leveraged (cited from [Bída et al., 2007] page 1[8]):

- **To increase performance of agents living in dynamic environments.** There is no doubt that emotions play an important part in adaptive decision making and are helping organisms to survive in complex and hostile environments [Plutchik, 2003]. The idea here is to use the emotions as a mechanism for adaptation for IVAs increasing their intelligence.

- **To improve human computer interaction.** Firstly, if computer systems were endowed with the ability to understand and work with human emotions, it would increase performance of humans interacting with such systems. Secondly, implementing affect simulation in NPCs and IVAs could help to produce more believable life like behavior of these characters, increasing aesthetic experience of games and IDS systems.

From the IDS perspective, the latter use case is more relevant. The emotions could be leveraged to make the behavior of IVAs *believable*. From the psychological perspective, there are many approaches to the problematics of emotions. Next section will summarize this field from the author's perspective.

1.3.1 Theories of Emotions

In this Section, psychological theories of emotions and approaches suitable for the use case of affect simulation in IVAs are to be discussed.

Basic emotions

This group of emotion theories often views the emotions as a kind of universal programs that have a behavior or a group of behaviors associated with them,

[7]Damasio [Damasio, 2010] believes that the fundamental mechanisms from which the emotions emerged later were already present in the first one-cell organisms millions of years ago.

[8]Translated from Czech language.

e.g. fear emotion and the fleeing behavior. These theories often view emotions as innate and biologically predetermined. For example [Ekman and Friesen, 1971] defines six basic emotions: anger, disgust, fear, sadness, happiness and surprise with each of those emotions corresponding to a particular facial expression. On the other hand, [Plutchik, 2003] defines eight basic emotions that form complementary pairs meaning only one behavior from a complementary pair can be active at the same time: acceptance and disgust, anger and fear, anticipation and surprise, joy and sadness. Moreover, Plutchik claims that the primary emotions can be combined together to form secondary emotions, e.g. combination of *joy* and *acceptance* forms *love*.

One of the problems of the *basic emotions* approach is that there is only a partial consensus over what emotions should be considered basic. Moreover, recent research [Krämer et al., 2014] is indicating that the universality of the basic emotions might not be that simple.

Dimensional theories

These theories do not consider discrete emotions but instead model emotions as defined in a multidimensional space. The number of dimensions that are used to represent emotions varies. Two of the commonly used dimensions are *valence* (or pleasure) and *arousal* (or excitation), e.g. [Posner et al., 2005]. The *valence* dimension should capture how pleasurable is the state to the agent. *Arousal* then captures how activated the organism is. For example, *anger* could then be defined as an affect with high arousal and negative valence. Now reader could argue that *fear* might be also viewed as an affect with high arousal and negative valence, so how could these two be distinguished? Due to that some of the theories are adding new dimensions that should help distinguishing some more special emotion types. For example [Mehrabian, 1996] added *dominance* dimension that can be used to distinguish between *anger* (high dominance) and *fear* (low dominance). One of the advantages of dimensional theories is that more often than not, from computational perspective a lower number of concepts used to represent emotions can be advantageous in a computer system. Having pleasure and arousal dimensions ranging from -1 to 1 might be the only thing that is needed in IDS system or a computer game.

Appraisal theories

Appraisal theories of emotion bring to front the cognitive part of the emotion process. From the appraisal theories perspective, the emotions are generated by cognitive processes that are occurring in the brain. This appraisal can be both conscious and unconscious and is influenced by the state the brain is in at the given moment. Personality, memories, actual emotions and moods, current body state – everything can influence this appraisal. The result of this appraisal process are the changes in the mental and physical state of a human being which includes elicitation of emotions. Notable examples of these theories are [Scherer, 2001] and Ortony, Clore and Collins Emotion Theory (OCC) [Ortony et al., 1988]. Big advantage of appraisal theories from the computer science point of view is that they approach the emotion problem in a way that tends to be more compatible with the standard implementation methods. Typically, appraisal theories search

for a set of conditions and features that are important for the appraisal process and then they are trying to explain how is the appraisal process influenced by those features.

Summary

To conclude this brief overview of emotion theories, it is worth to mention that emotion research in psychology is a very broad topic and various theories view emotions from vastly different perspectives. For example [Kleinginna and Kleinginna, 1981] listed that there are 92 emotion definitions – nowadays there is probably even more. That only proves us that the topic of emotions can be difficult to study in a systematic way. Also note that the distinction of the approaches to the emotion problematic mentioned here is somewhat rudimentary. E.g. one theory of emotions could easily fall inside more of these categories. For example, Plutchik's theory is using basic emotions but defines their dimension as well creating an emotion space. On the other hand, OCC is an appraisal theory but it defines 22 discrete emotions. Nevertheless, simulating affect without psychological background would be a rather unwise thing to do and some theories can provide a methodology providing the ability to implement affect simulation in computer system.

For a more theoretical introduction to the problematic of emotions and games see [Ravenet et al., 2016]. For a broad overview tracking the problematic of affect simulation in agent architectures and AI system see [Reisenzein et al., 2013]. The topic of affect simulation is going to be further discussed in the Section 2.7.2 where the integration of ALMA model of emotions [Gebhard, 2005] with EWA architecture is described.

1.4 Agent Architectures

This chapter provides an introduction to agent architectures that are used in academia and industry for the specification of behavior of IVAs. As there are many agent architectures that aim at different kinds of use-cases – e.g. architectures used for multi-agent systems, cognitive architectures or robotic architectures, the overview below is limited only to those architectures that are directly relevant to either specification of *believable* behavior for IVAs, are relevant from the IDS point of view or are relevant because of the EWA architecture implementation.

1.4.1 The Warhorse AI System

The Warhorse AI [Černý et al., 2014] system targets creation of ambient AI in AAA Role Playing Game (RPG)[9] that would control the NPCs and maintain illusion of their daily lives. Warhorse's goal was to design an architecture that would allow for specification of complex behaviors in a not-so-complex way and the computational requirements of the architecture at run-time would be reasonably low to fit into the available computation budget that is usually occupied by

[9]The role-playing game is called Kindgom Come: Deliverance and should be set up in Czech country around the year 1400.

graphics of the game. The example of the complexity of ambient behavior the architecture should handle is a common pub with an inn keeper and a group of quests.

The Warhose AI system defines *brain* as the main component of the decision making system of an NPC in the game. The purpose of the *brain* is to distribute updates to individual AI components which are called *sub-brains*. *Brain* manages *sub-brains* deciding which *sub-brain* will get the priority or whether multiple *sub-brains* can run in parallel. Each *sub-brain* has a priority and are defining a sub-sumption architecture [Brooks, 1991] but unlike in sub-sumption architecture the *sub-brains* are not interrupted immediately when higher priority *sub-rain* is scheduled to run, instead a transition or clean-up behavior is used.

Moreover, to speed up the development of AI with the Warhorse AI system, a graphical editor was implemented by the authors allowing scripters to fast prototype the behavior of the NPCs. The system comprises of agent-based language that is inspired by behavior trees and is called by authors Modular Behavior Trees (MBTs).

Additionally, Warhorse architecture introduced a concept of *smart objects* [Černý et al., 2015] allowing the developers to manage ambient behaviors such as sitting on a chair, drinking beer or opening doors in a feasible manner.

The concept of transition behavior[10] in Warhorse AI system and the behavior priorities are similar to the approach taken in this thesis with the exception of EWA not supporting parallel behaviors[11] at this moment.

1.4.2 FAtiMA

FAtiMA (Fearnot AffecTIve Mind Architecture) [Dias et al., 2014] is an agent architecture that was used in FearNOT! [Aylett et al., 2007] and ORIENT [Aylett et al., 2009] IDS systems. FAtiMA is a modular agent architecture featuring several components. The main component is the FAtiMA Core that is used to define the basic architecture template and interfaces that are then implemented by a set of modules. The core interfaces define an appraisal process that is based on the appraisal theory of [Scherer, 2001] and OCC [Ortony et al., 1988]. The Core appraisal process is divided into two phases – appraisal derivation that determines the appraisal variables of the event for the agent and affect derivation that takes these variables and comes up with a set of emotions and mood that will be associated with the event and stored into the memory. This functionality is implemented by a set of modules (in so called "FAtiMA Modular"):

- *Reactive Component* – is responsible for appraising the events in the environment by a set of appraisal variables (strongly based on the OCC theory).

[10]The transition behavior is a concept that allows the developers to handle changes of the behavior of the agent from one to another. Imagine a gardener watering plants in his garden with a watering can in his hand. Suddenly, his daughter enters the garden and the gardener would like to hug her. Doing this with a watering can in his hand might not be a good idea. So, he should put down the can and hug his daughter. From the agent architecture perspective, watering the garden and greeting daughter are two well specified behaviors that can be implemented. The "putting down the can" is the transition behavior that should occur when changing from "watering garden" to "greeting daughter".

[11]With the exception of movement that can be done in parallel with other actions.

- *Deliberative Component* – is handling the goal oriented behavior and planning for the agent.
- *OCCAffectDerivation Component* – is responsible for generating the emotions out of the appraisal variables got by the *reactive component*.
- *Motivational Component* – is responsible for generating basic human drives such as energy. This can influence the agent goals and event appraisal (e.g. seeing food when the agent is hungry is more desirable than when full).
- *Theory of Mind Component* – is used to model internal states of other agents. This component determines the desirability of the events for other agents as the current agent perceives it.
- *Cultural Component* – adds a further cultural dimension that affects the agent's appraisal and behavior. For example, in certain cultures it is undesirable when the agents are too close together which is leading to different appraisal of the events.

The FAtiMA's main goal is to define a robust modular appraisal architecture that can be used to model a plausible agent behavior based on several components. From the computer games perspective, FAtiMA offers more than is usually required to model NPCs in the game. FAtiMA might be used in a thin range of use cases that focus on the maximum plausibility of the character's behavior (as in FeatNOT! or ORIENT). Based on the goal of the designer, the FAtiMA architecture might be used as an inspiration for various types of phenomena that might be useful to simulate plausible agent behavior.

1.4.3 A Behavior Language (ABL)

A Behavior Language (ABL) [Mateas, 2002] is a custom reactive planning language that can be used to define agent behavior. ABL is based on Hap [Loyall, 1997] which it extends in several ways, the main point being the coordination of multiple agents. Also, the ABL language uses a concept of dramatic *beats* that helps organizing the agent behavior from the dramatic perspective (e.g. to maintain the ideal *tension curve* in the drama). ABL was used successfully to implement one of the most advanced IDS systems up-to-date – an interactive drama Façade [Mateas and Stern, 2003].

ABL uses a concept of behaviors to define short sequences of actions that solve a particular case or produce a particular piece of dramatic events. An example of such a behavior might be the agent waiting for someone to knock on the door. When this happens, the agents *sights*, comes to the door, opens them and greets the player that is entering. Each behavior has a *success test* a method that determines whether the behavior was successful and can be now discarded. ABL also defines a concept of *priority*. *Priority* is used to determine the order of actions when there is a conflict of multiple actions being scheduled to perform. ABL can handle parallel behaviors, it also supports joint actions and behaviors that are used to handle the coordination of multiple agents. The execution of joint behaviors in ABL is negotiated as follows (cited and slightly modified from [Mateas, 2002], p. 74):

1. *The initiating agent chooses a joint behavior for the joint goal based on signature matching, precondition satisfaction, and specificities.*

2. *If a joint behavior is found for the joint goal, mark the goal as negotiating and broadcast an intention to enter the goal to all team members, otherwise fail the goal.*

3. *If all team members respond with an intention to enter the joint goal, add the joint behavior (and behavior children) to the ABL.*

4. *If any team member reports an intention to refuse entry to the joint goal, broadcast an intention to refuse entry and fail the behavior when all team members respond with an intention to refuse entry.*

The exit from the joint behavior is handled in a similar fashion of broadcasting the exit intention and all agents confirming.

It was proven that the ABL can be used to implement a strong *story-based* system such as Façade [Mateas and Stern, 2004]. ABL language allows the designer a good control over the narrative that can be used to fulfill the author's dramatic goals while allowing for the player to interact with the story in a meaningful way. From the *character-centric* vs. *story-centric* view, the ABL philosophy and its use in Façade would indicate that it falls more towards a *story-centric* use case approach – or at least that it was the intended design pattern of the ABL authors.

1.4.4 Summary

There are many approaches to the design and implementation of an intelligent behaviors of agents or robots. Different approaches look at the problematics from different perspectives – e.g. designing a control architecture of robots one wants to deal more with reactive behavior and with solving the problems such as noisy sensory data and solving the problem of behavior oscillation that can result from that. On the other hand, multi-agent systems are dealing with hundreds or thousands of agents living in the environment where the goal might be, e.g. to elicit the optimal policy solving problems such as which aircraft should land first. The so called cognitive architectures such as ACT-R [Stewart and West, 2007] are looking into ways of modelling human like reasoning and deliberation by creating a cognitive models and while important from the research perspective they fall out of the scope of this thesis as their goal differs from the one of the thesis[12].

This Section presented the four agent architectures directly relevant to the research endeavor of this thesis. The Warhorse AI system was described as an example of state-of-the-art architecture from the computer game development perspective. The FAtiMA architecture was used in two advanced IDS systems and finally the ABL that was used to create one of the most complete IDS system up-to-date. There is one more agent architecture that has not been mentioned yet – *Comme il faut* [McCoy et al., 2014] that was used in a kind of a sandbox

[12]To produce a believable, story coherent behavior for IVAs living in a computer game like worlds allowing for interaction with the user.

social simulation game *The Prom week*. *Comme il faut* is discussed in Section 1.5.4.

For a broader overview of agent architectures with emotions see [Lin et al., 2012]. From AI in computer games perspective a broad overview of state-of-the-art approaches was created as a follow up of Dagstuhl seminar *Artificial and Computational Intelligence in Games* in 2013, where the author also participated on [Muñoz-Avila et al., 2013] and [Cowling et al., 2013] is [Yannakakis and Togelius, 2015].

1.5 Overview of Current IDS Systems and Games

This Section provides an overview of current state-of-the-art IDS systems and games.

1.5.1 Firewatch (PC game)

Firewatch [Campo Santo, 2016] is an adventure computer game in 3D environment seen from a first-person perspective released in 2016. There are two main characters in the game – a guy named Henry, who after going through a difficult life situation decides to spend several months working on a fire lookout in Shoshone National Forest and his supervisor Delilah. Henry (the player) communicates with Delilah by using a walkie-talkie and can steer the dialog by selecting one of the possible options. The outcome of these interactions then shapes the relationship between Henry and Delilah. The gameplay comprises of the player walking through the woods, solving various tasks such as fixing broken windows, communicating with Delilah, putting out fires or investigating the strange things that start happening in the forests one moth after his arrival. The game was received well[13] with generally favorable reviews praising its story, characters and dialogs.

Figure 1.3: Firewatch game seen from the first-person perspective set out in 3D environment. Copyright Campo Santo 2016.

[13]https://en.wikipedia.org/wiki/Firewatch states that by September 2016 the game reached close to one million sales.

From a technical point of view, Firewatch is a game with a strong narrative. So how is this narrative represented and how much influence over the narrative does the player have? The narrative in Firewatch is an example of *beads on a string* approach. The story is linear and the game ends in the same way every time[14]. The approach the developers used in Firewatch is to give the player an opportunity to influence some parts of the game while leaving the story itself intact. When Henry and Delilah are talking to each other, the player has an opportunity to choose the responses Henry gives to Delilah. This way he can steer the conversation revealing some parts of the mystery while closing the doors on another parts. He can also control how much Delilah will know about his life which will change their relationship as well as some parts of the game environment (e.g. notebook pictures, journal notes or appearance or disappearance of objects in the environment).

Moreover, the narrative in this game is written in a way it makes the player curious about what is behind things. Playing the game for the second time and trying out another dialog options can reveal additional parts of the mystery or more about Delilah's life. The advantage of this approach is also the fact that authoring dialogs is generally easier than authoring new game locations in the 3D environment. Thanks to that, Firewatch benefits from a strong story line that is well written and executed on one hand and on the other hand, allows the player some agency in the game which, while feeling important to the player, does not require the developers to craft new story branches by introducing new game locations. This approach is not uncommon in computer game franchise – e.g. [Bizzocchi and Tanenbaum, 2012].

From the computer game industry perspective the open question of IDS would be – can it be done even better while keeping the development costs and time at reasonable levels?

1.5.2 Façade

Façade is one of the most cited Interactive Digital Storytelling system up to date and rightly so. It is one of the most developed examples of a working IDS system until now. Façade was developed by Mateas and Stern in 2003 [Mateas and Stern, 2003]. It is an interactive drama game where the player takes the role of a long time friend of a married couple that is currently going through a crisis. There is no introduction to the story or the settings, the player is simply thrown into the game by entering the apartment of Trip and Grace – two main protagonists of the story. The game is seen from the player's perspective in pseudo 3D mode. The player communicates with the characters using natural language typed in the game chat window. The text input is then mapped to approximately 20 acts the NPCs can understand. The whole story takes place in the Trip and Grace's apartment and one game takes approximately 10 minutes of game play. Throughout the gameplay the player reveals information about Trip and Grace and in the next game run the player can use that information to push the characters towards a particular ending. The game can end in many ways, from the player being thrown out of the apartment, to the player saving the

[14]There is one alternate ending where the player can decide not to leave the woods at the end of the story, however, this has no further impact as the game ends at that point.

marriage of Trip and Grace. One gameplay can differ significantly from others and at the end of each game run Trip and Grace summarizes the outcome of the story as they understood it. E.g.: "So you are saying we should be more open to each other and attend a marriage counselor. Ok, we will think about that.".

Figure 1.4: Trip and Grace greeting the player in Façade. Copyright Mateas and Stern 2002.

From technical a point of view, Façade was authored using a behavior language (ABL) [Mateas and Stern, 2004] that is used to implement the character's decision making system. From the designer perspective the game is represented by beats. A beat defines a part of the story usually up to 30 seconds of duration where something interesting happens. Each beat has its tension level which is used by the system to pick up the right beat that will be next to maintain optimal tension curve of the story. The beats are programmed in a generic way allowing some of them to be interchangeable. The advantage of this is that this approach allows for emergent stories but still gives the developer some control over the overall dramatic impact.

There are some drawbacks to that approach. Firstly, there is a lot of work needed to author all the beats in a way that everything is working together. Also, Façade is not a sandbox simulation. On one hand, the story allows unprecedented freedom to the player, on the other hand the story has still a strong grip over what happens next and there is no possibility to for example jump out of the window or perform some other game breaking action (which makes perfect sense from the designer's point of view). Façade was evaluated in [Mehta et al., 2007] and [Roth and Vermeulen, 2013]. In [Roth and Vermeulen, 2013] it was also revealed that users have a problem understanding of what is their role in the story, what is the story about and how to interact with the game in a meaningful way through the text input. Some of the users ended up abusing the game as much as it was possible while some of the others did not have a clue of what was happening and how to influence it. Part of this could be solved by a tutorial – which probably would not fit in this type of the game. Other problems could be addressed by a

more robust NLU behind pattern matching algorithms used in Façade – but that is a general, open question – how to understand the user text input properly.

1.5.3 FearNOT!

FearNot! (Fun with Empathic Agents to Reach Novel Outcomes in Teaching) is a virtual drama aimed on anti-bullying education of children between 9 to 13 years of age [Aylett et al., 2007]. In FearNot!, the player takes a role of an invisible friend of a child that is being bullied. This child is a boy or a girl between 9 to 13 years of age that is a newcomer to a local school where he or she experiences constant bullying by his/her classmates.

The game works in two stages that are repeating. In the first stage, the player sees an episode from the victim's life – usually this is a scene where some bullying behavior occurs. In the second stage, the victim of bullying asks the player for advice. Based on this advice the victim might change (or not) the behavior in the upcoming episode that usually differs.

FearNot! is set up in a 3D environment (Fig. 1.5) and the NPCs speak with natural language. The player communicates with the victim through a text input (Fig. 1.6). The narrative in the game is not scripted but is a result of the *stage manager*. Moreover, NPCs in FearNot! are controlled by *Fatima* affect-driven agent architecture Fatima [Dias et al., 2014] that further promotes occurrence of emergent narrative.

The emergent narrative in FearNot! comes from the integration of *stage manager* and Fatima control architecture. The *stage manager* prepares the scene and settings and the agent affective architecture that controls the characters takes care of the rest.

Figure 1.5: FearNOT! game environment. The victim meets two bullies. Copyright Aylett et al. 2007.

Figure 1.6: FearNOT! game environment – advice window. Second stage of the game where the player gives advice to the victim. Copyright Aylett et al. 2007.

1.5.4 The Prom Week

The Prom Week [McCoy et al., 2013] is a social simulation game featuring high school students. The gameplay takes place during the last week before the Prom. The goal of the user is to solve social puzzles, e.g. "How to get Zoe to date Zack who is not popular in their class", etc. The narrative of the game is built on top of social relations between different characters. The player can steer these social relations by controlling the characters and selecting appropriate actions. The actions available are based upon the actual relations between characters that are conversing. The game is *turn based* and is set in 2D environment (Fig. 1.7).

Figure 1.7: The Prom Week game environment. Copyright McCoy et al. 2013.

As a control architecture of NPCs, a model *Comme il faut* [McCoy et al., 2014] was created that allows for rich social relations between characters. In particular, *Comme il faut* models (cited from [McCoy et al., 2014]):

- relationships (3): friends, dating, and enemies;
- social networks (3): buddy, romance, and cool;
- statuses (34): popular, embarrassed, angry at, pities, cheater, heartbroken, cheerful, confused, lonely, excited, popular, desperate, trusts, has a crush on, anxious, etc.;
- traits (44): competitive, sex magnet, witty, attention hog, brainy, deep, shallow, humble, arrogant, hothead, emotional, self destructive, etc.;
- social fact database labels (13): cool, lame, romantic, failed romance, gross, funny, bad ass, mean, nice, taboo, rude, embarrassing, and misunderstood;
- cultural knowledge base (CKB) adjectives (10): cool, lame, romantic, gross, funny, bad ass, mean, nice, taboo, and rude;
- CKB connection types (4): likes, dislikes, wants, and has

Note that shifting from a constrained, turn based, almost static environment as in Prom Week to a full-fledged 3D real time environment might pose significant

challenges for character design (e.g. animation synchronization, navigation, etc.). It is important to overcome these challenges for a wider applicability of believable characters in practice. Having said that, the *Comme il faut* presents a robust system for plausible simulation of IVAs social relationship which could be a nice addition to the EWA architecture.

1.5.5 Other Notable Examples

There is a couple more notable examples of academia and industry projects relevant to the IDS field.

Interactive Fiction

Interactive Fiction[15] [Ziegfeld, 1989] is a term that refers to parser based programs or games where the user can influence the story or the simulated world by telling the main character what to do at specific points in time. The world simulation is typically done by text descriptions the player is reading and the input to the system is often submitted likewise. Alternatively, multiple options of possible actions at the current story point are visualized to the player, e.g. through clickable options or hyper links. Sometimes, these games or works are referred to as text adventures as well.

There are large online databases where thousands of these systems can be found, for example ifarchive[16].

These systems can be seen as a step between the interactive game books and modern computer games that are available now.

Also, Nelson and Mateas explored the state space of Interactive Fiction game *Anchorhead* by using the Search-Based Drama Management approach, see [Nelson and Mateas, 2005].

Academia Projects

From the Academia's point of view project ORIENT (Overcoming Refugee Integration with Empathic Novel Technology) [Aylett et al., 2009] should be mentioned. ORIENT is built upon the FAtiMA framework and takes the player to a journey to an alien world inhabited by intelligent reptile creatures with unique and unfamiliar cultural background and customs.

Nothing for dinner [Szilas and Ilea, 2014] is a game where the goal of the player is to assure that family evening goes according the plan without anything bad happening. The problem is there is nothing for dinner and the father suffers from a mental illness and does not want to take his medicine. The player is playing a son of the father and the player's actions can steer the outcome of the evening.

Examples of immersive IDS systems are Madame Bovary [Cavazza et al., 2007] or Merchant of Venice [Porteous et al., 2010] that feature characters in 3D environment playing out parts of famous novels with the user allowed to

[15]The definition here is based on the definition available at http://www.ifwiki.org [28.12.2016].

[16]Available at http://www.ifarchive.org/ [28.12.2016].

participate in. From a technical point of view, both approaches are using planning techniques as a means to represent the story.

Beergarden [Endrass et al., 2011], [Damian et al., 2011] is a research project featuring a virtual beer garden scenario that is aimed on studying the differences between cultures when it comes to personal communication such as body language, use of gestures, spatial positions of interactors, etc.

[Rishes et al., 2013] presents Natural Language Generation (NLG) system that is able to generate different forms of the same story based on the persona of the story teller. The novel part of their work is an integration of the semantic representation of the story with NLG tool.

[Traum et al., 2015] created a digital system that allows people to have a conversation with a holocaust survivor. The system is based on pre-recorded conversations with the holocaust survivor that concerns various topics from common questions to survivors philosophical points of views and his experiences from and after the war. The system offers a face-to-face conversation with the survivor and uses natural language as an input. The evaluation results show that most users are highly engaged within the interaction and that the system is able to respond to most of the user questions.

Industry Projects

Life is Strange [Dontnod Entertainment, 2015] is a story based adventure game set up in a 3D environment. The player controls the character of Max Caulfield, a student of photography that discovers she can manipulate time by rewinding it at any moment. That introduces an interesting game mechanics – at every decision point in the game the player can make a choice, see what happens and then rewind time and pick the other variant. However, the problem is that only the most recent events can be rewound and it can be hard to estimate the impacts of decisions in the future. From a technical point of view, the game uses beads on a string approach where the decision points affects the game in the future by changing the relationships between characters. The game excels in providing high sense of agency for the player on one hand and keeping the narrative authoring maintainable on the other hand.

The Walking Dead Series [Telltale Games, 2012] is a survival horror video game with strong emphasis on storytelling. From the technical perspective, Walking Dead is an adventure game in 3D environment viewed from third-person view, where the player influences the actions of a main protagonist Lee Everett who leads a group of survivors in a world in the midst of zombie apocalypse. The game is focused on character and story development.

Heavy Rain [Quantic Dream, 2010] is a psychological thriller, action-adventure videogame set up in a 3D environment. The game features four protagonists that are involved in the mystery surrounding a serial killer. The player is able to influence the outcome of the various scenes and the game endings (this is a typical example of beads on a string approach).

The Sims Series [Electronic Arts, 2014] is a video game series focused on a life simulation in sandbox 3D game environment. The player is able to control the actions of multiple characters changing their habits, customs and social relationships. Player can also change the environment the characters live in (e.g. rebuild the house, etc.). The games are featuring a robust simulation of the characters

needs – from the basic needs such as hunger to a more complex ones such as a need to meet other people (being social). The Sims are praised for their plausible character model and the amount of freedom the game offers.

The Mass Effect Series [BioWare, 2007] is a science fiction, video, role-playing game series set up in 3D environment and featuring action gameplay. The player can influence the relationships of the main protagonist – Shepard – and characters in his squad by making choices in dialogs at certain points of time. For more in-depth analysis of Mass Effect see [Bizzocchi and Tanenbaum, 2012].

Coming Out Simulator [Case, 2014] is a simple web based game that can be used as a minimalistic prototype of beads on a string approach to interactive narrative. The player relives a critical life experience of the game developer and have a limited influence over the game by making decisions in the dialogue controlling the things the main protagonist of the game says to his parents and to his date.

From other notable examples of games featuring complex interactive stories there is the Final Fantasy Series [Square Enix, 2016], Star Wars: The Old Republic [BioWare, 2011], Knights of the Old Republic [BioWare, 2003] and certainly many others.

1.6 Summary

This Chapter provides an overview of the IDS field, theoretical perspective and modern approaches solving problems of the IDS, an introduction to problems of believable human-like characters design and a summary of most influential agent architectures and IDS systems from Academia's and Industry's point of views.

In Academia, the problem of IDS is solved by creating custom agent architectures or languages tailored for a particular use case and/or combining them with the planning approach. The Industry approaches the problem with the beads on a string approach maintaining the illusion of interactivity for the player but constraining the development work needed to create all the story lines, scenes, assets, conversations and characters. Neither the Academia nor the Industry has managed to solve the *wicked problem* of IDS completely yet. That is expected as solving the *wicked problem* here would be a huge step forward to general AI.

Moreover, there seems to be a little reuse of concepts or approaches between the two. This is not surprising as the Academia and Industry are solving different kinds of problems [Yannakakis and Togelius, 2015][17]. The following Chapter proposes on a conceptual level[18] a minimalistic affect modulated agent architecture suitable for the development of IDS system for a medium-sized drama. The hope is that by introducing only the minimal concepts necessary the architecture might be useful both from Academia's and Industry's standpoint.

Note that because of a broad scope of topics that are considered the most relevant related work is presented and discussed at appropriate places later in

[17]The academia tend to be more interested in the general approaches and the architecture conceptual layout caring less about the ease of user or the performance of the system. The industry, on the other hand, needs system that run in real time with limited resources and are simple to use.

[18]Implementation is discussed in Chapter 3.

the thesis[19].

The whole research endeavor of this thesis leads ultimately to the Goal 4 *"Design and implement a methodology for estimating the quality of virtual drama produced by a game or some IDS system."* that warrants a theoretical background section of its own. However, as the Goal 4 is addressed in the last Chapter of this thesis the relevant theoretical analysis and related work are presented in Chapter 4.

[19]Some of the works from this Chapter will be discussed again later at an appropriate place.

2. Emohawk Agent Architecture (EWA)

The previous Chapter introduced the IDS field and the problems connected with developing a working IDS system. Main aspects of IVA's behavior that are necessary for *believable* behavior in normal daily life situations has been sketched. These cover the problematic of how to design systems that can produce intelligent decision making of IVAs in general. Then how to enrich these systems with affect simulation – if necessary. Moreover, the decision making system with affect simulation can produce social behaviors that need to be simulated in the environment. This can influence movement styles of the agents so a system for socially believable movement of agents in 3D environment is necessary. Last but not least, as the simulated scenarios get longer, the need to maintain the coherency of the resulting narrative that emerges from the interactions of the agents and the user is more important and needs to be somehow tackled by the resulting system.

This Chapter analyzes the problems above in a greater detail. The goal is to design a minimalistic agent architecture suitable for developing IVAs effectively for the purposes of simulation of medium-sized dramatic situations (Goal 1). By "medium-sized" it is meant a computer game producing a coherent story that lasts for ten minutes[1]. By "developing effectively" it is meant that the architecture should introduce only the most necessary concepts needed for the specification of the above not to overburden the developers and allowing to prototype agent behaviors quickly. Alternatively, that requirement can be understood as minimizing the number of concepts needed to specify a believable IVA behavior in 3D environment.

The outcome of this Chapter is a general specification of concepts of an agent architecture called EWA (Emohawk Agent Architecture). It aims towards the specification of minimal concepts that are required to represent effectively behavior of characters in medium-sized dramatic situations helping to tackle the *wicked problem* of a medium-sized drama. There is no need to start from scratch – over the years there have been many agent architectures, AI systems, agent languages and platforms specified, yet it seems that none of those is the silver bullet approach for designing AI in a computer system (e.g. a computer game) in general. There are a number of reasons. To name a few, typically every approach to AI design is tightly coupled with a particular use case it tries to solve – this is a natural thing and not bad in general, it just means that such AI system may not scale well to another use cases. To give an example, consider a robotic vacuum cleaner (RVC). The RVC drives around in a room, avoids obstacles, vacuums the floor and when the battery is low it goes back to the recharge station to get more power. There is no doubt that this kind of behavior can be effectively represented by a set of if-then rules, e.g.:

1. **If** the battery is low or vacuuming for 30 minutes **then** go to the recharge station (and avoid obstacles).
2. **If** there is an obstacle in front of me **then** turn.

[1] One can imagine a Little Red Riding Hood story complexity.

3. **If** true **then** go forward and vacuum.

Reader will probably agree that if simple if-then rules are sound enough to solve the decision making of RVC, there is a little gain in doing this differently (e.g. using FAtiMA modular [Dias et al., 2014] for RVC is probably an overkill). But if-then rules will not necessarily scale well to another use cases such as finding an optimal plan for assembling a submarine or design an NPC with emotions and believable behavior in a computer game. For these use cases, there is a need for better abstractions and different approaches (e.g. planning approaches, a Belief–desire–intention software model (BDI)). In general, the problems the architecture or the agent language is trying to solve are important. For example contrasting Academia AI research with the state-of-the-art computer game development industry can be seen a disjoint between goals that are being solved [Yannakakis and Togelius, 2015]. The bottom line of that is that there is a little re-use of approaches between the two[2].

The last issue causing problems with penetration of a particular agent architecture/language to become mainstream is technical. Simply put – the implementation does matter. Having a game engine using JavaScript to script AI for NPCs, it might be troublesome connecting C++ based AI implementation into the system, so perhaps scripting just the important concepts of that engine within the JavaScript might be a more effective approach. Or let us say there is a Java based if-then rules system for RVC but the new RVC prototype uses C++ API – again it might be better to re-implement rather than hack the Java architecture inside (e.g. for performance reasons).

Thinking about how to circumvent the issues outlined above and how not to fall into the gap of yet another agent architecture this Chapter proposes below an architecture containing a minimal number of simple concepts that need to be captured in order to successfully achieve the goal of designing and implementing behavior for IVA in medium-sized drama in a 3D environment. All of those have been implemented and used for developing SD One and SD Two games. However, the actual implementation is tightly coupled – if not with Pogamut platform – then definitely with the underlying 3D engine technology. While the architecture could be in theory used in other engines, the author of the thesis believes that re-implementation of the concepts in the framework, platform or the engine the architecture should run on is warranted and is also a necessary requirement for the real world application – such as a computer game.

That is why this Chapter is more conceptual and the actual implementation of EWA within the SD One and SD Two use cases is discussed in Chapter 3. There, the architecture's strong and weak points are summarized with the lessons learned.

To summarize this Chapter provides a high-level overview of the EWA architecture that should allow the developers to re-implement the concepts in their own systems adjusted to their custom needs and requirements.

[2]Though it seems this is changing for better, e.g. [Černý et al., 2015].

2.1 Problem Analysis

How to produce an intelligent behavior in a computer system? There are many solutions to that problem starting from some basic expert systems that are composed of a set of if-then rules, finite state machines that are – when small – easy to design and understand, behavior trees, neural networks, planning systems, cognitive architecture, etc. The best suited approach depends heavily on the problem being solved. For example, doing image classification with a set of if-then rules might work to some extent, but a system based on a neural network will produce better solutions. The decision of what approach or abstraction to use for the particular problem is not that easy all the time. The if-then rules, Finite State Machines and Behavior Trees can be used to solve similar kinds of problems, however, for some of those problems one approach might outperform the other ones, e.g. when it comes to saving time spent on designing.

The problem solved in this work is the creation of an architecture allowing for designing *believable* behavior of IVAs and producing the *emergent narrative*. The resulting experience is interactive – the player needs to be able to influence the narrative somehow – e.g. by controlling one of the characters or playing her part in the story as well. The architecture is meant to be used to control fully embodied characters in a 3D environment (e.g. in a computer game with The Sims [Electronic Arts, 2014] like graphics, e.g. Fig. 2.1). This requirement is actually quite important as solving the problem of virtual agents in the 3D environment is much harder than in 2D.

Figure 2.1: Example of embodied intelligent virtual characters living in 3D environment of the game SD Two.

As probably every good writer knows – to create a good narrative, one needs to create believable and interesting characters in the story. As the simulated situations contains characters engaging in social behavior, looking at this problem from the character-centric perspective seems to be a good idea [Louchart and Aylett, 2004]. This means that to achieve the goal of producing a medium-sized drama designing a system suitable for the development of believable intelligent characters is a good starting point. From the perspective of IVA *believability* as discussed in section 1.2.1 this means that the characters need to be able to:

- Perform goal oriented behavior.
- Express emotions. As there are usually more characters in the IDS system, not just the character's emotions but also relationships between different characters need to be captured.
- Be able to move around in the 3D environment in a plausible fashion with respect to social relations between the characters.
- Be able to communicate with the other characters and the player.

Taking the character-centric approach, the resulting narrative then emerges from the interaction of the player and the IVAs in the environment. As there will be more IVAs in the game world there will be a need for synchronization of the behaviors of different IVAs, e.g., when the IVAs will be interacting with each other or move together. This means the architecture needs to account for behavior interruptions and transitions – e.g. when the player triggers an action in the middle of the conversation of characters.

As the IDS system will get more complex, there will be a need to debug the system. Graphical debugging tools help the agent architectures to scale better[3] – it would be beneficial to design the architecture with this in mind.

Putting everything together the requirements on the capabilities of the architecture can be summarized as follows:

R1. The architecture should allow for defining the overall story shape yet to generate behavior in an emergent manner within the story's boundaries.
R2. The architecture should allow for reactive behavior with transitions (to swiftly change behavior and depict a transition behavior).
R3. The architecture should allow for affective behavior to portray emotions, especially the relations between characters should be represented.
R4. The architecture should allow for user interaction.
R5. The architecture should allow for synchronizing of the characters with respect to their behavior.
R6. The architecture should allow for *believable* spatial behavior in the 3D environment (e.g. when the characters are walking together to reach some place).
R7. The architecture should support debugging of the agent behaviors.
R8. The architecture should allow all of the above using minimal concepts necessary, enabling easy specification of behaviors for the developers.

2.2 Related Work

2.2.1 General Approaches

BDI paradigm

In the agent community a popular approach for decomposing and developing the agent behaviors is the BDI paradigm [Rao and Georgeff, 1995]. BDI decomposes the agent behavior into three concepts:

[3] [Korstanje et al., 2016] showed that more experienced developers use more features of the Integrated Development Environments than novices.

- *beliefs* – capturing the information about the environment that are available to the agent
- *desires* – capturing the goals the agent would like to achieve, more desires can be active at the same time and some of them might be in conflict (e.g. an asocial person needs to go shopping)
- *intentions* – capturing behavior plans that tries to solve some of the agent desires

The BDI approach is often used in agent languages such as AgentSpeak(L) [Rao, 1996] or in multi-agent systems [Wooldridge, 2000]. The EWA architecture has been heavily influenced by the BDI paradigm as well – events and actions form the EWA's belief base, the EWA's goals resemble desires in the BDI, and the EWA's intentions map to the BDI intentions well.

However useful the BDI paradigm is, it needs to be grounded to the implementation level. Questions such as what should be used to specify the low level behavior of the agent or the behavior plan executed by intention need to be answered. Fortunately, it is not necessary to start from scratch as there are many approaches frequently used in AI programming. These are the scripting languages used for programming AI in games such as Lua [Ierusalimschy et al., 1993], Finite State Machines (FSMs) and their hierarchical counterparts the (h)FSMs [Harel, 1987] and Behavior Trees (BTs) [Colledanchise and Ögren, 2016] that are becoming more and more popular among computer game developers (see [Champandard, 2017] and Chapter 6 in [Rabin, 2013]).

Scripting

Scripting languages offer a lot of expressive power to developers on one hand and are usually simplifying the development of behaviors either because they tend to be high level or they limit the number of programming concepts they use, e.g. [Ierusalimschy et al., 1993]. There is a little doubt that it would be possible to achieve most if not all of the requirements with the use of a scripting language. However, the low level abstraction of a scripting language would make the implementation cumbersome if not impossible.

Having said that, the scripting languages remain useful in specifying small pieces of code that achieve a simple, well defined thing. A nice addition to EWA architecture would be use of a scripting language to script logic in low-level actions.

Finite State Machines

The FSMs and the (h)FSMs [Harel, 1987] enable the encapsulation of pieces of behavior in states. Moreover, the (h)FSMs enable the creation of hierarchies of these states. This seems like a natural representation of a complex behavior of an NPC in a 3D environment and indeed, the FSMs are used in computer games to prototype the AI for characters. The states are used to represent a particular behavior such as attack, and transition edges are handling the switching of the active behavior to another one.

However, it turned out that there are many problems connected with using the FSMs for behavior specification [Champandard, 2007]. The FSMs are not

easily extensible and do not scale well. Also, the FSMs are reactive and the goal directed behavior is hard to express using them. Having said that, the concept of FSMs can be beneficial if used right. As with the scripting languages, the FSMs might be used to encapsulate simple, well defined reactive behavior – such as follow the agent somewhere. In EWA the FSMs are often used to specify behavior on intention and action level.

Reactive Planners

Reactive planning [Bryson, 2001] is a concept often used in robotics when designing reactive behavior for robots based on sensors. Here the term *planning* is not used in the meaning of the classical planning such as a search for a plan to achieve something, instead it is typically a fixed behavior that aims towards solving a particular problem, such as the obstacle avoidance of the robotic vacuum cleaner. Reactive planner, such as POSH [Bryson, 2001] or yaPOSH [Gemrot et al., 2013] enables quick prototyping of reactive behaviors, and in the case of yaPOSH were used to prototype behavior of the NPCs living in a 3D environment.

However, standalone reactive planners are insufficient on their own as they do not support explicitly (R1), (R3) and (R5). However, the concept of reaction to the changing state of the environment was used in the EWA's Reactive Factories (see Section 2.5.1).

Classical Planning

As mentioned in Chapter 1, the classical planning approach is being applied to the IDS [Riedl et al., 2011]. Overall, the planning approach seems to be more compatible with the strong-story approach and not the character-centric approach – e.g. planning over the low level action's abstraction might be problematic as the plans might get very long which then slows down the performance of the planners.

Also, not all of the requirements for the architecture can be solved by a planner, e.g. (R3) and (R6). Nevertheless, the abstracting the higher level plot concepts of stories and using a planner to get a consistent story might be a future direction that could enable the architecture to scale better regarding the plot consistency and its length.

Behavior Trees

The Behavior Trees (BTs) [Colledanchise and Ögren, 2016] are becoming popular among game developers (see [Champandard, 2017] or Chapter 6 in [Rabin, 2013]). It seems they present the right level of abstraction for behavior specification in a typical computer game. The BTs decompose behavior into nodes that are organized in a tree structure. The nodes in a BT return success or failure based on whether the behavior defined in the sub-tree, defined by a particular node, succeeded or failed. There can be many node types in a BT based on the interpreter used. The most common ones are the "and" nodes that require all child nodes of a particular node to succeed before the node declares itself as succeeded, and the "or" nodes that require only one of the child nodes to succeed. Another advantage of BTs is the fact they can be visualized and edited graphically. However,

for the EWA's use case they do not explicitly support (R1), (R3) and (R5).

Nevertheless, BTs are a useful concept that was used in the EWA to define a hierarchy of goals. The hierarchy of goals is formed using "and-or" trees.

Cognitive Architectures

In general, cognitive architectures such as ACT-R [Stewart and West, 2007] and LIDA [Baars and Franklin, 2009] are trying to model a human decision making on a detailed level. This comes at a price though. Concepts used by these architectures are usually not easy to use and the starting time needed to grasp the architecture intricacies is non trivial. Also, the case solved by these differs from the one in this thesis in multiple ways. For example, human level decision making plausibility is not required.

2.2.2 Selected Agent Architecture and Languages

This Section presents selected languages and agent architectures that are the most relevant to the EWA.

ABL

There is no doubt ABL can be used to implement a full-fledged IDS system producing a medium-sized drama such as Façade [Mateas and Stern, 2004]. However, from the *character-centric* vs. *story-centric* point of view, the ABL philosophy and its use in Façade would indicate that it falls more towards the *story-centric* approach – or at least that it was the intended design pattern of the ABL authors. As the EWA architecture tends to stay more on the character centric side of things, the use of ABL seems to be sub-optimal.

FAtiMA

FAtiMA [Dias et al., 2014] is an agent architecture with a main goal of defining a robust modular appraisal architecture that can be used to model a plausible agent behavior. From the computer games perspective, FAtiMA offers more than is usually required to model NPCs in the game. FAtiMA was used in FeatNOT! [Aylett et al., 2007] and ORIENT [Aylett et al., 2009] projects that required the maximum plausibility of the character's behavior.

Based on the goal of the designer, FAtiMA architecture might be used as an inspiration for various types of phenomena that might be useful to simulate plausible agent behavior. In particular, EWA is a modular architecture and some of the modules closely resemble those introduced in FAtiMA – especially an approach similar to the *Reactive Component* is used in EWA for event appraisal and the integration of ALMA emotion model could be viewed as implementing the *OCCAffectDerivation Component*.

However, FAtiMA is addressing issues beyond our needs, such as introducing a Theory of Mind component or aiming on emotional and cognitive plausibility. Also, the overall complexity of FAtiMA might go against the requirement R8.

Warhorse AI system

Commercial solutions such as the Warhorse AI system [Černý et al., 2014] are not publicly available and in general are not designed to support (R1), as the *emergent narrative* approach is not "safe" from the computer game design perspective for it might result in sub-optimal player experience when the author's goals are not conveyed due to the emergent nature of the system. Note that [Černý et al., 2014] also emphasize the importance of transition behavior as proposed by the author of this thesis in the previous work [Bída et al., 2011a]. However, the notions of smart objects and smart areas that encapsulate the behavior of NPCs engaging in daily life scenarios seems to be the right direction to take. From this perspective, the Warhorse AI system represents a promising starting point for the implementation of a large-scale drama.

Comme il faut

The *Comme il faut* system [McCoy et al., 2014] used in The Prom Week game [McCoy et al., 2013] represents a robust solution to a social relations simulation. The entire Prom Week game is built around social relationships and their manipulations, and represents one of the best systems in games so far. In this thesis, there is a need for a social relationship simulation as well but a simpler approach was used to support the ease of use and debugging.

Moreover, their system is tweaked for a turn-based 2D game use case. Here, on the other hand, IVAs in real time 3D game environment are required. Also note that while a need to simulate social relations is present here as well, the level of detail the *Comme il faut* architecture provides is not required. Nevertheless, the social relationship model of *Comme il faut* could be a nice addition to the architecture in the future work.

Other Notable Architectures

Advantages of complex solutions, e.g. [Porteous et al., 2013], [Marsella and Gratch, 2009], addressing issues beyond our needs, such as equipping agents with general planning abilities and/or making them plausible emotionally and cognitively, come at a price: increased design time and/or slower real-time computation. Complex approaches that work in a timely fashion such as SOAR [Laird, 2012] can still overburden the designer on one hand (R8) and are not handling (R6) explicitly on the other hand.

Solutions used in multi-agent systems (MAS) such as Jason [Bordini et al., 2007] which is an extension of AgentSpeak(L) [Rao, 1996] solve different kinds of problems, such as cooperation of agents in task solving or negotiation between agents. Although there seems to be some commonalities (e.g. MAS architectures are often using BDI paradigm), in general, it seems that the usage of MAS architectures for solving computer game like problems might be problematic [Píbil et al., 2012]. For a broader survey of approaches to MAS see [Bordini et al., 2006].

2.3 The Proposal

As none of the approaches presented above did suit the requirements on its own, there was a need to address the gap of a missing architecture that would explicitly support requirements (R1-R8) in a medium-sized drama. A custom agent architecture is proposed for development of IVAs: a minimalistic affect-modulated action selection mechanism working with transition and affective behaviors with a component for plausible movement of characters in a 3D environment and with a *story controller* component for synchronizing the characters and making high-level adjustments to the story in run-time. The architecture supports implementation of graphical tools that can be used to overview the IVA's behavior in the runtime, helping the developer to make sense of what is happening in the IVAs minds. The architecture is fully integrated with an emotion model and a navigation manager that allows for socially plausible movement of IVAs in 3D environment.

Technically, the architecture can be conceived as an extension to classical finite state-machines and rule-based systems. The strength lies in adding several features without which the development of medium sized drama systems would be problematic. The architecture was implemented and used to develop two IDS systems, SD One and SD Two. Next, an overview of the architecture is presented.

2.4 Basic Overview

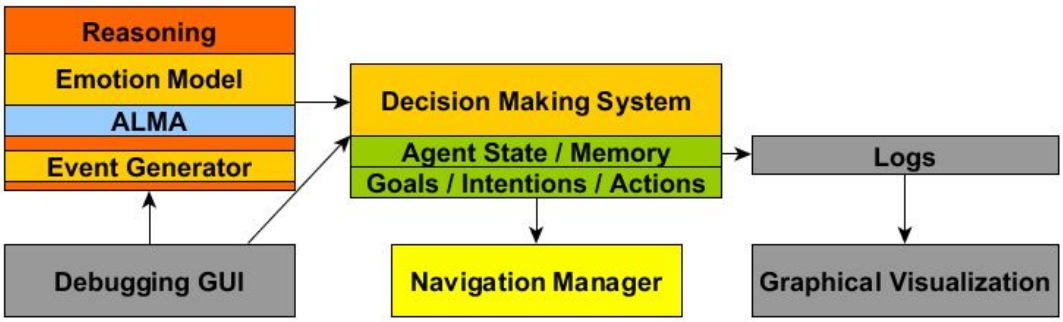


Figure 2.2: The EWA architecture overview. The key components of the EWA architecture are portrayed in the picture. Reasoning components that consists of Event Generator and Emotion Model handling the affect simulation for our agents are used by the Decision Making System that works with Goals, Intentions, Actions and the Agent State. The Navigation Manager component is able to handle the movement of IVAs in a 3D environment. The Debugging GUI can be used to overview the agent current state. Moreover, the changes in the Agent State are saved to log files that can be visualized graphically, more details in Section 3.5.

From a high-level overview, EWA integrates together the four main components (see Fig. 2.2): the navigation manager – handling the spatial movement of the characters, the reasoning component with the integrated emotion model – handling the affect simulation of the characters, the decision making system –

deciding "what to do next" and the graphical debugging tools that can be used by the author to debug the IVA's behavior and to graphically visualize the (offline) logs of what happened in the environment.

The main concepts defined and used by the EWA architecture are:

- **Events.** Represent an event that happens in the environment. That is, in general, a change of state in the environment, typically caused by an action performed by another agent. Events are stored in the IVA's memory and can be used to reflect what happened in the environment. They also serve as inputs to the emotion model, giving rise to the character's emotions. Events are part of the agent belief base, among other things, such as the current state of the environment.
- **Actions.** Representing an action that can be performed by an agent in the environment. This can be an atomic action such as playing an animation on the character's body (e.g. waving) or a move action that can last for a longer period of time. Actions define an *"API"* the developer uses to specify the behavior of agents.
- **Goals.** High-level goals of the agent (resembling desires in the BDI paradigm). These might be: "to have a conversation with other agent", "to reach a target place in the environment" or "to build castle defences". Goals can be hierarchical and if they are not, they are typically associated with an intention.
- **Intentions.** Implementations of behaviors that are supposed to fulfill a particular goal or group of goals (resembling the intention in the BDI paradigm). Intention is a place where the implementation of complex logic controlling the IVA's behavior takes place.
- **Emotions.** Emotions generated by the emotion model are based on the events in the environment. Emotions are stored in the memory and can be used to modulate goals or agent's behavior implemented by intentions.
- **Feeling.** Custom affect defined in the emotion model implementation that captures relations between the agents in the environment.
- **Steerings.** Algorithms implemented by the navigation manager that are used to move the characters in the environment. These can be highly parametrized and can take into account the relationship between agents.

```
while true do
    receiveUpdateFromTheEnvironment();
    generateAndProcessEvents();
    selectWhatToDoNext();
    executeAction();
end
```

Algorithm 1: Architecture Main Update Loop

The basic reactive loop of the architecture (Alg. 1) works as follows. Whenever there is an update of the state of the environment – which should be received regularly every N milliseconds (e.g. 250 ms) (*receiveUpdateFromTheEnvironment()*), the "reasoning phase" is triggered. During that phase the update is processed and events are generated (*generateAndProcessEvents()*). Afterwards, the

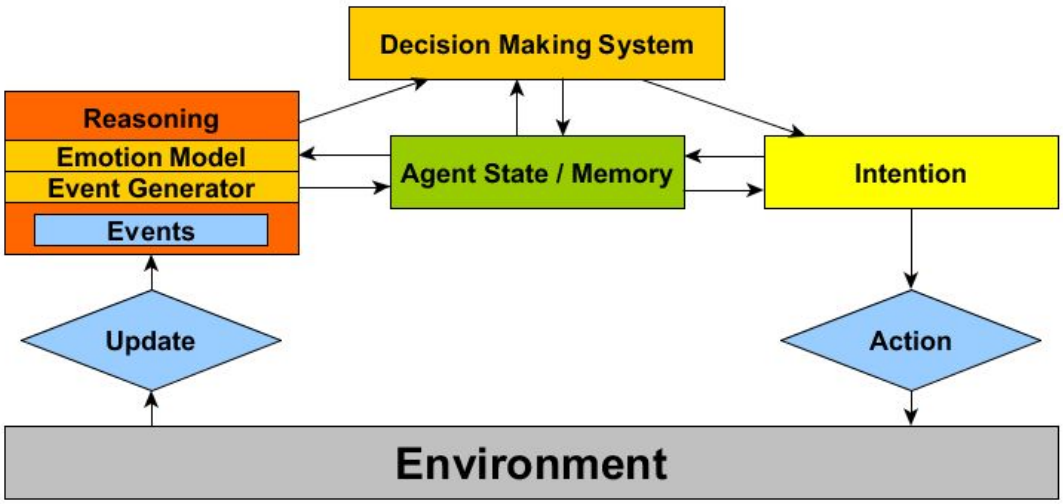


Figure 2.3: Basic decision making flow of the EWA architecture. When an update of the environment state is received it is processed by the "reasoning" components that generate events and change the agent's state. Then, the Decision Making System component is triggered. The outcome of the Decision Making Component is an intention that should be executed next. The intention then selects an action that is performed by the agent in the environment.

Decision Making System is triggered (*selectWhatToDoNext()*) and choses which action should be executed next (or whether the agent wants to continue executing the previously selected action). Then the action is executed in the environment. Schematically, this is portrayed in Fig. 2.3.

2.4.1 Representing the Agent Behavior

This Section discusses how the agent behavior is represented by EWA with *Goals*, *Intentions* and *Actions*. Goals represent conditions defining the moment when a particular behavior is fulfilled and the hierarchy of complex behaviors. An intention is an implementation of behavior that typically tries to fulfill one or more goals at once. The actions are low level actions IVA can perform. A simple example of a behavior defined in EWA can be found in Fig. 2.4.

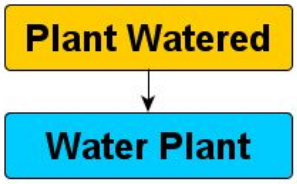


Figure 2.4: Example of a simple behavior representation in EWA. Suppose the agent is required to treat plants properly and water them. To define this in EWA, two things are needed – first, a goal that checks whether the behavior outcome was reached needs to be created, second, an intention that will implement this behavior (e.g. find a watering can, approach plant and water it) needs to be created.

Goals

The goals are used to specify success and failure conditions of a particular behavior. An agent can have multiple goals defined at the same time in the decision making system but only one goal can be active at a time. The active goal then determines what kind of behavior IVA will perform by scheduling a particular intention for running or by defining a set of sub-goals that need to be fulfilled in order for this goal to be fulfilled. This can form a hierarchy of goals, see Fig. 2.5. The parent goal can define whether all of the sub-goals need to be fulfilled (the "and" tree) or whether just one of the sub goals needs to succeed in order for this goal to succeed (the "or" tree)[4].

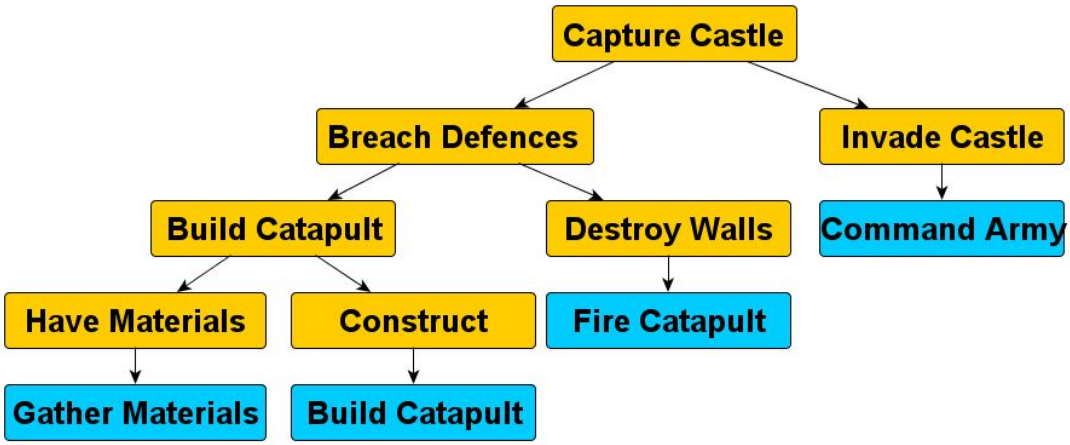


Figure 2.5: Example of a complex hierarchical Goal. The top level goal here is to invade an enemy castle. This can be decomposed into two goals – breach the castle defences and invade the castle. The breach defences goal can be further decomposed into building and firing a catapult. All of the sub goals need to be fulfilled here in order for the goals to succeed but it is possible to require only one of the goals to succeed for the parent goal to succeed (the "and-or" tree). The "leaf" goals will typically schedule an intention to run that implements the behavior leading to the goal fulfillment.

As more top level goals can be defined at once in the decision making system, there is a need to arbitrate which of the top level goals will be performed. This is done by checking the goal priority that is explicitly defined for every particular top level goal in EWA.

When a hierarchical goal is executed, a list of its sub-goals is checked. For every sub-goal in the list, starting from the first, a success or failure conditions of the goal are checked. If the goal has not succeeded nor failed, it is scheduled to run and the same algorithm is applied to this goal recursively, as the goal might have its own sub-goals defined. Note that a leaf goal in this hierarchy (see Fig. 2.5) will typically schedule an intention to run, to fulfill the conditions of this goal.

[4]The goal hierarchy forms "and or" behavior tree.

Intentions

Intentions implement a particular behavior in the EWA architecture that is typically goal-oriented, in the sense that it tries to fulfill the success conditions of a particular goal or a set of goals. Which intention will be run is determined by the currently active goal hierarchy. In EWA, only one intention can be executed at once. The intention monitors the state of the environment and the state of the agent and decides what action to do next.

As there are typically more goals defined and the decision making system can decide it is time to change the currently active goal, there is a need for support of transition behaviors that implement the change of the behavior from one to another. This is an important requirement as it is hard to plausibly implement a transition between two different behaviors without an explicit support for the transition. Imagine a lumberjack cutting down the trees in a forest. Suddenly, his wife comes and brings him lunch. The lumberjack should now probably stop cutting down the trees, greet his wife and eat his lunch. Before this happens however, he should put down his axe. Putting down the axe is the transition between the "cut trees" behavior to the social behaviors of "greeting wife" and "eating lunch"[5].

In EWA, the transition behaviors are implemented on the intention level. When the goal is changing, there will usually be an intention active that was scheduled by the previous goal. The new goal can cause a different intention to be executed. Before the new intention can execute, the previous one will be notified that it is time to "freeze" or "switch out" of the behavior defined by this intention to some kind of a "neutral" state from which the agent can continue with the next intention. The previous intention will be notified about which intention will run afterwards, so it could define some special tweaks to account for some special cases.

The transition behaviors define two execution phases of an intention lifetime but the intentions define more phases, to account for other special phases such as initialization. The list of all intention execution phases defined in EWA is as follows:

- **Initialization.** This stage is executed the first time the intention is scheduled to run. Here, all the preparations for the behavior defined by the intention should be specified. This can be limited to initialization of particular modules or even preparing the scene in the environment by e.g. approaching another agent in the world.
- **Execution.** This is the main executing state of the behavior defined by the intention. Here, the main logic of behavior is defined. In EWA, there are no constraints on what kind of mechanism will be used to determine next action – it can be a simple FSM, a list of if-then rules or a behavior tree.
- **Freezing.** It may happen that a goal with a higher priority should take control over the agent and the currently active goal and its intention should be interrupted. Before that happens, the currently active intention freezing phase is executed, which allows to specify the transition behavior ("put down the axe") if needed.

[5]The author presumes that it is advisable not to do this with an axe in the hand.

- **Resuming.** If this intention was already being executed and was frozen by another intention, this phase will be called first should this particular intention be executed again ("pick up the axe").
- **Finishing.** When the goal associated with the intention succeeds or fails, clean up actions or a transition to some "neutral" agent state can be executed before the intention is discarded.

A transition behavior will occur when a) a currently active goal is interrupted by a goal with higher priority, b) a currently active goal succeeds or fails and a goal with a previously frozen intention is resumed, or c) a currently active goal ends and a new goal with a new intention is initialized. In each case, the transition behavior has an outgoing and an incoming part, which can be implemented in respective stages of the two behaviors. The two parts can be linked smoothly since the two behaviors are informed about each other.

Actions

Actions are used to capture simple, atomic or durative behavior of IVAs. This could be actions such as "approach another agent", "pick up or put down the axe", "wave at someone" etc. An example of a durative atomic action can be: "follow agent one somewhere in the environment". The action goes through an initialization, an execution and a clean up phase:

- **Initialization.** When the action is scheduled to run, it will first enter the initialization phase where the developer can initialize all the modules the action requires. For example, the action "follow" will initialize the navigation manager module in this situation.
- **Execution.** The main action is the execution phase. Simple actions such as "wave at somebody" will have a single line of code, making a gesture or an animation with the agent's body and then will end. Other actions can last longer and their execution phase will be called repeatedly, so the action can react to the changing conditions in the environment. When the action is finished (which should be a part of this phase as well), this phase will end and the next phase will be entered.
- **Clean up.** When the action was finished or cancelled, the clean up phase is being run to stop or de-initialize any modules or behaviors that are not required anymore.

2.4.2 Summary

This section presented basic concepts of the EWA architecture that are used to represent IVAs behaviors. The simplest behaviors are handled by *actions* – e.g. *make the agent tell a joke*. Actions are scheduled by intentions that represent complex behaviors such as "to greet another agent" which can comprise of approaching that agent and saying "Hi to him/her.". Intentions are scheduled by an active goal that defines the success or failure of a particular behavior. For more complex behaviors that require multiple complex steps that need to be conducted, a hierarchy of goals can be used (see Fig. 2.5, goals form the "and-or" trees). Actions and intentions have several execution phases in EWA and there

is a support for transition behavior on the intention level. To be able to react to the environment, events that capture all the important concepts the agent needs to understand are introduced.

The world abstraction EWA agent is working with is formed by a union of actions and events. Everything the other characters in the system do needs to be captured by either action, event or both. These concepts are then stored in the memory. Also, to promote the debugging support, all the actions and events that occur during the agent's lifetime are stored, e.g. by saving them to the disk.

The next Section describes how the Decision Making System works in a greater detail.

2.5 Decision Making System

The Decision Making System in EWA works as follows (Fig. 2.6): An event occurring in the environment is first processed by the *Reactive Factories* and *Story Controller*. These components are managing the so called *Goal Stack* where a list of all the goals that should be performed by an agent is stored. Every goal has a fixed priority that is used by the Decision Making System to decide on which of the goals should be scheduled to run (see Alg. 2). When a Goal is scheduled to run, it becomes an Active Goal and based on the Goal specification, an intention is given control over the agent's behavior. The intention logic is typically implemented by the FSM and selects an action that should be performed by the agent to achieve the success condition of the goal. The action can make the agent move by calling the Navigation Manager component and it can control the agent by playing an animation on its virtual body or displaying an emoticon[6] in the environment.

```
for Goal g in Goal Stack do
    g.evaluate();
    Remove Succeeded or Failed Goal;
end
ActiveGoal = pickHighestPriorityGoal(Goal Stack);
if ActiveGoal == PreviousGoal then
    ActiveGoal.logic();
else
    PreviousGoal.freeze();
    ActiveGoal.logic();
end
PreviousGoal = ActiveGoal;
```

Algorithm 2: Decision Making System Loop

There can only be one intention executed at a time and only one Active Goal at a time. Typically, multiple goals are in the Goal Stack and the DMS picks the Active Goal by looking at the goal priority. The goal priority is expected to be an integer ranging from 0 to infinity[7] that represents fixed priority of the goal. In

[6] Emoticon is a comic like bubble appearing over character's head – more details in Section 3.3.2.

[7] Though author recommends using a reasonable upper limit such as 200.

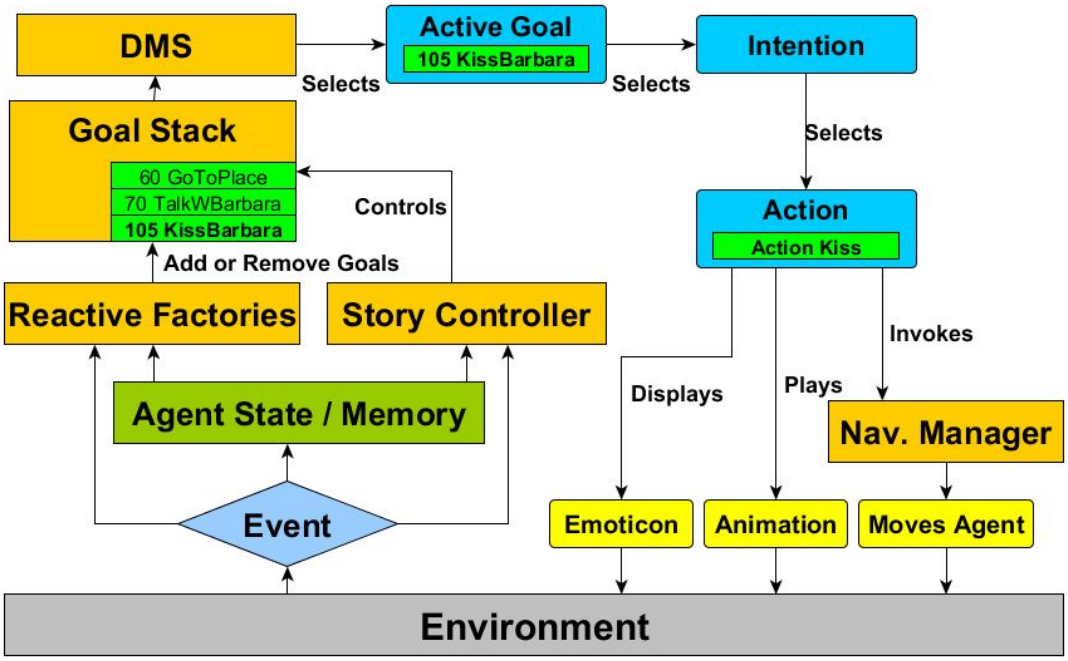


Figure 2.6: EWA Decision Making System Overview. The visualization of decision making components in EWA. Events are generated by the Reasoning Components that are not visualized. The numbers, e.g. "60 GoToPlace" represent the priority of a particular goal. See the detailed description of the process in this Section.

every tick, the DMS always selects the goal with the highest priority. If this goal was active in the previous tick, nothing special happens and the goal continues to be executed by continuing with the execution of the intention. If a different goal was active in a previous tick then a transition behavior needs to occur (if not defined the transition is instant). This works as follows – the previous goal is suspended and currently running intention is notified a new one will be scheduled to run by receiving a *freeze* signal and starts to perform the freezing behavior that can be specified in the intention. This behavior will usually last for several ticks of the architecture and the intention will continue to run until this is finished. Then the intention scheduled by the new Active Goal is performed. Note that the Active Goal can form a hierarchy of the goals, but for simplicity sake the fact is not explicitly mentioned as even for a hierarchy of the goals only one intention can be executed at a time.

Next, Reactive Factories (RFs) that handle managing of goals in the Goal Stack are explained in a greater detail.

2.5.1 Reactive Factories

Reactive Factories (RFs) are a concept introduced in EWA to encapsulate various reactions to the events in the environment. From a high-level perspective, RFs can be viewed as reactive if-then rules[8] that are monitoring the environment's state and are adding (or removing) goals that need to handle the situations that

[8]The whole concept was inspired by reactive planning approach as mentioned in Section 2.2.

occur in the environment.

In other words, RFs are places where the decision making logic of DMS is implemented. Depending on the number of monitored concepts in the scenario implemented by EWA, the number of RFs used can range from tens to hundreds. RFs can be turned on and off during the life time of the character, enabling dynamic changes in the character's reactions to the events.

An example of a RF is a factory that reacts to the situation when the lumberjack's wife appears and brings lunch. In this instance, the RF will add a goal "greet wife" and "eat lunch" to the goal stack, so the lumberjack should react appropriately to this new encountered situation.

The RFs are capturing reactions of a single agent to events in the environment and every RF can modify the DMS of only one particular agent. In a normal IDS application, more agents are usually present. Also, if the IDS system is required to generate and maintain a particular story, a need for a synchronization of agent's behaviors arises.

While RF can be used, e.g. to add a particular goal to the agent's goal stack when a particular situation in the environment occurs, they cannot be used effectively to synchronize the behavior of multiple agents, as one RF can modify the goals of only particular agent it is bound to.

2.5.2 Story Controller (Drama Manager)

The Story Controller (SC) is an object used to synchronize the behavior of multiple agents in the system. From the technical perspective, SC is the main component that "ticks" the logic of all the agents in the systems and loads the story definitions. For multiple agents in the system implemented by EWA, there is only one SC that is synchronizing the behavior of the agents and also maintaining the story progression.

If a certain situation arises – e.g. all characters in the story meet – SC can "hijack" all character's behaviors, clear their goal stacks and schedule a goal necessary to handle the situation. The concept of SC is rather useful as it allows the encapsulation of the code that handles the conditions on a more global level than RFs would. Note that SC can be viewed as a drama manager component as discussed in section 1.1.3).

Representing the story

From the "strong story" versus "strong autonomy" point of view, the EWA architecture is leaning towards the "strong autonomy" and is character-centric. To represent and maintain the story (R1), EWA offers three key elements to the designer.

First, the designer starts with capturing the basic story shape by using the Reactive Factories for scheduling behaviors with a known time of execution, e.g. the designer can set the behaviors of the lumberjack to a) go cut down trees in the morning and b) return home when it gets dark.

Second, the designer defines reactive behavior with RFs that are monitoring the agent or the environmental state and generate/remove behaviors accordingly. This mechanism enables two things: executing reactions to some events, e.g. by adding the "greet wife" behavior after she appears, and executing story-important

behaviors that do not have a fixed time of execution, e.g. "turn right" after the character arrives at a particular crossing. The former may also trigger a short sequence of follow-up behaviors (after greeting his wife, the lumberjack will probably engage in a small talk with her).

This mechanism also allows for generating (removing) the *future-directed* goals such as "go pick up strawberries for the lumberjack's wife" if she asks for it later in the day by scheduling the "pick strawberies" goal later in the future.

Third, the architecture features a *story Controller* component that allows for synchronizing agents and drama management. The Story Controller can change the overall story shape by removing or adding goals from/to the stack of a particular agent at important story points, such as when a couple breaks up.

The architecture is also integrating the Navigation Manager from [Popelová et al., 2011] and [Brom et al., 2012] to allow for specific movement setup required in dramatic situations, such as when three agents are arguing with each other in a virtual quarrel-like scenario or simply using this manager to steer the characters when they are walking along each other.

2.6 Reasoning in EWA

The EWA character perceives the world through events. Events generated by the game engines are typically low level – such as "a character appeared" or "the agent picked up an item". When simulating behavior in social situations, there is a need for more high-level events such as "a friend or foe appeared" or "somebody did something our agent despises". In decision making, it is much easier to work with high level concepts, e.g. a negative action was performed by someone.

In EWA, the generation of events is handled by the *EventGenerator* component that processes all low level information from the environment and converts them to high-level concepts, called events and actions, that the agent understands. That can be simple, such as monitoring the chat console in the system – when a certain text is detected, representing for example that someone has just told us a joke, an appropriate action (Action Joke) in EWA is created. Or even a more complex situation, such as using the emotion model to process events in the environment, generating events such as "agent B kissed agent C" and since our agent A has positive feelings towards C, there will be some negative emotions associated with this event (OtherAgentKiss).

The processes above are a form of a rudimentary reasoning that is performed by the *EventGenerator* and is a necessary component of any higher-level agent architecture.

2.7 Emotion Model

In human social relationships, emotions play an integral role. Simulating dramatic situations or simply everyday life situations featuring IVAs in 3D environments without explicit notion of emotions could be problematic as discussed in chapter 1. Now, developing a plausible emotion simulation is not an easy task and would warrant an extensive research or another PhD thesis. For this reason it was decided to integrate an already implemented emotion model ALMA [Gebhard,

2005] that is built upon the OCC theory of emotions [Ortony et al., 1988]. As the support for affect simulation is one of the key elements of EWA, the OCC theory of emotions is presented in a greater detail, followed by the emotion model ALMA overview and afterwards summarized with EWA and ALMA integration.

2.7.1 OCC Theory

The emotion theory of Ortony, Clore and Collins (OCC) was introduced in 1988 [Ortony et al., 1988] and falls into the cognitive/appraisal theories of emotions. This theory defines 22 distinct emotions that fall into three main categories. A notable fact about this theory is that one of the use cases of this theory was to design it in a way that it could be implemented in computer systems [9].

The OCC theory assumes that emotions are a result of certain sequences of cognitive appraisals and interpretations. According to the authors of this theory, emotions are valenced (meaning positive or negative) reactions to events. Their purpose is to evaluate events happening in the world. There are three main groups of emotions in the OCC model:

- **Emotions as reactions to events.**
- **Emotions as reactions to actions of agents.**
- **Emotions as reactions to objects.**

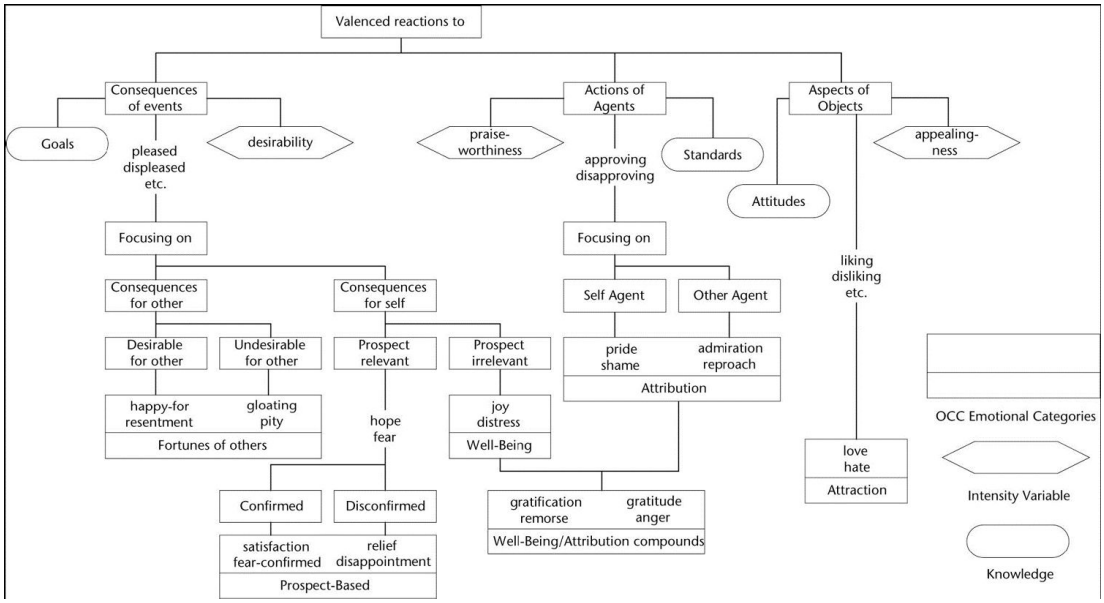


Figure 2.7: Structural part of the OCC theory. Notice the three main categories of the emotions in the rectangles at the top of the figure – consequences of events, actions of agents and aspects of objects. The most important variables are visualized with hexagons – desirability, praiseworthiness and appealingness. The emotion hierarchy is visualized by the sub trees. Figure reused from [Bartneck, 2002], page 1. Copyright Bartneck 2002.

[9] Note that some of the emotion theories while sound and interesting do not give the developer much to work with when it comes to implementation.

Notice the emotions here are always associated with something – they are reactions to something. You cannot just feel angry without associating that with a concrete action of some agent in the environment. Also, the theory understands emotions more as a short-term affect lasting in the range of seconds to minutes. So, this theory is not concerned with feelings or moods and the way these influence emotions. Also, this theory is not concerned with how the emotions affect each other.

The OCC theory specifies a set of variables that are used to appraise events in the environment, see Fig. 2.7. These variables can be either global – affecting all the emotions or local – affecting just a certain emotion category. The theory also gives a general guideline according to which the intensity of these variables should be set. The three main variables used in the theory are:

- **desirability** – measures the outcome of an event (from 1 to -1, pleasant or unpleasant). It captures how desirable the event is for the agent. The value of this should be assigned according to the agent's *goals*.
- **praiseworthiness** – measures attitude towards an action of a different agent (from 1 to -1). It expresses whether the agent approves or disapproves of the action. The value of this should be assigned according to the agent's *standards*.
- **appealingness** – measures personal attitudes toward objects. Defines whether the agent likes or dislikes a particular object. The value of this should be assigned according to the agent's *attitudes*.

Moreover, the theory defines a set of if-then rules that are working with these variables and result with a set of emotions associated with a certain event, action or an object. Note that one event, action or an object can have more emotions associated with it. Also, some events in the environment can be appraised from multiple perspectives e.g. let's say a person is playing football and someone has tackled her in a way she fell down and now her knee is hurting. That situation would get appraised as an event – not a desirable one. Also someone tackled her – another agent. She will also appraise this as an action of that agent – praiseworthiness will be low, she will probably get angry. Now these multiple appraisals need to be implemented in the system itself. The theory just gives guidelines that need to be implemented by the developer, not the actual implementation.

Note that there are some well-known shortcomings of the OCC model [Bartneck, 2002]. The theory does not solve emotions decaying over time, interactions between emotions or the physiological part of the emotion phenomenon. Some of these problems are being addressed by [Steunebrink et al., 2009] or [Gebhard, 2005]. Nevertheless, the OCC is probably one of the most widely applied emotion models used for affect simulation to date.

While the affect simulation is a necessary part of a *believable* IVA architecture, implementation of our own plausible emotion model was out of the scope of this thesis. The following Section presents the properties of the OCC based emotion model ALMA [Gebhard, 2005] that are integrated with the EWA architecture.

2.7.2 ALMA Overview

ALMA (A Layered Model of Affect) [Gebhard, 2005] is based on the OCC cognitive model of emotions [Ortony et al., 1988]. ALMA features three types of

affect that differs in the means of duration. It provides emotions for a short-term affect, moods for a medium-term affect and personality for a long-term affect. ALMA makes it possible for IVAs to be able to react to events in the environment with emotional reactions, so it captures for example what made the agent *angry*. However, ALMA does not specify how these emotional reactions should be used to alter the agent's behavior. Also, the mapping of environmental events to ALMA inputs is pretty much in the hands of the developer. This enables, on one hand universal use of this model, on the other hand, the user will always need to provide a mapping from the environment to ALMA input as well as to create the actual changes in agent's behavior according to ALMA's affects. ALMA also features the GUI that can be used to inspect current character's emotions and moods or set them to particular values. Next, ALMA emotions, mood and personality constructs are presented, followed up with the ALMA input specification and lastly the EWA and ALMA integration is discussed.

Emotions. ALMA defines 24 OCC-based emotions – see Table A.2) for a complete list of ALMA emotions. Emotions in ALMA are used for a short-term affect and are always attributed to a certain event or action in the environment. The emotions in ALMA are decaying over time, being able to last for a maximum of a minute or so[10]. ALMA emotions were used to compute the *feeling* value representing the social relations between the characters in EWA.

Mood. The mood in ALMA is based on Mehrabian's dimensional theory of emotions [Mehrabian, 1996]. It is defined by three dimensions – pleasure (P) that defines how positive or negative the mood is, arousal (A) that defines how activated the character is, and dominance (D) that defines how much "in control" the character feels. These dimensions then define 8 mood octets the character can feel (see Table A.1). The mood is changing based on the emotions of the character but slower – in a time scale of minutes.

Personality. Personality in ALMA uses the OCEAN (or *big five*) model described in [McCrae and John, 1992]. The personality affects how the agent reacts to events, actions and objects when processing the emotion variables. This means that the same values of variables can lead to a different emotion outcomes with a different personality. The agent's default mood is also affected by the personality. The five dimensions of the OCEAN personality model are as follows:

- *openness* – reflects interest in intellectual issues, unconventional values, aesthetic sensitivity, need for variety.
- *conscientiousness* – reflects task-oriented characteristics such as being dependable, responsible and orderly.
- *extraversion* – reflects a tendency to be sociable and experience positive affect.
- *agreeableness* – reflects a tendency to be inter-personally pleasant and compliant.
- *neuroticism* – reflects a tendency to experience anxiety and other negative emotions.

[10]Though this can be changed by the developer.

The personality of the character can be set but then it stays fixed.

Emotion Eliciting Conditions. Input to ALMA is provided through the *emotion eliciting conditions* (EECs). The Emotion Eliciting Condition (EEC)s are a sub-set of OCC theory variables. List of EECs variables (ranging from -1 to 1) follows:

- **desirability** – used to measure an event outcome, if desirable (in accordance with agent's goals) the desirability will be set to positive and vice versa
- **praiseworthiness** – used to measure other agent's actions, if our agent approves the action, it is set to a positive value and vice versa
- **appealingness** – how much an object or agent that is taking a part in a recent event is liked
- **likelihood** – used to measure prospect based events – how likely the event will happen
- **liking** – how much is the agent is attracted to another agent
- **realization** – when some prospect based event occurred here is set how much it was realized
- **agency** – can be set to "self" or "other" providing information about whether this event or action is caused by the agent or by another agent
- **elicitor** – who caused this event or action. May be some other agent or some other object (e.g. an environment).

When the event or action is annotated by EECs the input is then sent to ALMA emotion model via API specified by the ALMA interface. This is then processed by a set of ALMA emotion generating rules and a set of OCC emotions is generated. These are then internally processed by ALMA to get the change in the mood and are also sent back, so a list of OCC emotions and an event or action can be associated together. Note that the ALMA model keeps its own list of currently active emotions of the character. As there can be more events eliciting one type of emotion, e.g., joy – always the highest emotion value is visible in ALMA for each emotion category (joy, anger, fear, love, etc).

2.7.3 EWA and ALMA Integration

The EWA architecture implements an Emotion Model component that is tightly integrated with ALMA. The EWA's Emotion Model defines a set of events and actions that are recognized by EEC Factories and are submitted to ALMA model for evaluation (see Fig. 2.8). From the technical perspective, EEC Factories are object factories listening to events in the environment that are generating annotated event data that are processed by ALMA emotion model. The result of this evaluation is a list of OCC based emotions associated with the event. This is stored in the characters memory and can be used by decision making process as additional attributes influencing the decision making. EWA supports full range of ALMA OCC emotions, moods and the personality.

Moreover, EWA defines additional affective variable called *feeling*. *Feeling* ranges between 1 and -1 and represents social relation between two characters – as in "do they like each other or not". This attribute resembles the pleasure

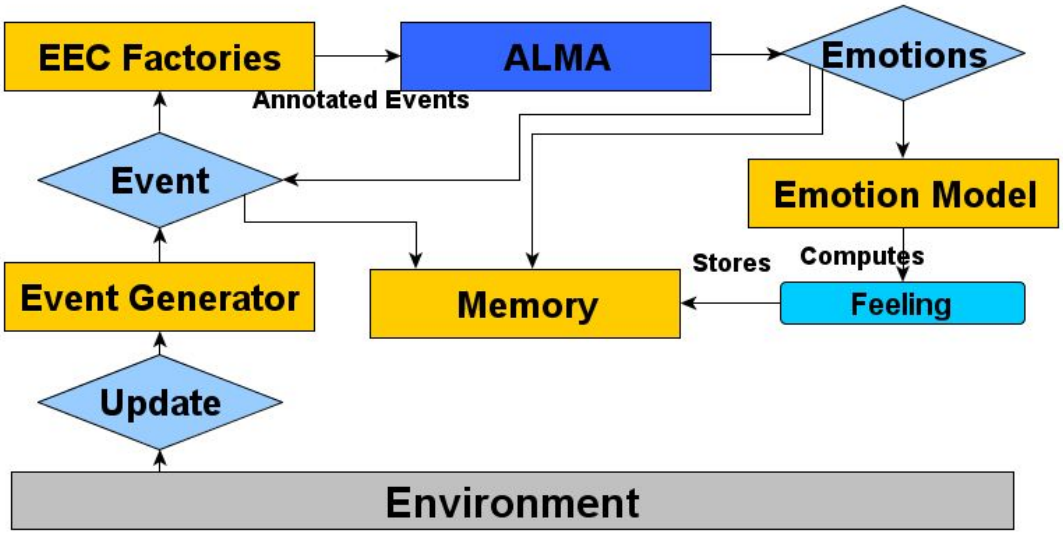


Figure 2.8: EWA Emotion Model Overview. An update in the environment is processed by the Event Generator. The resulting event is annotated by the Emotion Eliciting Conditions Factories that are part of the Reasoning Component. This is forwarded to ALMA which returns a set of resulting OCC emotions. These emotions are associated with the event and stored in the memory. The emotion model than computes a feelings value for this event and a possible change of the feeling values between the characters occurs.

dimension from dimensional theory of emotions but in the EWA architecture it is attributed to a particular character. Hence, one character in the system has a feeling value associated to every other character in the environment. The feeling concept was introduced to capture social relationships between characters in the environment.

Positive	**Negative**
Joy	Distress
Gratitude	Disappointment
Liking	Disliking
Love	Hate
Admiration	Fear
HappyFor	Anger

Table 2.1: Feeling emotions. Positive and negative emotions used to compute feeling value of an event. Note that the emotion intensity values range from 0 to 1.

To compute the value of a *feeling* value of an event in the environment the resulting OCC emotions associated with the event are processed. A subset of ALMA emotions is used and split into two categories – positive and negative emotions (see Table 2.1). The default *feeling* value between characters is set in advance (this depends on the particular scenario that is being simulated). Every time a new event or action occurs in the environment, the following formula 2.1

is used to compute the feeling value of the event and afterwards, according to this value, the feeling values between the characters are adjusted[11].

$$feeling(event) = \sum_{i=1..n}^{PositiveEmotions} \frac{Intensity_{(i)}}{n} - \sum_{i=1..n}^{NegativeEmotions} \frac{Intensity_{(i)}}{n} \quad (2.1)$$

The resulting *feeling(event)* then shifts the feeling between two characters in the following way. Firstly, a new target feeling value is computed based on the previous feeling and the change introduced by the feeling of an event. Secondly, that value is used to slowly shift the feeling between the characters to the new value. For exact algorithms see Algorithm 3 and 4 in the following Chapter.

Note that the feeling between characters is not symmetric – character A may like character B but not vice versa.

Also note, that the event and action evaluation by EECs is done by the EWA architecture. ALMA is not defining how should the developer do that and the actual settings will influence the model significantly. Author best judgment is used to annotate the events and to tweak the parameters to suit the needs of the applications designed.

2.7.4 Other Computational Models of Emotions

Emotion model ALMA was selected over other alternative models as it matched the best to EWA's use cases. Advantages of ALMA are that it is implemented in Java making it easy to integrate with other Java applications. Also, ALMA is universal and not bound to particular environment and offers robust, psychologically plausible simulation of several affect types.

Next, a brief overview of other computational models of emotions used for IVA and NPC's affect simulation is presented.

The Cathexis emotion model [Velásquez, 1998] is inspired by neuropsychology, AI and ethology and views emotions as concepts arising from drives such as hunger. Overall, it seems Cathexis represents low level concepts that are not required in EWA.

The FLAME (Fuzzy Logic Adaptive Model of Emotions) [El-Nasr et al., 2000] is an appraisal based model inspired by OCC theory consisting of three components – learning component, emotional component and decision-making component making the model tightly coupled with the simulated NPC behavior of a pet dog.

EMA (EMotion and Adaptation) [Marsella and Gratch, 2009] is an attempt to create a general computational model of emotions based on the appraisal theory of [Smith and Lazarus, 1990]. It was used to design and control behavior of conversational IVAs[12].

Broader general overview of computational models of emotions is available in [Marsella et al., 2010].

[11] Presuming the event was caused by another character.

[12] Conversational IVAs are also referred to as Embodied Conversational Agents (ECAs).

2.8 Navigation Manager

This Section describes the Navigation Manager that was implemented in [Popelová et al., 2011] and [Brom et al., 2012] and that was integrated with EWA architecture. The navigation manager allows the developers to solve a complex task of navigating IVAs in 3D environment with regard to their social relations and dramatic situations the IVAs might find themselves in.

2.8.1 Motivation and Related Work

Navigation and movement are important topics in AI and computer games. For every *believable* agent it is absolutely crucial to be able to navigate effectively in the virtual environment. This gets even more complicated when simulating social context of movement. One can easily imagine that a lovers will walk along as a couple differently than friends. How to simulate these kinds of rich behaviors when it comes to movement?

Craig Reynolds defines *steering* techniques as an approach to move entities in the 3D environments [Reynolds, 1999]. The entities can be of various types – inanimate objects such as cars or projectiles, animals such as herd of sheep or flock of fish [Reynolds, 1987] and even human-like NPCs [Reynolds, 1999]. Reynolds steerings are based on a three level architecture. The top level tackles the high-level decision making process that should decide "where to go" and retrieves high-level path from the environment, the middle layer specifies reactive behavior that is used to steer the entity according to the local information in the nearest surroundings and the lower layer solves the problem of the locomotion – such as which animation to play – and is usually implemented in the underlying engine (e.g. a computer game).

Reynolds steering techniques present deterministic and computationally cheap approach to navigation of entities in 3D environment and are used, for example, in crowd/pedestrian simulations, traffic planning, computer games and movies.

There are number of works that are using Reyonolds steerings for NPC or IVA navigation. [Guy et al., 2010] and [Karamouzas et al., 2009] use steerings for avoiding collisions with other characters or static obstacles. [Rojas et al., 2013] uses steerings to model realistic corner turning, [Park et al., 2013b], [Park et al., 2013a] and [Karimaghalou et al., 2014] are using steerings for simulation of small groups of agents in social situations.

Group conversation dynamics and human territorial behavior during social interactions has been researched also by [Pedica and Vilhjálmsson, 2008], [Pedica and Vilhjalmsson, 2010], [Pedica et al., 2010] and [Ricks and Egbert, 2012]. Spatial behavior inside small groups of agents has been tackled by [Karamouzas and Overmars, 2010] and [Peters and Ennis, 2009]. [Jan and Traum, 2007] modelled spatial behavior of agents that are conversing with each other in a small group including aspects as turning, re-positioning and joining and leaving the group. Small group conversation was researched also within BeerGarden project [Damian et al., 2011].

EWA integrated *steering* techniques to simulate socially plausible behavior of IVAs in general [Popelová et al., 2011] and with regards to plausible positioning of IVAs during a virtual quarrel of three agents [Brom et al., 2012]. The

used approach differs from previous work in two ways. Firstly, the implemented *steering* techniques are coupled with EWA architecture allowing for, e.g., using the emotions to directly influence the *steering* layer and secondly, while some of the previous work integrated *steerings* with an agent architecture, their goal was to simulate behavior of crowds and groups with regards to movement. Approach used here is concerned not just with movement but with general scenarios exhibiting agents in various social situations within medium-sized drama. Moreover, the steering techniques implemented by [Popelová, 2011] were built on top of the Pogamut platform [Gemrot et al., 2009] making the integration with EWA easy.

2.8.2 Steerings

[Popelová, 2011] (in Czech)[13] implemented three-layer steering architecture (Fig. 2.9) based on Reynolds steering techniques [Reynolds, 1999] for controlling the movement of NPCs in 3D environment.

The architecture is composed of a navigation layer, a steering layer and a locomotion layer. The top navigation layer decides which steerings should become active and what their parameters should be. The middle steering layer is responsible for computing the velocity vector of the agent that should be set next time it is asked for an update. The lowest locomotion layer is then responsible for playing the correct animation on the characters body and for moving the character in the environment.

The steerings use only local information – current position of the agent, current velocity of the agent and the same information for other visible agents. Steerings detecting static obstacles such as walls or trees are using a set of rays that can be configured (number or rays, their angles and length). More steerings can be combined together using the velocity vectors they produce. The next velocity vector of the agent is then computed as a sum (weighted) of all active steerings combined with the actual velocity vector the agent has.

EWA architecture supports the following eight *steerings* (descriptions based on [Popelová, 2011]):

1. **Target Approaching** – basic steering that makes IVA to reach a certain point in the environment, usually used together with Obstacle Avoidance steering. Based on [Reynolds, 1999].
2. **Obstacle Avoidance** – basic steering that steers the IVA away from static obstacles such as building or trees. Based on [Reynolds, 1999].
3. **Path Following** – basic steering that steers the agent, when the agent is following a path in the environment represented by interconnected virtual navigation points. Based on [Reynolds, 1999].
4. **Leader Following (LF)** – basic steering that makes the IVA follow another IVA in the environment. The LF steering allows for setting the agent's relative position to the leader, which is our innovation to [Reynolds, 1999] version of this steering.
5. **Wall Following** – basic steering that makes the agent to walk along a wall in the environment. Based on [Reynolds, 1999].

[13]The author of this thesis was the supervisor of the bachelor thesis [Popelová, 2011].

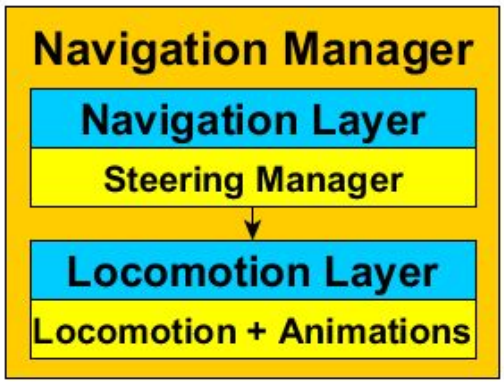


Figure 2.9: General Overview of Steerings Architecture. Navigation Layer is responsible for high level decision making – obtaining the path the IVA wants to go, *steering* manager is responsible for steering the agent around obstacles and other agents and also maintains "social" constraints, e.g. when the agents are walking as a couple, locomotion layer is responsible for moving the agent and playing correct animations at proper times. Figure based on Figure 1 from [Brom et al., 2012].

6. **People Avoidance** – social steering that makes the agent to steer around other agents in the environment. A similar approach as in [Karamouzas et al., 2009] was used.
7. **Walk Along Steering** – Walk Along steering implemented to simulate a behavior of couples walking together is detailed in [Popelová et al., 2011].
8. **Triangular Steering** – social steering used to position three agents communicating with each other. This steering was used to simulate a virtual quarrel between three agents [Brom et al., 2012].

The overall *steerings* architecture is detailed in [Popelová, 2011].

2.8.3 Steerings and EWA integration

Navigation Manager component implemented by [Popelová, 2011] is integrated with the EWA architecture. This enables the designer to steer agents with regard to their social relation, e.g. by Walk Along Steering. This is done on an action level. Any action that moves the agent can initialize the Navigation Manager with a set of parameters that define the required form of the movement. Moreover, these parameters can be influenced by the state of the agent – e.g. by the *feeling* value towards another agent. This can for example influence how close together are the characters walking.

Moreover, the EWA integrates also the *virtual quarrel* simulator from [Brom et al., 2012] that adds the triangular steering to the Navigation Manager component. There are more comments in the next Chapter on the integration of particular *steerings* and EWA on examples from two applications EWA was used to build: SD One and SD Two.

2.9 Summary

This Chapter introduced EWA – a minimalistic affect modulated control architecture for developing believable behavior of NPCs in IDS systems and hence solved the Goal 1 of the thesis. The architecture was described on a conceptual level and presents a step forward towards solving the *wicked problem* of medium-sized computer-game like IDS systems. In the following Chapter, the actual implementation of the architecture is discussed on use cases of SD One and SD Two games.

The architecture is an interesting orchestration and integration of several modules that are required when one wants to effectively develop NPC behavior in medium-sized drama scenarios. The architecture is somewhere in between simple mechanisms, such as finite-state machines, and complex solutions such as ABL language, *Comme il faut* or FAtiMA. It goes beyond the simple mechanisms in that it enables easily i) modulating behaviors by emotions, ii) representing transition behaviors, iii) believable character spacial movement in dramatic situations, iv) synchronizing the characters centrally and adjusting the whole story at important plot points and v) provides sound behavior abstraction. This is achieved by introducing minimal concepts necessary to specify the points above which should make the architecture easily usable by developers. The integration of i), ii), iii), iv) and v) in medium-sized drama use case is, as far as the author knows, novel.

From the behavior specification point of view, the architecture is also an interesting combination of BDI paradigm concepts (goals and intentions), behavior trees (used in the goal hierarchies), Finite State Machines (used in the intentions – more details in the next Chapter) and traditional programming language (Java was used to implement low level behaviors in actions and intentions).

By using appropriate concepts on appropriate levels of behavior specification the behavior design stays reasonably simple not to overburden the developer with complex intricacies of complex solutions decreasing the development time and allowing the developers to concentrate on the interesting part of the development curricula such as game design and plot structure instead of how to use the architecture to achieve their requirements.

Next Chapter introduces SD One and SD Two that served as a case-studies for evaluating whether the architecture works well and concludes with lessons learned from this endeavor.

3. Applications

This Chapter presents three applications that were created as a part of the research endeavor of this thesis – a tool for scripting short movies in 3D environment of a computer game[1] called StoryFactory, a short dating game SD One and a longer dating game SD Two.

To develop SD One and SD Two games, EWA architecture described in previous chapter was implemented. Moreover, connection between UE2 engine and Pogamut was implemented, graphical assets for UE2 were created and framework enabling the developer to package i) UE2 environment, ii) graphical assets for UE2 environment and iii) custom scenario implemented on top of EWA (e.g. SD One, SD Two or a new custom one) into one click installer making the development of new scenarios and/or games easy. That solves two thesis goals (G2. and G3.).

This chapter begins with a brief overview of the Pogamut platform that was used as a basis of all the other applications implemented in the thesis. Next, the StoryFactory tool is introduced, afterwards the SD One and SD Two are summarized concerning the implementation of EWA architecture. Lastly, a set of debugging tools that were implemented on top of EWA is presented. The chapter summarizes by lessons learned.

3.1 Pogamut

Pogamut[2] [Gemrot et al., 2009] is a Java framework for rapid prototyping of IVAs living in the 3D world developed at Faculty of Mathematics and Physics at Charles University.

The main features of Pogamut 3 include:

- a binding to the virtual world of the Unreal Tournament 2004 videogame (UT2004), StarCraft video game, Minecraft videogame, Unreal Development Kit environment and Unreal Engine 2 environment (implemented as a part of this thesis)
- an integrated development environment (IDE) with a support for debugging
- a library with sensory-motor primitives, path-finding algorithms, and a support for combat behavior suitable for first person shooter games
- a connection to the yaPOSH [Gemrot et al., 2013], which is a reactive planner for controlling behavior of agents with a graphical editor
- a support for running experiments and automatic tournaments between agents

Pogamut is used for teaching the basics of AI for IVAs in 3D environments at Artificial Beings course at Faculty of Mathematics and Physics, Charles University in Prague and is also used abroad, e.g. [Korstanje et al., 2016].

[1] These short movies in 3D games are also called *machinimas*.

[2] Note that the author participated on the development of the Pogamut platform by leading a grant project GA UK 0449/2010/A- INF/MFF with the said platform as outcome and also by implementing the GameBots platform part responsible for the low level API of the platform at the game level [Bída et al., 2012].

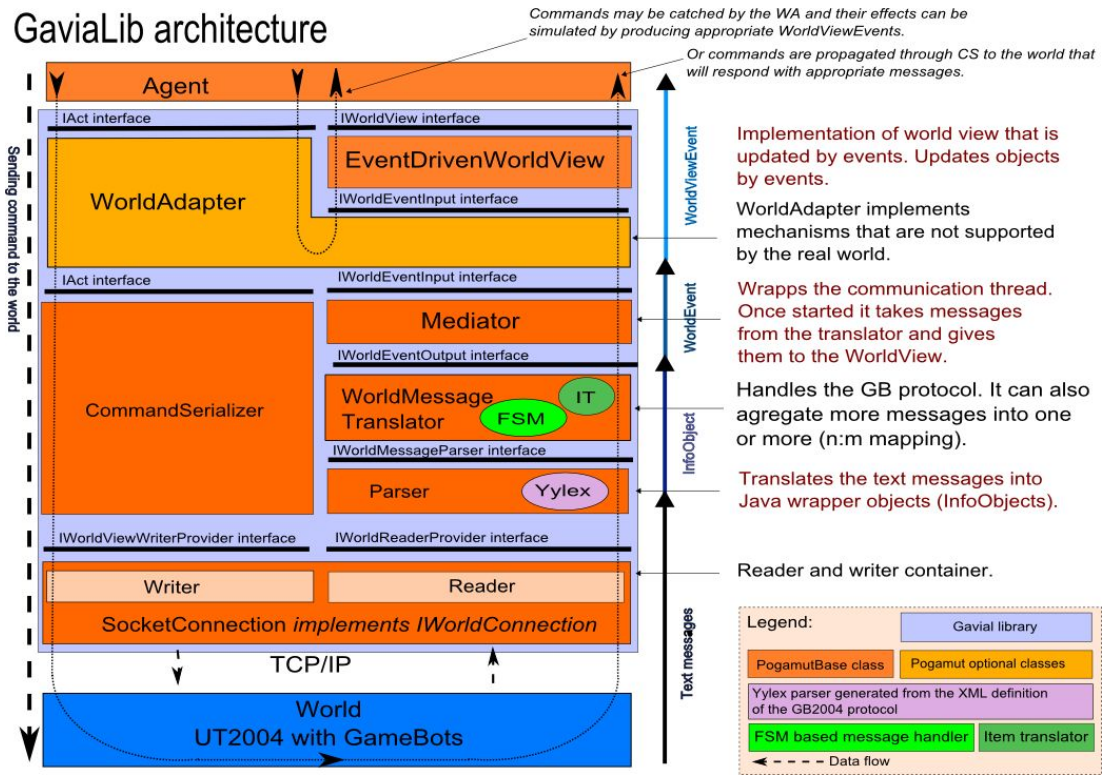


Figure 3.1: Pogamut 3 architecture overview. Notice the loop between the agent (up) and the environment (bottom) that resembles the typical reactive agent approach. Pogamut defines a couple of interfaces that makes the development of IVA easier for the developer as the agent – environment loop is handled by Java API and interfaces. Figure created by Jakub Gemrot, used with permission.

Pogamut is tightly coupled with Unreal Engine 3D environment allowing the developer to quickly prototype IVAs living in this world. Moreover, Pogamut defines Java API that can be used to get the information about the agent surroundings, e.g. the position or what the IVA sees and also to control its action, e.g. there is a function *move(x,y,z)* that makes the character move to desired location (if possible).

Pogamut with the extensions made as part of this thesis was used as a basis of the StoryFactory tool and the SD One and SD Two games.

3.2 StoryFactory

The previous chapters showed that designing *believable* scenes in computer games and IDS systems is a complex task. Designing such scenes often requires many iterations of trial and evaluation stages. To help to prototype various types of dramatic situations and also as part of our broader initiative on promoting the education in the field of computer science and ICT at high schools and universities [Brom et al., 2009], the StoryFactory tool [Bída et al., 2011b] was created. StoryFactory enables users to script short movies in a 3D virtual world. This allows for a) fast prototyping of dramatic situations in 3D environment and b) in an engaging way, this allows for introducing challenges posed by scripting 3D virtual characters and screenwriting. Apart from using this tool to prototype dramatic

situation this tool is supposed to be used in ICT and/or media education classes. Here, StoryFactory tool is presented along with first results from its evaluations.

Figure 3.2: StoryFactory 3D environment. The graphical content of StoryFactory in Unreal Engine 2. Figure is taken from [Bída et al., 2011b].

StoryFactory runs on a freely available version of Unreal Engine 2 (UE2). UE2 is 12 years old, making it more likely that the applications will run on usually slightly outdated school hardware[3]. The virtual city of EmohawkVille is about 500 x 500 m large. Dozens of virtual characters (including animals) are equipped with about 5–200 motion captured animations each and their surface textures may be alternated (e.g. color of their t-shirts, etc.). Most of the content was developed by a research group the author participates in[4]. The user controls the movement and animations of characters using a bird's-eye view map of the city and timelines (Fig. 3.3). The user can also use two cameras to allow for cuts and for capturing the situation from different angles.

Evaluation. This subsection is cited with modifications from [Bída et al., 2011b], section Evaluation, pages 3 and 4.

With regards to the educational goal (b), there were four small scale evaluations made. Firstly, during 2010 an informal evaluation of the graphical content with high school students during several lectures about 3D virtual characters taking place at Czech high schools was conducted. Results suggested that the graphical content was largely accepted by this usually very critical audience. Thus it was decided to use the virtual environment of UE2 and the graphical assets of EmohawkVille city and characters as the virtual environment where the IDS systems presented in this thesis are implemented (solving thesis Goal 2).

Secondly, an evaluation with high school teachers as part of summer school Lipnice in August 2010 (N = 29) was conducted. The results suggest that a) the background knowledge of Czech ICT teachers of 3D graphics and animations is minimal, but b) can be substantially improved in one 90 minutes long lecture and one 90 minutes long practical seminar with StoryFactory, and c) the acceptance of StoryFactory is positive (in several evaluative questions, the majority of teachers chose one of the two most positive scores on 5-point Likert scale). The third

[3]Note that there is also a version running in Unreal Engine 3 (UDK) environment.

[4]StoryFactory was developed as a software project on Faculty of Mathematics and Physics at Charles University in Prague. The implementation team was lead by the author of this thesis and comprised of Markéta Tomková (formerly Popelová), Jakub Tomek, Jan Havlíček and Jan Vyhnánek with consultants of Cyril Brom and Edita Bromová (formerly Dufková). StoryFactory graphical content was created by Ivor Diosi, Zbyněk Krulich and Michal Červenka using Mayang's Free Textures library: http://mayang.com/texturesMayang [1.11.2011] and RocketBox 3D Animations Package: http://rocketbox.de [3.3.2012].

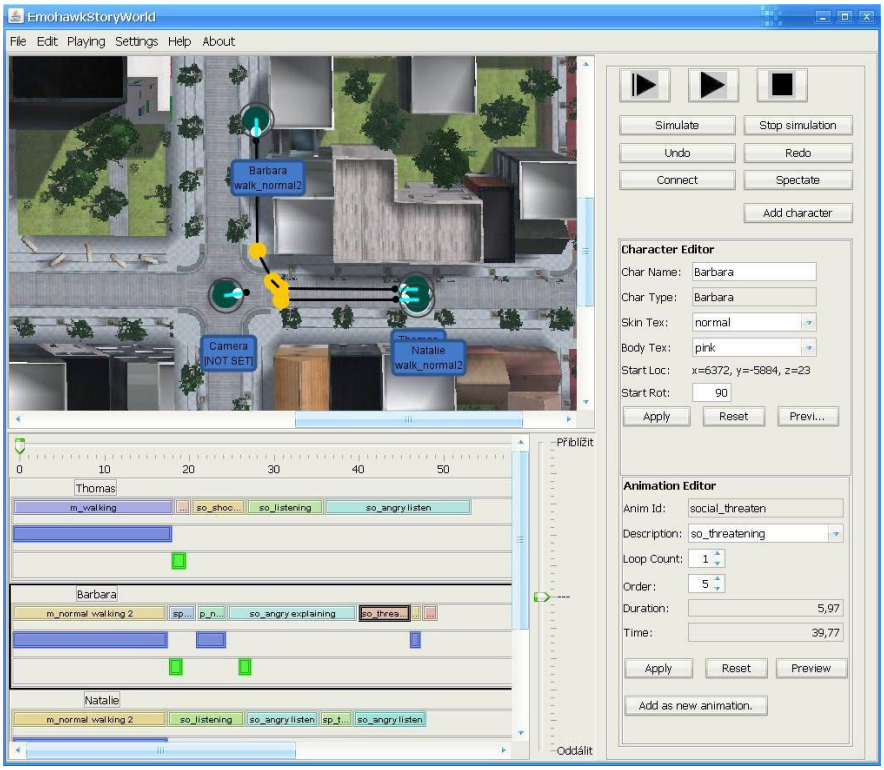


Figure 3.3: The StoryFactory's graphical editor. Notice the bird's-eye view of the city in the upper left corner and the animation timeline for each character and camera below. Figure is taken from [Bída et al., 2011b].

evaluation was conducted in November 2011 with high schools students (2 ICT classes, 24 participants altogether). The results indicate that a) the students are able to use StoryFactory effectively after a 90 minutes long tutorial, b) the acceptance of the tool by the target audience is positive, including the graphical content (one of the two most positive scores on 5-point Likert scale were chosen by the majority of students). Finally, in spring 2011 – a StoryFactory competition for small teams of high school students and teachers took place. The goal was to create the best short movie (up to three minutes) using StoryFactory running on Unreal Engine 2. The competition itself was successful with 20 short movies received (16 of them submitted for the competition).

The feedback from both students and teachers was positive in general. Notable comments are that a) even UE2 does not run smoothly on some school/home computers, yet some students would prefer to work with the UDK version, which is even more hardware demanding, b) students complained that they are missing some animations, but missed animations were surprisingly few.

Summary. StoryFactory is a tool allowing users to author short movies in a 3D virtual world using IVAs. The main audience of StoryFactory are designers interested in prototyping of social behaviors of IVAs in dramatic situation from the perspective of (a) and high school students interested in basics of 3D animations, virtual characters, screenwriting, and film editing from the perspective of (b). With regards to (a) the StoryFactor helps the designers to understand the problems of the dramatic scene she wants to design.

From the educational perspective (b), the StoryFactory objective is to help

teachers to elaborate on the topics of IVAs, 3D animations, screenwriting and film editing in detail using the tool, possibly attracting students to further ICT/new media studies. In that regard, it is not claimed that the StoryFactory is a silver bullet: students should be involved in other engaging and meaningful ICT-based activities beyond developing machinimas. Our tool differs from similar tools in that its content and the usage methodology are explicitly tailored to the high school audience. To our knowledge, a tool for that audience was lacking. The tool, including the graphical content, was positively accepted by both students and teachers, as demonstrated by our evaluations. StoryFactory was also used in spring 2011 in a new media class for undergraduates, who developed machinimas similarly to high school students. The tool is freely available in Czech language at `www.storyfactory.cz`, where the students' movies submitted for the competitions are available too. The English version of the tool is available upon request.

From the scope of this thesis, the virtual environment of StoryFactory is used the virtual world for SD One and SD Two. This solves the thesis Goal 2. Also the tool was used to prototype dramatic situations in SD Two such as virtual quarrel that occurs at the end of the game. From these prototypes a range of the behaviors the characters should support was identified along with a list of animations that were missing. With regards to virtual quarrel simulation in SD Two, where the goal was to simulated a dramatic situation where a boy is on a date with a girl and second girl he is also dating appears, there was conducted also a more profane research to identify missing pieces in the behavior of the agents and the supported animations. A group of actors and a director from DAMU[5] have been asked to improvise on the situation portrayed above and the material was used to further improve the virtual quarrel simulation – more details in [Brom et al., 2012].

3.3 SimDate3D Level One (SD One)

SD One[6] [Bída et al., 2011a] is a simple academic game featuring two characters Thomas and Barbara (see Fig. 3.4). The game is implemented with EWA architecture and takes place in 3D virtual city called EmohawkVille designed in Unreal Engine 2. The setting of the game is as follows:

Thomas and Barbara are friends, they meet by coincidence one day and one of them suggests they could go to the cinema. It is approximately two minutes walk to the cinema from where they met, but a lot of things can happen in these two minutes and whether they reach the cinema or not is in the hands of the player.

3.3.1 Game Overview

The player appears in the game as a ghost without the body observing the scene and has the opportunity to influence the behavior of the boy towards the girl. In case of no player input, the characters will be engaged in a casual dating

[5]The Theatre Faculty of the Academy of Performing Arts in Prague.

[6]The game is available at http://pogamut.cuni.cz/pogamut-games [16.12.2016].

Figure 3.4: SimDate3D Level One Overview. Upper left: an overview of the city. Upper right: Thomas performs "silly" walking. Lower left: Characters argue. Lower right: Thomas is angry. Figure is taken from [Bída et al., 2011a].

conversation; however, the player can make the characters argue with each other by making Thomas acting strangely. The story has three possible ends: either the characters make it to the cinema together or Barbara breaks up with Thomas or Barbara and Thomas decide to skip the cinema and go for a date instead. The exact course of the story unfolds and depends on the player's actions, the characters' current state and a limited random element.

Thomas and Barbara communicate through actions. Too many negative actions result in characters depart. No actions lead to boredom that leads to depart as well. Also, if the communication between the two is too much positive they skip the cinema and go for a date resulting in game loss. Moreover, each action costs some points. At the beginning the player has 6000 points available. More points left at the end of the game means higher score for the player.

EWA *feeling* value is used to represent the relationship between the characters and the current feeling value is visualized with colored balls around characters heads. Yellow balls means positive feeling, pink very positive feeling, green negative feeling and brown very negative feeling. The player needs to pay attention to this as the green and the pink colors mean there is a high probability the characters will not make it to the cinema. So to put it in another way the goal of the player is to keep the characters feeling values towards each other in the "middle" corridor – not too positive, nor too negative.

Game Controls

The player can move manually with arrow keys and the mouse. Left clicking the mouse focuses the view on one of the characters in the scene – this is the recommended way of viewing the game. The mouse wheel controls the observing distance and mouse movement changes the angle.

The player can make Thomas a) perform a positive or a negative action to interact with Barbara, b) increase or decrease his distance from Barbara, c) change the angle in which he is following her, and d) switch between a normal walking and a "silly" walking style. Barbara's reaction depends on her emotional state and the action of Thomas. For instance when Thomas starts walking silly, Barbara may ask him to stop. Her action may also trigger a Thomas' reaction, resulting in a short sequence of characters' interactions (triggered by the first action of the player). The following actions are available:

- Action Joke – mildly positive action (350 points)
- Action Compliment – very positive action (350 points)
- Action Blame – mildly negative action (350 points)
- Action Insult – negative action (350 points)
- Action Speak – neutral action, more topics available – weather, shopping, generic (300 points)
- To trigger a random positive action – the greater the feeling the higher chance more positive action is triggered. (250 points)
- To trigger a random negative action – the smaller the feeling value the higher the chance of a negative reaction being triggered. (250 points)

The player can also change their walking style:

- Switch between normal and weird walking – can result in both positive and negative answer. (250 points)
- Increase or decrease the distance between the characters. (50 points)
- Change the angle the girl is following the boy at. (100 points)

The fewer actions are used to finish the game, the higher score is reached. This mechanics helps to motivate the players to solve the game in an original way without spamming, i.e. chains of positive/negative actions in a row. For the list of the key controls, see Table C.1 in the appendix section.

3.3.2 Implementation

SD One was implemented using the EWA architecture. Characters exhibit eight different complex behaviors that are triggered by around 20 reactive rules. The behaviors are expressed by means of about 200 mocaped[7] animations, 50 emoticons and coloured bubbles above characters' heads expressing their overall feeling. One game play lasts around two minutes.

EWA agents are built on top of Pogamut platform that offers connection to the virtual environment and the low level API that allows the agents to get information about the surrounding world and to perform actions in the environment

[7]Motion capture is a technology of capturing animations from movements of real actors by the use of several high-frequency cameras capturing the scene from multiple angles.

– e.g. movement. The EWA architecture was used to implement decision making logic of the characters as well as the game mechanics. In particular, concepts such as goals, intentions, events and actions were used to implement the SD One scenario along with the affects defined by EWA and ALMA to simulate relations between agents.

In Pogamut, the decision making logic ticks typically four times per second. In other words, four times per second an update of the state of the environment is sent to the agent. That setting was reused in SD One as well. The Pogamut listeners that allowed for reacting to events in the environment were used as well. These events can, for example, be another character appeared or a character performed an action. Normally, the action pool in Unreal Engine is somehow limited. As there was a need to support additional actions – such as telling a joke – the chat channel that is provided by the engine to send text messages between characters was exploited to define additional actions (joke, compliment, insult, etc.). Implementation-wise, EWA listens to the chat channel and if a new message is detected, it is parsed and a respective event in EWA is generated. Note that the chat channel is by default invisible to the player. To display actions to the player, emoticon bubbles above characters' heads are used.

SD One features almost exclusively reactive behavior of agents – everything agents do is a reaction to a specific event, action or change in the state of the game (e.g. no action from the boy in 15 seconds). EWA Reactive Factories (RFs) offer an effective way of capturing such behaviors in game-like settings.

From technical point of view, RFs are implemented as object factories and listeners to certain types of events in the environment. In EWA, this might be actions of other characters, general events happening in the environment or even the changes in the agent's state of emotion. Example of an RF might be *ActionLeaveFactory* – this factory monitors whether the other character indicated that she wants to end the interaction with the current character. If that happens, the *ActionLeaveFactory* makes sure that the goal of handling the end of the interaction between the characters gets added to the goal stack. Also, this factory ensures that all the goals that are currently in charge of the interaction fail prior to adding a goal of making the characters depart.

Reactive factories offered an effective way of encoding the reactions to other actions and events. To implement these reactions, intentions were used.

Emoticons

To visualize the actions to the player *emoticons* were implemented. Emoticons are comic-like speech balloons that represent actions conducted by the characters (Fig. 3.5). For example, joke can be represented by a balloon containing the word "joke" or by a funny image such as "godzilla eating a kitten", compliment action is represented by a heart or thumbs up emoticon and arguing can be represented by a smiley face that is shouting. Using this approach, it was possible to define all of the actions of the characters explicitly mitigating the challenges – e.g. with NLU used on textual input.

Figure 3.5: SimDate3D Emoticons. Emoticons are used to represent agent actions by coloured comic-like speech balloons above character heads. Here, Thomas has made a compliment and Barbara responds positively by smiling.

Engine Level Modifications

As a graphical engine in SD One, a free version of UE2 Runtime [Epic Games, 2002] was used. UE2 offers developers 3D graphical engine capable of rendering rich 3D scenes. However, there is almost no content provided. In order to be able to implement SD One game, there was a need – among other things – to prepare the graphical assets in UE2. In particular, the virtual city of EmohawkVille needed to be developed and prepared[8]. Moreover, virtual bodies of Thomas and Barbara needed to be designed and created. This meant not only creating the 3D models and textures, but also their animations. Creating animations for virtual characters is a problem on its own. We have used RocketBox 3D animations package[9] and then motion capture technology to get additional animations that were not part of RocketBox and were unavailable for SD One use case[10].

Moreover, the game code needed to be modified[11] in such a way it allowed:

[8]The content created in scope of the StoryFactory project was re-used.

[9]Available at htpp://rocketbox.de [3.3.2012].

[10]Performed by Markéta Tomková (formerly Popelová) and cleaned by Michal Červenka.

[11]The programming work connected with issues above was conducted by the author of this thesis. Graphical assets were created by Zbyněk Kruchlich, Ivor Diosi with cooperation of

1. To connect UE2 Runtime with Pogamut platform. This meant to implement a low level API inside UE2 for exporting the agent surroundings from UE2 to Pogamut and also to implement the code that enabled performing commands of Pogamut platform in UE2. Eventually, additional environment of UE2 was added to Pogamut portfolio of virtual environments.
2. To develop a code that allowed us to play custom animations on characters' bodies and change the default movement animations. To address this use case, some parts of the StoryFactory implementation were used (these were implemented and reused by the author of this thesis).
3. To create a graphical emitter that was able to display coloured bubbles representing emotions of the virtual characters.
4. To create a code allowing for displaying emoticons speech balloons above characters' heads.

Affect-modulated Behavior

The behavior of agents is affect-modulated on the level of action selection. That is demonstrated when the agents are having a conversation with each other which happens every time the two agents are together and one of the agents initializes the conversation by any "speak" action – e.g. neutral action, telling a joke or paying a compliment. The conversation between agents is turn-based – every time one of the agents says something, the other agent responds. To compute the response action, a method defining a reaction to various actions based on probability is used. For example, if an agent was speaking about food, there is a good chance that another action representing food conversation will be selected. However, to make things less deterministic, a random variable is used to make the change of the topic possible – the agent switches to school for example.

Moreover, these reactions are modulated by the *feeling* representing the relationship between the agents. If it is too low, then even neutral actions such as conversation about sports will trigger negative reaction for example, blame or insult. The negative action is followed by the reaction of the conversation partner.

Feeling Implementation

This paragraph deals with the feeling computation in SD One and SD Two. Every time an event occurs in the environment, a feeling value of the event is computed based on the Formula 2.1. That is then used to change the feeling between the characters in the following way. Firstly, a new *target feeling* value between characters is computed by Algorithm 3. Then, the feeling between the characters is shifted slowly every tick (250 ms) towards the new value – Algorithm 4. This implementation ensures smoother changes in the feeling values of the characters.

Michal Červenka, Czech Technical University and Marketa Tomková (formerly Popelová).

```
MAX_VALUE = 0.15;
double eventFeeling = getFeeling(event);
Character C = getCause(event);
//first we get the old value
double oldValue = getTargetFeeling(C);
//then we compute maximum feeling change for this character and feeling
 values
double sign = Math.signum(eventFeeling);
double feelingMagnitude = 0;
if oldValue * sign > 0 then
    feelingMagnitude = Math.min(MAX_VALUE, 1 - Math.abs(oldValue));
else
    //boost increasing/decreasing feeling in opossite than current direction
    feelingMagnitude = MAX_VALUE;
end
//then we compute new target feeling value for this character
double newValue = oldValue + feelingMagnitude * eventFeeling;
setTargetFeeling(C, newValue);
```

Algorithm 3: Target Feeling Computation

```
FEELING_ADJUST_STEP = 0.005;
for Character C in Characters do
    double oldValue = currentFeeling(C);
    double newValue = targetFeeling(C);
    double sign = Math.signum(newValue - oldValue);
    double magnitude = Math.abs(newValue - oldValue);
    double adjustStep = sign * FEELING_ADJUST_STEP;
    if magnitude > 0.2 then
        adjustStep = 2 * adjustStep;
    else
        if magnitude < FEELING_ADJUST_STEP then
            adjustStep = newValue - oldValue;
        end
    end
    new_feeling(C) = oldValue + adjustStep;
end
```

Algorithm 4: Feeling Shift Algorithm

Technical Overview

Technically, EWA used Pogamut bot classes to represent agents in the environment. Bot class allows to easily register Pogamut *listeners* using Java annotations to various low level events in the game environment. Also, Pogamut offers classes that automatically spawn an agent in the environment and a way to describe textual API protocol between the engine and the Pogamut that is then wrapped with Pogamut Java objects. Extension to Pogamut platform was made to allow for connection of Pogamut to UE2 environment and to implement and wrap additional necessary commands such as spawning a certain item in the environment.

Initialization. *ScenarioLauncher* class is responsible for starting SD One game. This is done from the simple GUI that is implemented in Java Swing framework. This GUI constructs *GameController* class that is the descendant of *StoryController* and then calls ScenarioLauncher that invokes Pogamut internal methods that are able to spawn the bot class *EmohawkAgent* that represents the agent internally. EmohawkAgent class is responsible for initialization of all the modules an agent requires. These are especially *EventGenerator* processing and generating events, *DecisionMakingSystem* component that is handling the goals and *EmohawkAgentHistory* that serves as agent's memory and a kind of blackboard that is shared among all the components processing what happened to the agent in the environment. These classes are initialized inside EmohawkAgent class.

Logic Ticking. Method *logic()* is used in Pogamut as a standard place where the agent decision making system is implemented. Normally, Pogamut agents have their own Java thread that ticks the logic of the agent usually four times per second. However, in EWA it was decided to use a single threaded approach to ticking of the agents to get rid of the concurrency issues[12]. It is implemented through StoryController class that has a thread of its own. To do that, *ControlServer* agent was implemented in Pogamut. ControlServer provides a connection to the engine environment that does not spawn any agent but serves as a hook that can be used to get information from the environment and to control it. Also, this class receives updates N times per second (based on the settings) and uses them to tick the logic of all of the agents in EWA. ControlServer was implemented both on Pogamut and engine level.

Behavior Execution. The process that effectively leads to an action being executed by an agent in the environment starts in ControlServer. Every time the ControlServer receives an update from the environment, the logic of all the agents is ticked via the StoryController class. This leads to the DecisionMakingSystem component being ticked which checks the list of all the goals that are stored in GoalStack and picks the one with the highest priority. DecisionMakingSystem than ticks *logic()* method of a particular goal that defines what should happen behavior-wise. The goal needs to manage sub-goals and intentions. From Goal *logic()*, an intention *logic()* method is being invoked – there, an appropriate phase of intention lifetime is called and the decision regarding the next action is made.

[12]Note that single thread approach is preferred in game engines as well, e.g. [Unity Technologies, 2005], [Epic Games, 2004] and [Epic Games, 2002] all use single threaded approach when executing AI.

Note that as the actions can be executed for several ticks, the intention needs to check the status of the active action to see if there is a need to pick a new one. After the next action is selected, the *logic()* method of Action object is called. This invokes appropriate phase of Action lifetime. From there, the low level API of Pogamut is called to play animations on agent's body, display emoticon balloons or move the agent.

Scenario Specification and Installation. During the development of SD One, it became apparent that to make the testing of the environment easy, there is a need to be able to quickly deploy the code of the game into executable platform that can be run on any computer. Thus, a build script and set of settings were developed to secure a quick building and installing of desired scenario onto another person's computer. *ScenarioSettings* class is used to setup all the agents and the components of their decision making logic such as Reactive Factories. The build script then makes sure that all the graphical assets and UE2 engine are packaged into one-click installer the players can then use to install the scenario.

Finite State Machines (FSMs)

To represent complex behavior in intentions and actions, FSMs are used. It is done for several reasons. FSMs are explicit and easy to use. While they do not scale well, in EWA they are used for reasonably small tasks (one FSM is handling one execution stage of the intention – e.g. Fig. 3.6 and Fig. 3.9) to limit the occurence of the scalability issues. The most complex FSM used in SD One or SD Two has less than 20 states. This indicates that the behavior decomposition is sound and helps the system to scale well. From the technical point of view, custom FSMs using Java standard syntax were implemented (there are open source Java implementations of FSMs but for our use case including these seemed to be unnecessary).

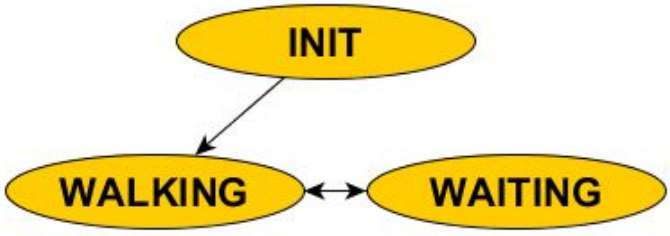


Figure 3.6: SimDate3D Level One – Simple FSM Example. Example of a simple finite state machine that was used when the characters were walking towards the cinema. When the distance between characters increased, the leading character entered the waiting state until the follower "caught up".

Leader Following (LF) Steering

One of the main game mechanics in SD One is the movement of the characters. Movement was implemented by using the Navigation Manager component taking advantage of the Leader Following steering from [Popelová, 2011]. From

implementation point of view, steerings are invoked from *PathFinding* class that provides an interface between actions defined in EWA and the Navigation Manager from [Popelová, 2011].

LF was used to steer the characters walking together to the cinema. This might seem as a counter-intuitive solution as [Popelová, 2011] introduced a dedicated steering solving exactly the problem of two characters walking along together (Walk Along steering). The reason it was decided to use the LF steering in this case was to introduce a new interesting game mechanics to the SD One where the user is able to influence the feeling between the characters by modifying the way the characters are walking together. While this can be also done with Walk Along steering, the possibilities are not that broad as with Leader Following steering, where the follower can walk in front of the leader enabling the user to create humorous situations in the game environment. More often than not, the goal of computer games is not the *psychological* plausibility of the behavior but a kind of *cartoonish* plausibility that still produces a *believable* behavior in the context of the actual game domain.

The LF steering is a basic steering where one agent is the leader and the other agent or a group of agents are the followers. The followers are attracted by forces to the leader but there are also repulsive forces that assure the followers "maintain" their distance from the leader. In the case of SD One, there were only two characters – the girl was the leader and the boy was the follower. This way the player could modify the properties of the LF steering and upset the girl by walking the boy too close or too far. There were two parameters the player could modify – a) the distance from the leader character and b) the angle the follower is following the leader at (from 0° to 360°).

If the user set up the parameters of LF steering in a way they produced "normal" behavior similar to Walk Along steering, the girl was reacting with positive emotions and vice versa.

3.3.3 Discussion

SD One served as a simple domain for evaluation of the basic concepts of EWA architecture. In particular, it was expected that the simple domain of SD One could be designed easily with concepts provided by EWA. SD One provided the first use case requiring EWA architecture which lead to the first EWA architecture implementation. Concerning user evaluation, there were many informal play session where users played and commented on SD One. The data were gathered for further analysis in Chapter 4.

Overall, it can be stated that the EWA concepts were sound and easy enough to represent the behavior of characters in SD One. However, the need for debugging tools became quickly apparent when developing SD One and so a set of debugging and logging tools that simplify the development of intelligent behavior in real time environments (see Section 3.5) was implemented.

As there was no need to use hierarchy of goals in SD One to capture the reactive behavior of agents, a one-to-one mapping between Goals and Intentions was used. From that, perspective it might make sense to use just one class for behavior specification. The *behavior* would implement both the success and fail condition of the behavior and the behavior itself that is now specified in intention.

This might be another addition to EWA architecture concepts to further simplify the process of behavior specification in the simple use cases.

The emoticon balloons turned up to be a nice abstraction of character actions visualizing the concepts of what the characters are communicating in a visually appealing and easily understandable fashion.

The game was also used as one of the domains for automatic drama analysis methodology introduced in Chapter 4. The game provided a good starting domain, but as the game goal can be achieved too easily, it was decided that to conduct more thorough analysis of the proposed methodology of automatic drama analysis, it is necessary to create a far more complex environment. This lead to creating SD Two introduced in the next section.

3.4 SimDate3D Level Two (SD Two)

SimDate3D Level Two (SD Two)[13] [Bída et al., 2013] is a sequel to SD One [Bída et al., 2011a]. There were two main reasons to create SD Two. Firstly, to show that the EWA architecture scales to domains larger than SD One. To prove it, a higher complexity use case was needed. Secondly, there was a need for a system capable of producing longer and dramatically more interesting stories as SD Two was supposed to be used as a second data source for evaluation of automatic drama analysis via the *tension curve* detailed in chapter 4 or in [Bída et al., 2013]. SD Two is situated in the same virtual city of EmohawkVille as SD One. SD Two is a character-centric IDS system capable of producing 5 to 15 minutes long game-plays[14] (or "stories") which is on par with other IDS systems.

The setting of the game is as follows:

Thomas, Barbara and Nataly live in a small town. Thomas has a girlfriend – Barbara – and... well... yet another girlfriend – Nataly. The girls don't know about each other. One day Thomas is on a date with Barbara when suddenly Nataly appears and things start to happen.[15]

In SD Two, the EWA architecture was combined with the steering behavior and with the virtual quarrel simulator [Brom et al., 2012].

3.4.1 Game Overview

The player is not directly present in the game but observes the scenario as a floating invisible entity (as in SD One). The game begins with Thomas and Barbara going to the cinema. The player has a limited control over Thomas' actions and the goal is to finish the scenario with the highest possible score. The score is gained by performing activities with the girls – such as going to the cinema or kissing them. To finish the game, the player has to "steer" characters so all of them meet and a virtual quarrel happens. The outcome of the argument depends

[13]The game is available at http://pogamut.cuni.cz/pogamut-games [16.12.2016].

[14]Note that there is not specified hard coded upper limit in the game. In theory, the player might play the game infinitely.

[15]Note that the game also features a symmetrical variant with one girl and two boys.

Figure 3.7: SimDate3D Level Two Overview. Upper left: Barbara refused a proposal of Thomas in the cinema. Upper right: Thomas, Barbara and Nataly at the beginning of a virtual quarrel. Lower left: Thomas is telling a joke and Barbara is laughing. Lower right: Thomas asked Barbara to go for a walk in the park and Barbara agreed.

on previous actions of Thomas towards the girls. There are four possible endings – Thomas breaking up with Barbara or Nataly, both girls breaking up with Thomas and all of the characters staying together in a "modern" relationship. The achieved ending type is shown in the middle of the screen when the virtual quarrel starts to occur.

When the player does not interact with the game, the characters continue to act on their own according to their current goal behaviors and emotional states. This natural behavior however results in boredom and indifference between the characters which is not desired by the player as it does not generate any score. Each intervention of the player in this natural story flow imposes a small score penalty. Therefore, to achieve the greatest score, the user tries to maximize the influence on the story flow with as few actions as possible. That constraint was added to demotivate the player to spam actions of certain type.

Places in the EmohawkVille

There are several activities the player can instruct Thomas to perform with the girls. These range from watching a movie at the cinema to going to the restaurant for a cup of coffee. These activities can be usually performed only at a specific place in the town. The list of all the places of interest in the EmohawkVille follows. Their location in EmohawkVille is visualized in Fig. 3.8.

- **Cinema** – marked by C on the map (characters watch a movie here)

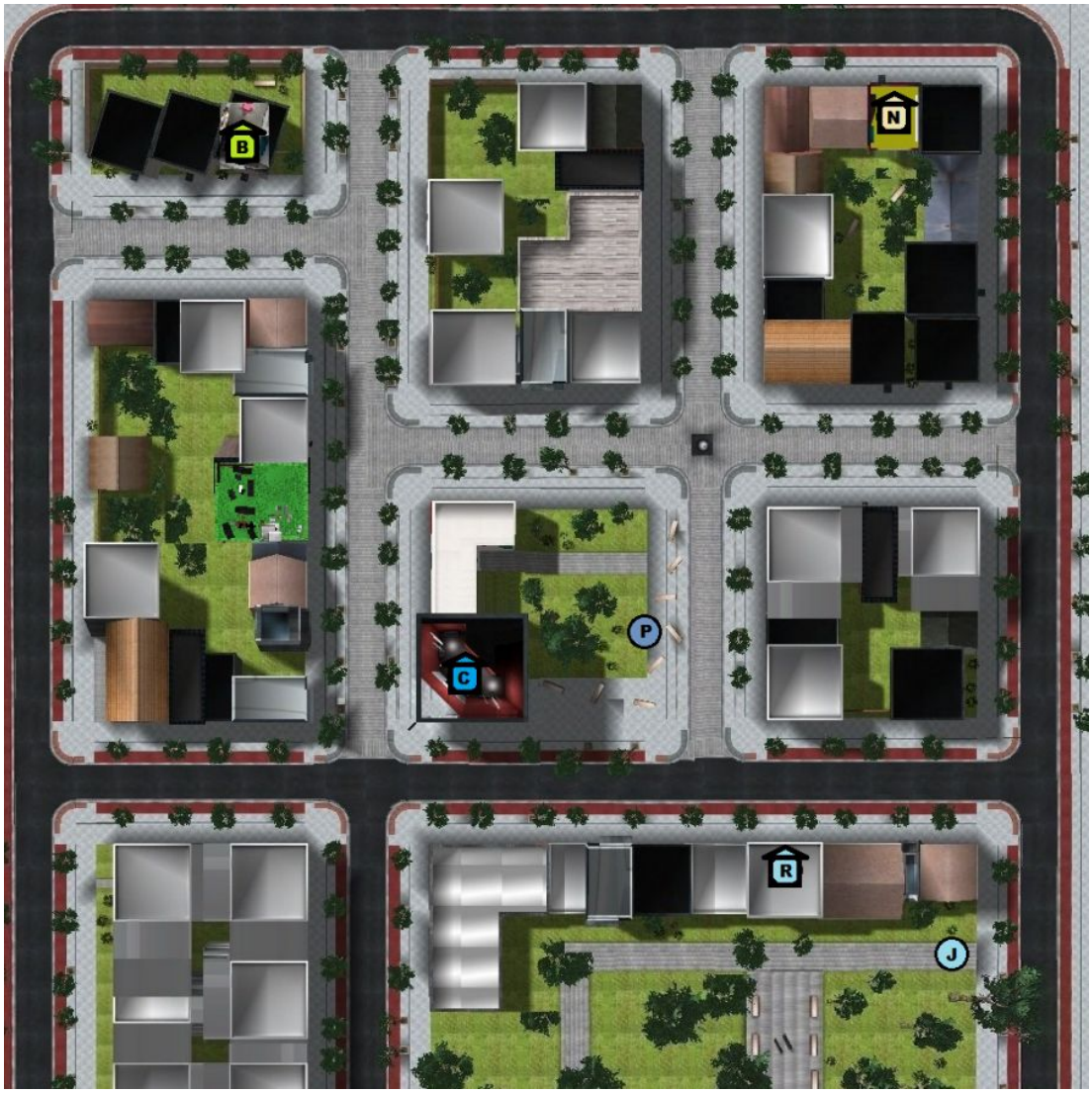


Figure 3.8: SimDate3D birds eye map overview. All points of interest are marked by icons. See the description in the text.

- **Barbara's home** – marked by B on the map (a good place to perform intimate actions with Barbara – the blond girl)
- **Nataly's home** – marked by N on the map (a good place to perform intimate actions with Nataly – the dark hair girl)
- **Restaurant** – marked by R on the map (a chatting place)
- **Central park** – marked by C on the map (a waiting place)
- **Monument park** – marked by J on the map (or "jogging" park – a sports place)

Game Controls

The player is present in the game as a ghost observing the environment and they can move manually with arrow keys and a mouse. Left clicking the mouse focuses the view on one of the characters in the scene. The mouse wheel controls the observing distance and the mouse movement changes the angle of view. For a complete set of game controls, consult Tables D.1, D.2, D.3 and D.4 in the

appendix.

There are three types of actions the player can force Thomas to perform. Most of them require a girl to be present and interacting with Thomas. Those three are the proposals that made by Thomas to ask the girl whether she wants to do some activity or an action with him – actions that trigger Thomas's reaction towards the girls such as telling a joke or changing Thomas' walking styles when walking together with a girl somewhere.

Proposals. The proposals are a new type of action that requiring the girl's consent which is not automatic. If the proposal to do an activity is rejected due to low girl's feeling towards Thomas, nothing happens and the interaction between the characters continues as normal. The player is able to use the following proposals in the game:

- Propose to go to the central park – if accepted, the characters will head to the central park.
- Propose to go to the jogging park – if accepted, the characters will head to the jogging park.
- Propose to go to the restaurant – if accepted, the characters will head to the restaurant.
- Propose to go to the cinema – if accepted, the characters will head to the cinema.
- Propose to go to Barbara's home (blonde girl) – if accepted the characters will head to Barbara home.
- Propose to go to Nataly's home (dark hair girl) – if accepted the characters will head to the Nataly's home.
- Propose a kiss – if accepted, the characters will kiss (successful kiss significantly raises score). Works only at home, cinema or restaurant.
- Propose a cuddling action – if accepted, the characters will cuddle (successful cuddling significantly raises the score). It works only when the characters are at home.

The girls react emotionally to all the proposals – accepted or rejected. Note that the proposals can be used also if Thomas is alone to relocate him to place defined by the proposal. This allows the player to move Thomas across the city. If there is a girl nearby, Thomas will propose to go to the desired place (this is a good way to start dating the girl that is passing by Thomas). If the girl agrees, they will head towards the destination. Note that sometimes the characters "finish" their previous conversation before heading out. See Table D.3 for the proposal key controls.

Actions. Actions available to player were changed from SD One to enable richer game-play. Each girl reacts differently to various actions, e.g. Barbara might like Thomas being bossy while Nataly might hate it. It is up the player to find out which action the girl likes the best and steer the interaction to use the actions that will satisfy the player's desired end game goal.

- Action Joke – tells a joke.
- Action Compliment – pays a compliment.

- Action Bossy – be bossy to the girl.
- Action Impress – tries to impress the girl.
- Action Insult – insults the girl.
- Action Speak – performs random neutral action.
- Action Goodbye – ends interaction with the current girl by saying goodbye. The girl and Thomas will depart and head home. It is useful when the player wants to start dating the other girl.
- Changing the walking style of Thomas[16].

See Table D.2 for the action key controls and Table D.4 for walking style key controls.

3.4.2 Implementation

SD Two was built on top of EWA architecture from SD One and features approximately 20 complex behaviors and 60 different actions the characters are able to perform[17]. Moreover, in SD Two, the Story Controller that is used to synchronize the character's goals and intentions when the story progresses was fully leveraged. It was necessary as the game logic of SD Two surpassed the one in SD One in every aspect. The game is longer, the action pool is larger, there is one additional character present and the player has a much larger control over the game course.

That was reflected by the implementation and the game logic module is now more complex. However, the concepts of the behavior of the agents are captured by EWA in the same way as in SD One. RFs were used to capture reactions, goals and intentions with FSMs inside (e.g. Fig. 3.9) helped to implement the behavior of agents. This is a good sign as it shows that the concepts in EWA are sound and can scale up to medium-sized complexity of the scenario.

In SD Two, a set of debugging tools on top of EWA that simplified the development of the agents were developed and used. These tools allow a) to debug the agent's behavior during runtime and b) to analyze the scenarios offline by visualizing all the important events with graphical tools (see Section 3.5).

When it comes to movement of the characters in SD Two, as in SD One, the Leader Following steering is used when the characters are walking along as a couple. Moreover, a virtual quarrel simulator [Brom et al., 2012] was integrated with EWA that is used to conclude each SD Two game to display the ending the player achieved. The simulator can be used to specify the behavior of the characters also with regard to their spatial position that can be viewed in the environment. To illustrate the complexity of designing a *believable* movement of multiple characters, the triangular steering that is used by virtual quarrel simulator is described next in a greater detail.

Triangular Steering

Triangular Steering (TS) is an interesting use case of a social steering from a *believable* IVA perspective. It demonstrates how the basic steering techniques

[16]Note that the player is able to change Thomas's walking style when he is following the girl the same way as in SD One. Note that the emotion reaction to these changes were diminished as this presently is not the essential part of the game-play, but it can still be used as an additional measure to influence the girl's emotions.

[17]Note that not every action can be triggered by the player.

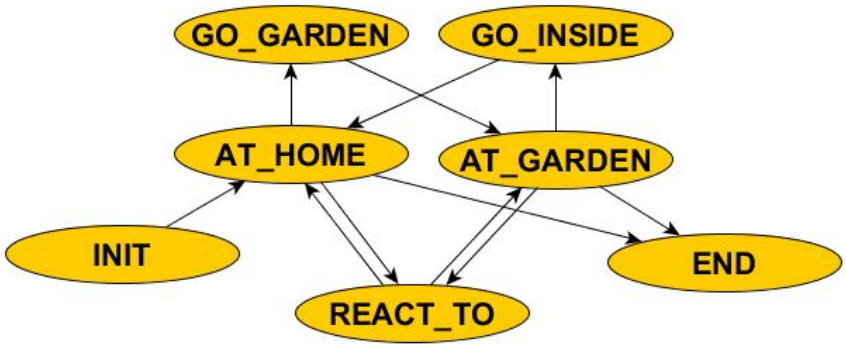


Figure 3.9: SimDate3D Level Two – FSM Example. Example of a finite state machine used in an intention of SD Two to represent character behavior when interacting at home. Both girls have a house with a garden and upon interaction, the characters can stay indoors or outdoors as they please. This behavior is represented by FSM that is also handling reactions to player-triggered actions by "react to" state.

can be used to simulate complex social relations between the characters. The following description is based on [Brom et al., 2012].

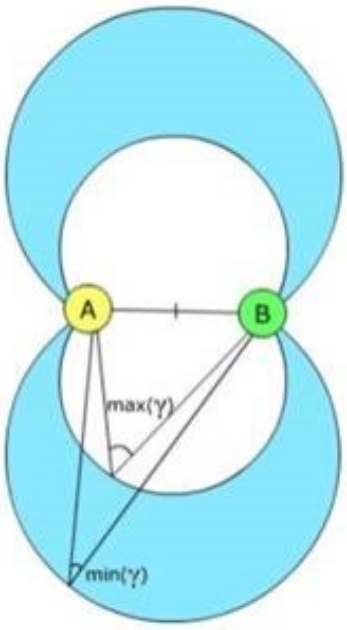


Figure 3.10: Triangular Steering Third Condition Visualization. Figure is taken from [Babor, 2012], page 14.

TS main use case is to position three agents interacting together in a triangle. The steered agent (named X) should keep a specific position and heading with respect to the other two agents in the conversation (named A and B). In our implementation, the position of agent X is defined by four conditions:

1. The distance between X and A as $< minA, maxA >$
2. The distance between X and B as $< minB, maxB >$

3. The angular interval specifying the required AXB angle as $< min(\gamma), max(\gamma) >$. The angle between location vector of X to A and X to B should range from $min(\gamma)$ to $max(\gamma)$ (see Fig. 3.10 for graphical visualization of this condition).
4. The preferred heading of X with regards to A and B is specified.

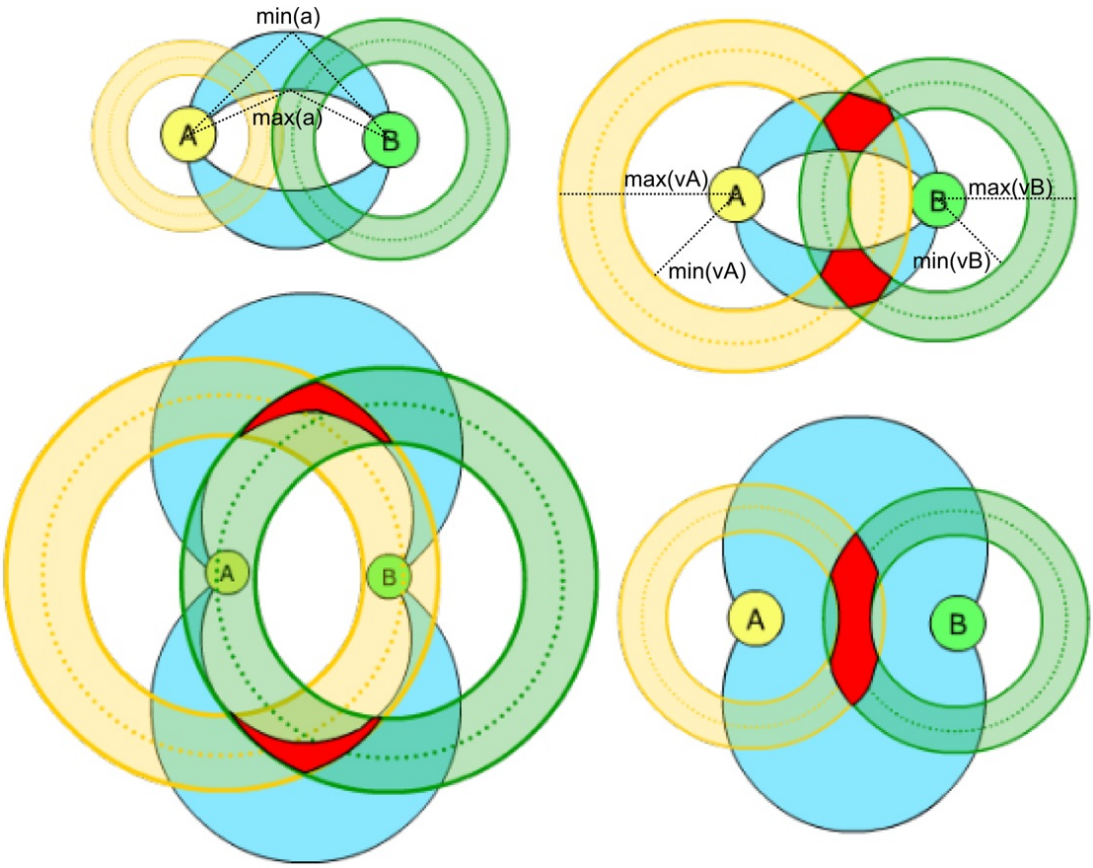


Figure 3.11: Four Results of Triangular Steering Conditions. Upper left image shows results with no intersection of the OR defined by conditions. Upper right shows a typical scenario with two separated coherent areas of OR where the X can be positioned. Lower left image shows the same in a different configuration and lastly, lower right image shows the situation where the OR is one coherent area. Figure is taken from [Babor, 2012], page 15.

The TS algorithm requires only the positions of A and B in the virtual environment. The result of TS is a point which attracts X and which represents the optimal position of X to A and B with respect to the conditions. Using the conditions, an *optimal region* (OR) defining the desired position of X is computed. Depending on the conditions, one of four results can be achieved (see Fig. 3.11). If there are two coherent OR areas defined by the conditions, the area closer the agent X is selected and X is positioned in its centroid. The centroid is used also in situations where there is just one coherent OR area defined by the conditions. In the unfortunate case with no overlap between the areas defined by the conditions, there are small attractive or repulsive forces defined with regards to the two other agents. These small forces lead to higher robustness and smoothness and help to solve situations when OR is empty.

Character's Reactions Customization

One of the differences between SD One and SD Two is the addition of a third virtual character – Nataly. When play testing the first version of SD Two game it became apparent that having both Barbara and Nataly react in the same way to Thomas's actions is not a desired feature of the system. Creating unique life-like characters means creating characters with unique reactions to situations in the environment. To support customization of character's reactions, *AgentParameters* class was used to set parameters for each character that later affected the way the character reacts to events in the environment when it comes to emotion elicitation and reactive behavior (e.g., Barbara might not like jokes). Thanks to the modular nature of EWA architecture, this was easy to achieve and it proved to be a useful feature improving the game play in SD Two.

Customized character's reactions can be seen the best when the characters are having a conversation. Now, not only the feeling value is used, as it is in SD One, but more parameters defining the "personality" of the character such as:

- *hysteria* – how likely the agent is willing to perform negative actions
- *jokesLiking* – how much the agent likes jokes
- *submission* – how much the agent likes being submissive "Action Being Bossy"
- *complimentLiking* – how much the agent likes compliments
- *impressLiking* – how much the agent likes show-offs (Action Impress)

are available.

The action selection based on the last action performed by another character in the conversation is implemented in class *FactoryReactionActionComputator* in SD Two.

Note that the feeling value between the characters is now also used when deciding whether the agent accepts or refuses a certain proposal, e.g. go home or kiss. For some of the proposals, the feeling value needs to be high enough for the agent to accept the proposal.

3.4.3 Discussion

SD Two is an upgrade of SD One in every aspect of the IDS curriculum. The game is longer, offers more actions, implements new mechanics and shows three characters that can now move almost freely around the city of EmohawkVille. The game-play is emergent – it is purely up to the player how the game is played. SD Two represents a much richer "story" domain over SD One.

EWA architecture scaled well for the purposes of SD Two. Two additional concepts were added to the architecture: a) the concept of Game Level Reactive Factories (GLRFs) and b) the set of debugging tools discussed in the following Section. GLRFs allowed for designing a more complex game logic easily and effectively by encapsulating the code into several classes. GLRFs react to any changes in the environment and the agent's goal states the same way Reactive Factories do. GLRFs have the ability to affect the decision-making modules of all the agents to adjust their behavior appropriately based on the current game state.

Apart from that, no additional concepts needed to be defined to specify the game logic or the behavior of the agents even though the game play and the behaviors became richer as can be seen e.g. in Fig. 3.9.

Overall, note that the SD Two does not provide a polished AAA[18] game experience. The goal was not to create a production-ready computer game but rather a) to show that the EWA concepts scale well for a larger application and b) to have a system producing stories of medium-sized complexity that are analyzed in Chapter 4. It resulted in a number of technical issues that can be solved by fine-tuning the parameters of the path findings, environment representation, navigation manager and the characters or a different behavior design. These issues are discussed below.

One of the issues that pop out in play-testing was the problem with synchronization of characters. While the behaviors of the agents are synchronized by EWA on goal and intention level, in a sense that Story Controller assures that all agents have the same goal scheduled at the same time with a proper intention, it was revealed that in real-time systems this might not be enough to achieve the best results of the joint behavior. In SD Two, the issue was demonstrated on:

1. Characters sometimes desynchronize at the cinema. One went to watch the movie, while the other waited, when the first one returned, the other one went to watch the movie.
2. When the girl is at home upon Thomas's arrival, sometimes she will "flee" to the garden with Thomas staying inside. This can be solved by proposing to go home with Thomas again.
3. Characters may take some time before they start jogging together in the monument park.

User interaction can be used to simulate the above issue easily – if the user waits until the agent starts performing certain behavior – e.g. going to watch the movie to the cinema or when the girl is going to the garden – she can interrupt Thomas's behavior by triggering an action that Thomas performs immediately. This puts the current intention to freeze and a new intention performing the action will be scheduled, run and then discarded. However, this is sufficient to make the behavior of the agents de-synchronized.

The root cause of the issue is that while the architecture supports a synchronized addition of the same behaviors to multiple agents, the problem is that the agents might be located elsewhere in the environment and their behavior execution steps may be triggered based on their position, which is different. Moreover, the user interaction can have many unforeseen consequences.

The mentioned problems can be solved by even stronger synchronization – e.g. Joint Behaviors from ABL language. However, adding these concepts poses a trade-off between the simplicity of the concepts that are used in the language and the behavior represented by agents. The problems might be also solved by a different game mechanics or behavior design. To give an example:

1. When user triggers an action, not just Thomas, but also the other character Thomas is interacting with is interrupted and thus behaviors remain synchronized.

[18] AAA pronounced "triple A" is an informal classification of computer games that fall into the category of games with high budget and usually well polished game experience.

2. The user is not allowed to interrupt characters when this could lead to de-synchronization of their behaviors.

The EWA path finding uses navigation point grid in EmohawkVille that is rather scarce[19]. This can lead to situations where the agents return to the nearest known navigation point by "going back", which might look suboptimal. The solution to this problem is either to add more navigation points to the environment or to exploit a newly added navigation mesh Pogamut feature generated by Recast [Mononen, 2009]. That would also prevent the issue at the end of the game where the virtual quarrel can occur at "odd" places. With better representation of the space, it is possible to compute a "good" place where all the agents fit based on the level geometry information for example.

Also, the obstacle avoidance steering was turned off not to interfere with user controlled Leader Following steering making the characters to bump into static obstacles occasionally.

User Evaluation. There were many informal play-testing sessions with students from Faculty of Mathematics and Physics of Charles University. Overall, the students liked the game and positively commented on the graphical content and the game play. That means that it is possible to create a working IDS product of medium-sized complexity with EWA. The Chapter 4 analyzes the domains of both SD One and SD Two in a greater detail.

3.5 Debugging Support

Developing agents living in a 3D environment is not an easy task. Trying to engage these agents in an interaction with each other and the user makes this problem even harder. Throwing into the mix narrative generation and affect simulation makes everything even more complex. To mitigate the challenges connected with developing agents with the EWA architecture for SD One and SD Two, the following tools were developed:

1. A set of graphical tools that can be used to debug the agent behavior on-the-fly during the run-time.
2. A log output of the EWA architecture that contains not only the log of everything that happened during the scenario, but also a graphical representation of e.g. changes in the agent's state over time.
3. A semi-automatic methodology that helps the developer make sense of what is happening in the IDS system that is being designed. More details are given in the second part of this thesis, in Chapter 4.

3.5.1 Run-time Debugging Tool

Run-time debugging tool consists of four main windows. The main window (Fig. 3.12) gives the designer the ability to run one of the defined scenarios and see the basic overview of all the agents that are in the scenario and their main properties. In EWA, this is for example emotion state of the agent.

[19]It was developed by the author manually.

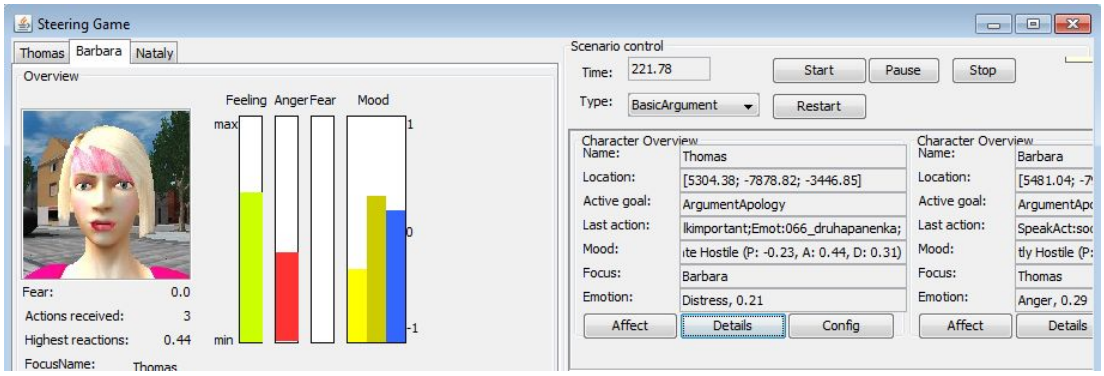


Figure 3.12: General Overview of the debugging tool. On the left, the developer sees the agent avatar along with general information about agent's current affective state. On the right, the developer has the ability to start, stop, pause and resume the scenario and inspect a pane with brief overview of all the agents present in the scenario where additional debugging information can be accessed by clicking on the respective buttons (Affect, Details and Config).

For every agent present in the scenario, the designer has an option to use three more overview windows. These are:

1. **General Agent Overview Window.** See Fig. B.1 in the Appendix. For a particular agent, this window shows a list of all the actions the agent conducted in the environment as well as the information of the current goals in the goal stack and the active goal that is currently being pursued as well as the state of the intention of the pursued goal.

2. **Agent Affect Overview Window.** See Fig. B.2 in the Appendix. An overview of the current emotion state is presented as well as the list of all the events that were processed emotionally by the emotion model along with their EEC values.

3. **Agent Settings Window.** See Fig. B.3 in the Appendix. The designer sees all the active Reactive and EECs Factories that can be turned on and off.

3.5.2 Graphical Tools For Log Analysis

Since the analysis of thousands of game plays produced by an IDS system manually is a daunting task, a support for condensed graphical log views that help the developer to analyze what happened during the single scenario produced by the system was developed. To support the analysis of the scenarios generated by EWA an XML format that is used to store everything that happened in the EWA as well as the current state of the agents recorded every tick of the DMS architecture is defined. While storing of all the data in the XML format is useful when processing these by computers, to support human designers, a number of additional graphical visualizations of the events, actions and agent's states during the course of one run of the system were implemented.

1. **General Overview of Agent State.** An overview plot consisting of a plot showing the overview of agent goals (Fig. 3.13) combined with a plot

showing overview of the agent's position in the environment (Fig. 3.14) is used as well as a plot showing the current focus of an agent – representing an agent the current agent is interacting with (Fig. 3.15).

2. **Evolution of Agent Affect Plot.** There are three plots used for visualization of agent affective state in EWA. There is a plot for general overview of the mood change over time (Fig. B.5 in the Appendix), a plot visualizing the change in the feeling values of particular agents (Fig. B.4 in the Appendix) and a tension plot visualizing the change in the tension values[20] (Fig. B.6 in the Appendix).

3. **Action and Events Log Output.** All the actions and events performed and processed by the agent are stored in the summary text files and the plots showing these on a timeline can be generated as well.

Combination of the plots from Figures 3.13, 3.14 and 3.15 provides a useful insight into high level scenario overview and was used as the basis for drama analysis discussed in Chapter 4 JFreeChart library [Object Refinery Limited, 2014] was used for the rendering of the plots presented here.

[20]This is discussed in detail in Chapter 4.

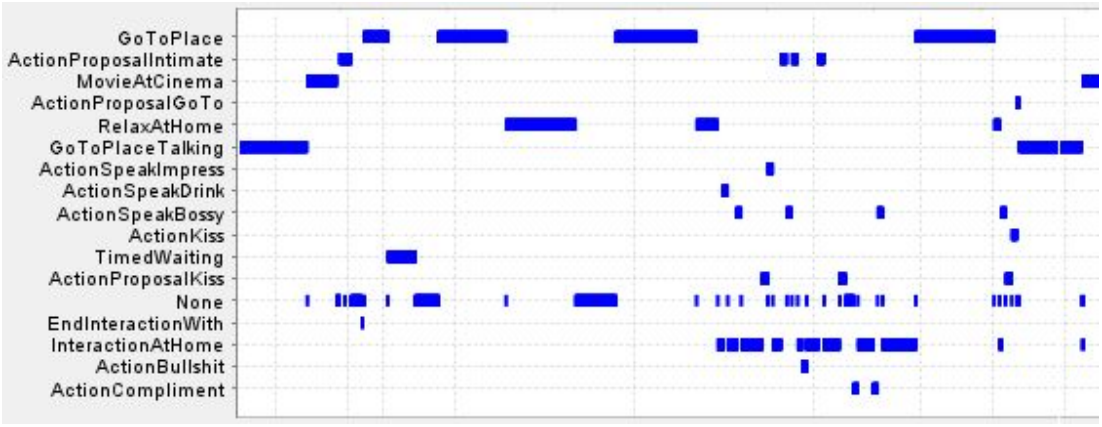


Figure 3.13: Log Visualization – Agent Goals. The plot shows all the goals that were active during the agent's lifetime with the respective time spans they were the active goal controlling the agent behavior.

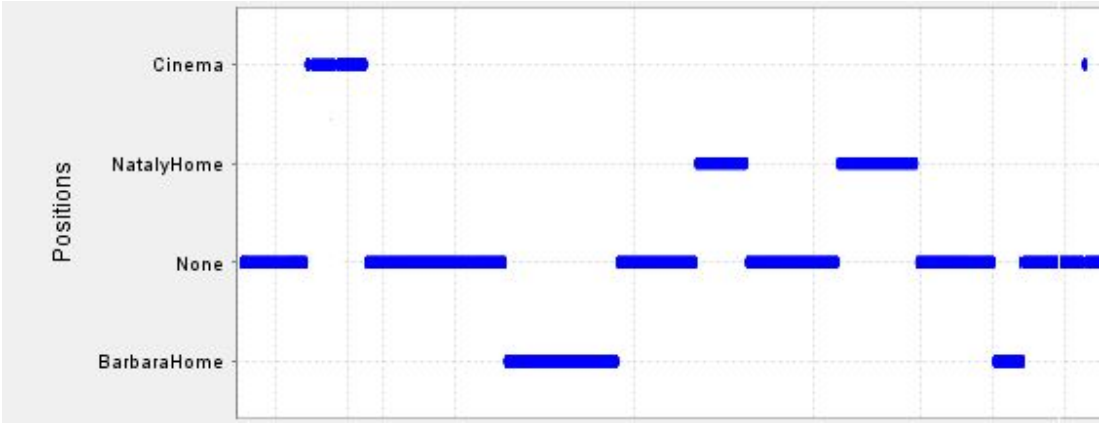


Figure 3.14: Log Visualization – Agent Positions. The plot shows the high-level location the agent was at during the agent's lifetime. The positions are specific to a particular IDS system and a defined scenario.

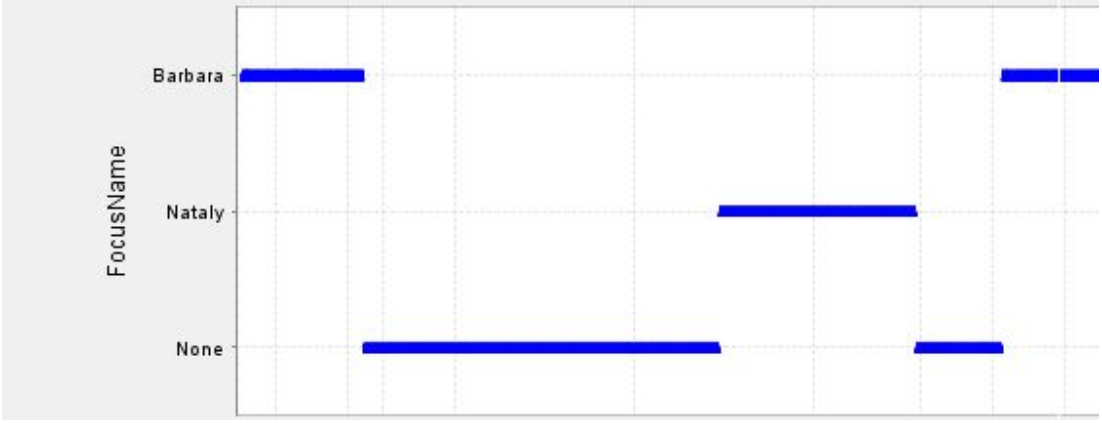


Figure 3.15: Log Visualization – Agent Focus. The plot showing the focus of the agent – typically another agent the agent was interacting with – during the lifetime of the agent.

3.6 Lessons Learned

Designing and developing an IDS system is a complex task – from the first behavior concepts proposal to beta testing of the prototype system, it involves many iterations of implementation and respective testing. In this chapter, two IDS systems of medium complexity – SD One and SD Two were presented. These are used to evaluate the concepts of EWA architecture for medium-sized drama specification. EWA accomplished to support requirements (R1-R8) as follows:

1. In both, SD One and SD Two, the concepts of the architecture were used to define a particular stage of the games. In SD One, fixed priority goals were used to model the starting behavior of agents and Reactive Factories were used to define game ending conditions and behavior. In SD Two, GLRFs were used to monitor game ending conditions and play the appropriate scenario with the virtual quarrel simulator. (R1.)

2. Both SD One and SD Two featured agents with reactive behavior. Agents reacted to the actions of other agents or to the changing state of the environment. This was implemented with Reactive Factories and intentions. (R2.)

3. Agents exhibited affective behavior portrayed by their reactions to the actions of other agents. For example, a lot of negative actions triggered by the user can lead to deterioration of the relationship between the characters that can end up in the agents departing and not wanting to interact with each other again even if the player forces them to. In SD One, the affective state was represented also by colored bubbles above the heads of the agents. (R3.)

4. Both in SD One and SD Two, the player interacted with the system by having a control of one of the agents. (R4.)

5. Both in SD One and SD Two, the behavior of the agents needed to be synchronized at certain points, e.g. at the beginning of the scenario with regards to SD One or at the end of the scenario with regards to SD Two. StoryController component was used as a synchronization point for the behaviors of the agents in SD Two. (R5.)

6. Steering Manager from [Popelová, 2011] was integrated with EWA and used for low level movement of the agents. (R6.)

7. A set of run-time debugging tools and tools for graphical visualization of game logs were developed to mitigate the challenges of the behavior development. (R7.)

8. Architecture, based on Java language, tried to minimize the number of the concepts needed for behavior specification to those that are considered state-of-the-art in the game development industry. (R8.)

In particular, the EWA architecture can be effectively used to specify, implement and debug the behaviors of IVAs living in 3D environments engaging

in social behaviors and dramatic situations. Next, the EWA architecture strong points, limitations and other lessons learned during the process of developing two medium-sized IDS systems are summarized.

Overall, it seems that the behavior abstraction defined by EWA architecture seems to be "about right" to solve the problem of medium-sized drama implementation. The design time in EWA is rather short, though thorough testing of the characters' resulting behavior is, of course, needed due to partly emergent nature of the plot of SD One and SD Two.

The ability to decompose the IVA behavior into goals and intentions at high-level and implement the behavior by FSMs on low-level proved itself to be beneficial as well. FSM are easy to use and implement but fail to scale. In EWA, FSMs are exploited on multiple levels of abstractions (actions and intentions) that allowed for effective behavior specification and reasonably easy debugging (e.g., the state of the FSM can be visualized in the custom logging message on an intention level).

It seems that EWA can be scaled well for scenarios of SD One and SD Two complexity with three characters and more than 15 different behaviors. In essence, it is a step forward to the solution of the *wicked problem* of a medium-sized drama.

Moreover, EWA architecture supports parallel execution of steering movement behaviors and high-level intentions. The architecture currently does not support parallel execution of two intentions. This was a design decision as managing parallel behaviors adds substantial complexity that can prolong the design time of the system and is not required for systems targeted by EWA.

The Emotion Model and the EEC factories enabled an easy to use affect modelling in SD One and SD Two. The interface implemented allowed an easy specification of the events that should be captured and annotated by the Emotion Model. The annotations with emotion variables were done manually, which posed a challenge when balancing the Emotion Model responses. However, this is hard to avoid given the fuzzy nature of the emotion research. Also, the particular character's reactions will be always tightly coupled with the system and the scenario the character lives in. An alternative approach to this problematics might be the use of machine learning techniques to balance the affect reactions based on an objective function. The main challenge is then to specify a proper objective function.

Additional interesting piece of information about affect integration in SD One and SD Two games is that for the scenarios of this complexity, there was no need for the full range of OCC emotions. In fact, the characters' behavior was modulated only with the *feeling* value, which was enough to capture the character's reactions in the scenarios of both games. However, it does not mean that for more complex and plausible simulations the author might find the full range of affect provided by the Emotion Model useful (as proven by other projects such as [Aylett et al., 2007]).

Customization of characters' reactions in SD Two turned out to be an important feature of the architecture. Having each character react differently to the events in the environment lead to the users perceiving the characters as different entities with custom personalities. The lesson learned here is that is not enough to equip the characters with plausible emotion model and decision making system, one also needs to account for the task of parametrization of these behaviors

so that the characters appear unique.

The debugging tools proved to be an essential component in SD One and SD Two development. Given the complexity of characters' behavior, it is absolutely vital to have tools allowing for the overview of agents' internal states and decision making system. Without these tools, the development with an architecture of EWA complexity would be hard, if not infeasible. Lesson learned here is that for IDS and AI systems, in general, it is worth to spend some time implementing a proper set of debugging tools that allow the developers to debug the system better. Moreover, it can increase the system scalability as the debugging tool allows the developer to understand the underlying system better enabling them to implement more complex behaviors.

Character's movement in 3D environment is one of the main topics of computer game development. The Navigation Manager integrated by EWA allowed us to tackle the problematics of plausible affect modulated movement in 3D environment in an effective manner. However, the general navigation of IVAs in SD One and SD Two could be improved. The goal of SD One and SD Two games was not to make a computer game ready for production but rather to prove the concepts proposed by EWA architecture are the right ones. This was the reason why the author did not spend much time optimizing the basic movement behavior of the characters as it was not the priority.

Another interesting outcome of the endeavor of designing EWA and implementing SD One and SD Two games is an informal methodology that can be used to design everyday life situation in virtual reality. What the author found to be an effective methodology to follow when sketching out the behavior of the characters in the virtual quarrel when the girls find out that the boy is dating both of them is the following:

1. Prototype the scene with StoryFactory tool to see how the resulting behavior could look like.

2. Discuss the proposal with professionals – e.g. actors, directors, game designers or in general, with people involved in the problematics of the simulated behavior.

3. Hire a set of actors who would improvise on the topic of the desired simulated behavior.

4. Annotate the resulting videos from the actor improvisation so that it is clear what needs to be simulated.

5. Implement the behavior model using agent architecture.

6. Test the resulting behavior with real users of the application.

To summarize, this Chapter discussed EWA architecture with respect to the SD One and SD Two IDS systems. Lessons learned from the development of the characters for storytelling systems present a great source of information that can be further employed to improve believable character design in many other applications. Moreover, creating SD One and SD Two systems enabled to pursue the Goal 4 of the thesis in the next Chapter.

4. Drama Analysis

Evaluation of IDS systems is a demanding process often requiring extensive effort. State-of-the-art evaluations of these systems involves either user surveys, e.g. [Schoenau-Fog, 2011], or technical evaluations such as comparing length and complexity of generated stories. Neither of those approaches scales well for a practical development. While user surveys can capture the quality of generated stories correctly, they are costly taking extensive amount of time and thus it is problematic to include them effectively within a regular development cycle. Since combinatorial explosion of the story space is in most cases a desirable property of an IDS system, it necessarily follows that for large-scale IDS systems, a thorough sampling of the story space by human users is costly at best and unfeasible at worst. While technical evaluations of the narratives are feasible for much larger story spaces than user studies and are significantly cheaper, they are only loosely related to the actual enjoyment of the stories by the user and do not allow the story author to understand what is happening in the generated stories and to shape them according to his artistic vision.

From a practical point of view, this is a problem as it makes the development of large scale IDS systems hard. When the evaluation methods available do not scale, the development of IDS system does not scale either limiting the complexity of IDS systems currently developed. Designing and implementing automatic or at least scalable methods mitigating the challenges of the evaluation of the IDS systems producing emergent narrative would allow the developers to create larger story spaces, longer narratives and richer computer games in general making a step forward to solving the *wicked problem* of IDS.

This Chapter tackles the problem of automatic analysis of stories generated by an IDS system and hence solves the Goal 4 of the thesis. The analysis outcome helps the developers to make sense of the story space generated by the system. The Chapter starts by outlying the related work in this area, then the problem analysis is conducted and a solution that is evaluated by a battery of experiments is proposed.

Note that the Chapter is loosely based on two papers of the author of the thesis [Bída et al., 2013] and [Bída et al., 2015].

4.1 Related Work

The Section presents an overview of literature relevant to story evaluation with preference of works proposing automatic or semi-automatic methods. The report is mostly a list of compelling ideas that could be reused in the context of an automatic narrative evaluation and clustering. The relation of the works presented below and the solution proposed in this Chapter is discussed in Section 4.3.4.

Note that sometimes it was hard to decide whether the work should have been placed in theoretical or technical part as most of the works were actually concerned with both. Author used his best judgment when organizing this Chapter.

4.1.1 Theoretical Approach

In [Brewer and Lichtenstein, 1982] authors propose a structural-affect theory of stories with three main aspects – surprise, suspense and curiosity. They claim that these aspects affect how much the reader likes the story and the authors outline how they manifest in stories. This work provides an interesting theoretical background that can be recycled.

In [Yannakakis and Hallam, 2007], Yannakakis and Hallam are tackling the question of "What is fun in games". Their paper presents a nice overview of psychological literature that deals with fun and flow theories in games. They also propose their own quantitative metrics that can be used to measure "fun" in predator/prey games[1]. Fun is measured according to three criteria (taken from [Yannakakis and Hallam, 2007]):

1. *When the game is neither too hard nor too easy.*
2. *When there is diversity in opponents' behavior between games.*
3. *When opponents' behavior is aggressive rather than static.*

Furthermore, they defined these criteria with functions and conducted study on the game of Pac-Man[2] where they showed their metrics indeed correlates with the reported player enjoyment. While their methodology limits its scope to predator/prey games, it still outlines some of the intriguing research directions that could be leveraged when defining general techniques capturing an elusive topic of the player enjoyment.

[Schoenau-Fog, 2011] discusses how to evaluate interactive narratives with users by questionnaires and intrusive methods. In particular, they propose following categories the qualitative study should be organized around:

- *Objectives* – the objectives set by the IDS system – usually hard coded by the author – referred to as *extrinsic objectives* or by the player when interacting with the IDS system (*intrinsic objectives*)
- *Activities* – a list of activities conducted by the player in order to solve the objectives
- *Accomplishment* – with regards to the authors or player objectives
- *Affect* – affect experienced by the player during the interaction with the system with respect to objectives and activities she is solving.

Although these techniques are not automatic, some of the ideas may be transferable to automatic evaluation systems and serve as good guidelines for good practices in the qualitative studies with human participants.

[Seif El-Nasr et al., 2013] reports on an in-depth qualitative analysis of interactive drama Façade conducted with a supervision of experienced psychologist. The analysis was carried out on 11 participants aged from 20 to 29 year. Authors reported on various playing styles based on participant gaming experience and cultural background. Three participants enjoyed the adventure, while eight

[1] Those are action games where the player is fighting virtual enemies. Typical examples are first person shooter games.

[2] For more details about Pac-Man game, see [Yannakakis and Hallam, 2005].

had a negative experience due to overall scenario setting of the Façade[3] or simply because the interaction with the game was limited. The paper shows the importance of qualitative studies as they can uncover many potentially unforeseen features of the IDS systems. Also, it features possible computer games user studies conducted by game developers.

[Zwaan et al., 1995] propose and test a model of how readers construct the representations of situations occurring in short narratives. They claim that the readers of stories update their mental models along five indices: *temporality*, *spatiality*, *protagonists*, *causality* and *intentionality*. These ideas could be used as a basis of automatic evaluation systems and some of them are already captured by our approach to automatically detect *sub-scenes* in a narrative.

[Vermeulen et al., 2010] in their paper "Measuring User Responses to Interactive Stories: Towards a Standardized Assessment Tool" come up with three essential categories when measuring the quality of the experience provided by IDS systems. Their approach is based on self-report questionnaires with questions on a likert scale from 1 to 12. The three categories the authors suggest the questionnaires should be organized around are (a) preconditions of meaningful user experiences, (b) common and frequent experiential qualities, and (c) concepts that reflect system-specific settings.

For part (a) their previous research identified the following aspects as critical (cited from [Vermeulen et al., 2010], slightly modified):

- *System usability* – interaction with the story must be technically smooth and error-free
- *Correspondence* of system capabilities with user expectations (the system needs to convey a reasonable expectation as to what kind of interactive influence users can exert on the story)
- *Presence* – users need to establish a sense of "being in the story world"
- *Character believability* – virtual agents must not damage user illusion, e.g. through irrational behavior or poor response to user input
- *Effectance* – users must be able to recognize when and how they have causally affected the story world.

Part (b) reflects the common user reaction patterns that can be observed within different types of IDS systems or narratives in general.

- *Curiosity* – what will happen next in the story
- *Suspense* – what suspense the user currently feels
- *Flow* – how immersed in the story the user is
- *Aesthetic pleasantness* – measures the positive experience of beauty or artistic impressiveness
- *Enjoyment* – how much the user enjoys the story

Part (c) should reflect questions that are tightly bound to the specific IDS system domain such as to the specific story content.

The authors tested their approach on an IDS system with N = 80 and confirmed the methodology can be used to asses the features of the system.

[3]Trip and Grace are going through marriage problems.

4.1.2 Technical Work

In [y Pérez and Sharples, 2001], y Perez and Sharples describe MEXICA, a system capable of producing emergent stories with user assistance. Among others, it uses *tension curve* of the stories to evaluate "interestingness" based on tension degradation and improvement in time. The tension curve could provide a good abstraction of the story that could be leveraged in automatic analyses approaches.

[Ware et al., 2012] present four quantitative metrics describing narrative conflict. The measurement and conflict detection are based on a planning representation of the narrative, making them harder to employ in non-planning IDS systems. Their evaluation showed the automatic evaluation matched user evaluations of conflicts. An interesting future work may be to detect conflicts in the story, evaluate these conflicts based on [Ware et al., 2012] and use the output as additional story features.

[Weyhrauch and Bates, 1997] proposes evaluation function for his interactive drama *Tea for Three* built upon *Moe* architecture that uses seven features to evaluate the quality of the experience. These are[4]:

- **thought flow** – whether one event in the user experience relates logically to the next
- **activity flow** – how bored the user feels when walking around uselessly
- **options** – measures how much freedom the user perceives she has
- **motivation** – whether the user actions are motivated by her goals
- **momentum** – measures the proximity of certain events the author prefers to happen together
- **intensity** – measures whether the user excitement builds
- **manipulation** – how manipulated the user feels

While these features are bound to the specific system and drama, they provide a good example of what can be measured in IDS and how it can be exploited in measuring the resulting user experience.

[Ontañón and Zhu, 2011] propose an analogy-based story generation system, where they evaluate the quality of resulting stories by measuring their *similarity* to "source" stories (input human-made stories). The *similarity* metric is based on MAC/FAC model [Forbus et al., 1995]. The concept of *similarity* is one of the promising directions that are used for automatic drama analysis. The interesting thing about the above approach that here the similarity is used to directly measure the quality of the stories, while in other approaches (some of them discussed below), the *similarity* is often used "only" to cluster the story space to provide the author a better overview of what happens inside many narratives generated by the IDS system.

[Cheong et al., 2008] present a system capable of extracting the story plan from game logs and generating a textual story summarization of the main plot events. Although the system is not concerned with story evaluation directly, story abstractions created by this method could be a good input for an evaluation metric.

[4]Cited from [Weyhrauch and Bates, 1997], p. 42, slightly modified.

Rabe and Wachsmuth [Rabe and Wachsmuth, 2013] propose an event and an episode similarity metric for their virtual character. The authors define *every observable occurrence as an event*. Each event has six main dimensions:

- *time* – time interval of the event
- *space* – name of the place the event is occurring (e.g. a restaurant)
- *protagonists* – name of the characters that are present during the event
- *intention* – the name of the BDI intention the agent is currently doing during the event
- *causality* – this represents what has lead to the event and it might be either a percept of the agent or an action of another agent
- *emotion* – contains the current state of the agent emotions represented by PAD space[5].

Furthermore, they recognize *episodes* as a sequence of *events*. And define a similarity metric called *episode distance* that captures how "far" the two episodes are from each other. Their *episode distance* is computing a number of event matches – it goes through all the events from episode A and compares them to every event of episode B and counts how many of the events of episode B are "close enough" to the event of episode A. The more events such as this, the closer the episodes are. This approach is then used to search in agent memory to find out all episodes that are close to current one to affect the decision making process of the agent. Although they are not using this in the context of story evaluation, the features they are working with might be reused in this context.

[Porteous et al., 2013] use Levenshtein string distance to measure differences between various stories generated by their storytelling system. By converting the events of the story to a textual representation (perhaps using some abstraction), this approach could be used as a means to automatically differentiate between stories generated by the IDS system.

[Aylett and Louchart, 2008] propose the use of double appraisal to increase the dramatic tension of an action. Their characters choose next actions based on their current emotions and based on the calculated expected emotional impact of their actions on other characters in the story. To simulate emotions and to measure emotional impact, OCC based model is used. This shows that the explicit emotions defined in the story can be a useful concept when manipulating or estimating the emotions of the user.

[Kadlec et al., 2013] uses compressibility and conditional entropy to measure similarity between sequences of actions gathered by human tracking and by daily corpora simulator[6]. While we could apply these algorithm to high-level abstractions of narrative in our systems, this approach might fall short in extracting the fine-grained features in the stories such as the building of the dramatic tension.

[Hawlitschek and Köppen, 2014] are studying the interaction of the players with the educational games. Their main concern is the detection of undesirable interactions with the educational games. They tested their clustering algorithm on game logs of a game "1961" – an educational game for history lessons[7]. Among

[5]Pleasure, arousal and dominance model based on [Mehrabian, 1996].

[6]They developed a system that simulates regular daily routine in a virtual environment with virtual characters.

[7]More information about the game can be found in [Friedemann et al., 2014]

features they clustered upon, they used actions conducted by a player in the game, time the players spent in the game and the time they interacted with the objects, NPCs and number of scene changes (which referred to the actual change of the graphical scene in the game).

[Wendel et al., 2014] presents an approach for automatic detection of situations in multi-player serious games. Their proposed approach is that first they define a set of situations in the game that are interesting from the game perspective. For each of these situations they define a set of criteria that can be fulfilled by the change of the state of the game. There criteria can be[8]:

- *atomic* – directly represented criteria such as "is the player moving"
- *spatial* – criteria taking in account the spatial information of agents and objects in the game
- *temporal* – criteria taking in account the temporal relations in the game
- *state-oriented* – criteria taking in account the game state, state of the objects or state of the players. From SD One and SD Two perspective this could be viewed as a current behavior that is performed by the character.

[Sali and Mateas, 2011] proposed visualization tools that are able to show the game logs to the author in a more conceivable way. They are demonstrating the soundness of their approach on an analysis of game logs from Façade. This could prove useful when conducting analysis of IDS systems in general.

The solution proposed in this Chapter is discussed with regards to the work above in Section 4.3.4.

4.2 Problem Analysis

Based on the overview above, it seems that the topic of dramatic evaluation involves general narrative and drama theories, psychological theories, human and computational story abstractions, algorithms allowing for comparisons and evaluations of these abstractions and data gathering methods. A multitude of works tried to approach the theme from IDS, computer games or psychological perspective, yet there is no general algorithm that could be used to estimate the "fun" level or "tension" in, e.g., SD Two. Nevertheless, there are many interesting approaches that could be distilled into concepts helping to measure these subjective perceptions. Moreover, even if solving the general problem of "teaching" the computer what a "good" story is turns out to be problematic, there can be always at least a semi-automatic approach that will help the author to estimate the quality of the stories by for example reducing the number of the stories that need to be viewed by the designer or suggesting particular stories as candidates for further inspection.

For starters, let's take a step back to Gustav Freytag's *Technique of the Drama* [Freytag and MacEwan, 1968]. Freytag proposed a 5 act dramatic structure that has later become known as *Freytag's pyramid* (see Fig. 1.1 in Chapter 1). From the Freytag's pyramid a notion of dramatic tensions (or suspense) can be generalized. That tension is usually low at the beginning of the story but

[8] As defined in [Wendel et al., 2014]

gradually rises until the conflict or climax is reached and then gradually decreases until the story ends. It is out of scope of this thesis to discuss how universal that tension development is but the notion of tension in the story could be a useful abstraction that could be used in general[9]. Let us assume there is an algorithm that computes a *tension curve* in a story. With such abstraction it could be possible to:

a) Define tension curve shapes that are appropriate for the studied IDS system.

b) If a) is problematic to define, there is an option to use the tension curve to categorize all the stories based on the tension curve shape in a way that stories with similar shapes end up in the same category.

Both a) and b) can help the author to evaluate the system. Having a), the author can see quickly which of the stories are producing "bad" drama that does not have an optimal shape of the tension curve. Having b), the author does not need to inspect all the stories but only several candidates from each category to get an overview of what the system produces.

Moreover, the solution b) hints at a possible side step from the ambitious goal of algorithmically defining the *proper* drama. Here, instead of solving the hard problem a method helping the author to evaluate the system by hinting where to look based on similarity of the stories based on their tension curve is proposed. Are there any other features of the stories that could be used to measure similarity? Indeed – starting from the story duration or number of characters in the story to the notion of which particular events in the story took place (e.g. a couple breaks up). The events could be for example mapped to particular stages of the dramatic structure of the Freytag's pyramid. Then, the author could filter the stories according to events taking place or not taking place at a particular stage (e.g. during the climax).

The story stages, their duration and content is another potentially useful abstraction of the stories. The challenge there is how to detect these automatically from the stories[10]. Having these more granular representations of stories would further help the author to see how the story evolves without the need to watch the entire story.

Apart from the general story features and abstractions, there is one more issue to tackle with IDS systems. Interactive IDS systems can generate potentially unlimited number of different stories. Sampling of this story space "by hand" is problematic at best. And while having the abstractions above might help, there is still a need to have a good sample of stories to run these analysis on. For generative IDS system such as [Sarlej and Ryan, 2013] that is easy to solve – the system runs for a while and the resulting stories are then clustered. However, having interactive systems where the user is part of the drama such as [Mateas, 2002] the need for the user interaction makes it hard to create a reasonably large sample of the story space to run the analysis on. That problem could be solved by introducing an artificial player into the mix. That entity would mimic the user

[9]For example, Mateas and Stern used concept of dramatic tension to advance story properly in interactive drama Façade [Mateas, 2002].

[10]The tension curve abstraction may prove useful here as well – e.g. point with the highest tension is the climax.

playing with the system in ways that would allow to get a good sample of the story space generated by the system. Next section presents a solution to some of the problems outlined above with regards to related work mentioned earlier.

4.3 The Proposed Solution

This section proposes a semi-automatic drama analysis method that pivots on a meaningful clustering of stories into groups that contain stories with similar features. In particular, the following is solved:

- **Finding useful story features that could be used as means for story abstraction.** General features of the stories are defined such as "tension curve", "sub-scenes" – particular stage of the story or "action strings" – high-level string representation of stories that are discussed in the next Section. Apart from clustering, these features could be further used to evaluate the story and its quality.
- **Clustering of stories to groups of similar stories.** A clustering algorithm is being applied on features above to group together stories in a meaningful way – ideally, stories sharing some important plot features should be grouped together.
- **Exploring the story space of the system automatically.** An artificial player is proposed and implemented to sample the story space generated by a game or IDS system.

From the IDS system developer perspective, the solution above would work as follows:

a) The developer uses artificial player to sample the story space of IDS system to generate large number of stories.
b) The developer runs the algorithm extracting the story features and clusters the stories into groups.
c) The developer reviews a sample of stories from each group.

Based on the findings the developer can now refine the artificial player parameters to search more particular story branches and reiterate a), b) and c). In the end, the story space of the system should be well sampled and evaluated saving the development and play-testing time.

The novel part of the approach is the automatic sampling of the story space combined with a definition of general features abstracting the stories and their extraction methods and with a clustering algorithm that clusters the stories into groups of stories that share similar features.

Next sections present in a greater detail the three points above starting with story abstractions, continuing with clustering and closing up with using the artificial player to sample the large story space of IDS system.

4.3.1 Abstracting The Stories

To be able to cluster, the features the clustering algorithm will be using need to be defined first. One of the features proposed is a *tension curve* is going to

be discussed later in this Section. What are the other features proposed? Let's imagine a normal story written as a text. Or even better, millions of such stories written by enthusiasts on the internet. Those could be easily compared by using some well defined String distance metrics, e.g., Levenshtein distance [Levenshtein, 1966]. Those metrics could be run on the text but apart from detecting plagiarism that might be of little value. The goal is to cluster together stories in the sense of the plot, events and characters and not in the sense of the actual written form.

So, is there any sense in comparing the of stories with these "dramatically naive" methods at all? What if the underlying representation of the story the algorithms are using would be changed to a more abstract form? What if instead of letters, words or sentences the algorithms would use a high-level story abstraction that would only capture the important plot events? Moreover, a whole story could be "serialized" to one textual string according the events that happened inside of it.

The "general" story feature that is proposed in this regard is: to represent the whole story as a String of letters. Each letter defines some story event, and the letters are ordered the same way as the events represented by them appear in the story. All of the events in the story could be used or just some of them, for example, based on the importance of the event concerning the plot.

So, what events should be picked and how to extract the actual events from the story in the first place? Concerning the latter, a typical IDS system is usually generating the story from some higher-level abstraction, e.g. Façade uses beats and EWA uses goals, intentions, events and actions. Those concepts could be leveraged and letters could be assigned to them to get the underlying string representation. For example, "Thomas greets Barbara" can be serialized as agent T performing action G, where the first letter T represents agent Thomas and the letter G represents the action of *greeting*. By concatenating those letter representations one gets the whole story represented by an *action string*.

Motivation for using string metrics in this analysis is supported by three main points. Firstly, there is an expectation that similar stories will be represented by similar action strings. Secondly, the action sequence is a reasonably precise representation of the story as everything that happened can be serialized. Thirdly, string distance is a well studied subject and there are some algorithms that can be applied right away, for example in the case of the Levenshtein distance – applying it on action strings can be perceived as a rudimentary form of story editing.

It seems that action strings might be a reasonable representation as they capture basically everything that happens inside the story. But capturing everything might turn out to be problematic. A lot of what happens in the story might be not that important plot-wise. That is why another abstraction that can also be represented by an action string is proposed. This abstraction is decomposing the entire story into a set of *sub-scenes*. The definition of a *sub-scene* is as follows.

A *sub-scene* is a time span of the story where:

a) The set of characters that are in the proximity of the main protagonist do not change.
b) The location of the main protagonist does not change.

Let's show on an example of the SD Two domain. Imagine that Thomas (the main protagonist) is with Barbara (character) in a restaurant (place) – that

defines one sub-scene "Thomas is with Barbara in a restaurant'. After 5 minutes of conversation, Nataly arrives and joins them. Now, the old sub-scene ends and a new one begins – "Thomas is with Barbara and Nataly in a restaurant". That decomposes the entire story into a set of "building blocks" that can be further analyzed. It is not hard to imagine that the scene "Thomas is with Barbara in a restaurant" may feature different interaction based on the actual state of the relationship of Thomas and Barbara. That can be captured by analyzing the character's emotions that are elicited during the sub-scene or by looking into how the relationship of the characters changes, comparing the beginning and the end of the sub-scene. That can be used to conclude that for example, "Thomas and Barbara had a fight at the restaurant because their relation measured by the feeling value deteoriated".

The advantage of sub-scenes is that they can be used as a general feature that is potentially applicable to any kind of story. Moreover, it is not hard to assume that the extraction of sub-scenes from game logs can be done easily. The time span of sub-scenes can vary. Especially, the minimal lower limit of a sub-scene duration should be enforced to avoid artifacts in the data (e.g. a sub-scene with a very short duration is probably not desired).

The sub-scenes can be represented by letters and form action strings as follows. If the game contains only a known set of sub-scenes, a particular letter can be assigned to each of those in advance. If that is not the case, each feature of the sub-scene can be represented as a letter and these letters can be then combined together to form a sub-scene string representation. Let's see on an example used in the SD Two domain. "Thomas is with Barbara in a restaurant" forms "TBR" and "Thomas is with Barbara and Nataly in a restaurant" forms "TBNR". The sub-scene sequence string is then simply a concatenation of strings representing individual sub-scenes (see Section 4.4.2 for a more formal definition).

Other features that might be useful in general are a number of characters in the story, a number of sub-scenes in the story and the story duration. The next section presents in a greater detail an additional feature of the stories that is proposed – the *tension curve*.

Tension Curve Extraction

How to extract a tension curve from generated stories in general? One possible way is to let the authors annotate the story with a vector of how the tension in the story evolves, getting the tension curve right away. The problem with this solution is that the entire story needs to be watched by the annotator, making these annotations time consuming.

On the other hand, the story tension is often related to emotions experienced by the story protagonists. Thus, the tension could be extracted from character emotions if the IDS system does explicitly represent them. That also has some limitations, e.g., imagine a dark figure following someone in a deserted part of the town. As the figure gets closer to the unsuspecting victim, the tension in the story rises. However, that may not be represented well by character's emotions as the one who is followed may not be aware of that someone following him/her. This issue could be solved by generating the tension curve from the simulated emotions of the viewer of the scene. The viewer has all the information necessary

to dramatically assess the situation and hence, when the emotion model would be accurate enough, the correct emotions of anger or fear should be elicited.

To summarize, a method of extracting the tension curve from a game or an IDS system is proposed as follows. If the system integrates an explicit emotion model[11], the emotions generated by the model could be leveraged. To create a tension curve, the emotions in the system would be sampled every N milliseconds. Then, a definition of how emotions contribute to the tension curve can be made resulting in a tension curve vector where the values range between -1 and 1. The vector then represents the tension in a particular story.

A general tension curve is a feature of the stories that is proposed. Moreover, in sub-scenes, a *feeling difference* is defined as the difference between feeling values at the beginning of the sub-scene and at the end of the sub-scene:

$$feeling_{diff}(s) = \sum_{a,b \in Agents} feeling(a, b, s_{beg}) - \sum_{a,b \in Agents} feeling(a, b, s_{end}) \quad (4.1)$$

Then, a *condensed tension difference* curve is defined as a sequence of *feeling differences* for all the sub-scenes occurring in the story. The *condensed tension difference* should be significantly shorter than the tension curve – although they might correlate with each other.

4.3.2 Clustering

To compute clusters the *k-means* clustering algorithm [Friedman et al., 2001] is used. The k-means clustering algorithm is a method of analysis which aims to partition n samples (in this case stories) into k clusters (k groups of stories) such that the distance between stories within a single cluster is minimized. The distance is computed with a user specified metric. The k is fixed and given by developers. The k-means algorithm requires initialization of the starting position of clusters.

The algorithm works in two stages (see Alg. 5). First, it assigns the nearest cluster to each sample. Then it moves all clusters by computing the average of members in the cluster and shifting the cluster towards the new average.

```
while clustering-steps <MAX and clusters-are-changing do
    for Sample s from samples do
        Cluster nearest = findNearestCluster(s);
        assingSampleToCluster(s, nearest);
    end
    for Cluster c from clusters do
        Centroid new = computeClusterCentroid(c);
        shiftClusterToCentroid(c, new);
    end
end
```

Algorithm 5: Clustering Algorithm

[11] Which is often the case as in SD One, SD Two or FearNOT! [Aylett et al., 2007].

These two stages are repeated until the clusters become stable or a maximum number of steps is reached. The idea of using k-means is that if the distance between stories is defined right, then every resulting cluster will contain stories that are somehow similar to each other.

Clustering Distance Computation

To compute the distance between stories using tension curves the following approach is used. The distance between tension values from one story and the values from the other story are summed as an absolute values as seen in Formula 4.2, where x and y are tension curve vectors of respective stories and n is the length of the vectors. If the tension curves differed in length, the shorter was expanded to match the size of the larger by adding zeros.

$$distance(x, y) = \sum_{i=1..n} \left| x_{(i)} - y_{(i)} \right| \tag{4.2}$$

To measure distance of action strings abstraction, three standard string difference metrics were tested – the Levenshtein distance [Levenshtein, 1966], the Jaccard index [Jaccard, 1901] and the Jaro-Winkler distance [Winkler, 1990].

Levenshtein distance between two strings is defined as the minimum number of single letter edits – insertions, deletions or substitutions – that are required to change one string into the other (see Formula 4.3).

$$d_{ij} = \min \begin{cases} d_{i-1,j} + 1_{del} \\ d_{i,j-1} + 1_{ins} \\ d_{i-1,j-1} + 1_{subs(a_i \neq b_i)} \end{cases} \tag{4.3}$$

Jaccard index is defined as the size of intersection divided by the size of union of two sets (see Formula 4.4). The sets are defined by strings in a way that the string is a set of all the letters inside it (not concerned with the actual ordering of the letters). That seems naive plot-wise but serves as a base line for the other metrics that are used.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.4}$$

Jaro-Winkler distance is based on the Jaro edit distance [Jaro, 1989] defined in Formula 4.5:

$$d_j = \frac{1}{3} \cdot \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t}{m} \right) \tag{4.5}$$

where:

- $|s_1|$ and $|s_2|$ are lengths of the respective strings
- m is the number of matching characters in the two strings
- t is the number of transpositions divided by 2 (characters are matching but are in a different order in the sequence)

Note that the two characters from strings *s1* and *s2* are considered *matching* only if they are the same and closer than:

$$\left\lfloor \frac{max(|s_1|, |s_2|)}{2} \right\rfloor \tag{4.6}$$

Jaro-Winkler edit distance is then defined as:

$$d_w = d_j + \big(l \cdot p \cdot \big(1 - d_j\big)\big) \tag{4.7}$$

where:

- d_j is the Jaro distance between the strings
- l is the length of the common prefix
- p is a scaling factor of how much is the score adjusted based on the common prefixes

All of the string distance metrics presented above were used to compare the distances between action strings representation of the stories.

To run k-means clustering with the distance metrics presented in this Section two more issues need to be solved: a) random initialization of clusters and b) computation of cluster average from its members with a given distance metric. Concerning a) the random initialization of clusters sometimes caused undesirable behavior where one or more of the clusters were so far from the samples that they stayed empty the whole time. To compensate for that the clusters are initialized as follows. For each cluster one story from the sample is set at random and defines the initial position of the cluster. Concerning b), the problem was that for string distance metrics it may not be clear what an average of several action strings means[12]. When computing the average from cluster members with string distance metric all members of a cluster were iterated and the member with the lowest sum of distances from the other members was selected (Formula 4.8). That member then became the new cluster average.

$$min_m\Big(m \in Cluster \sum_{n \in Cluster} distance(m, n)\Big) \tag{4.8}$$

4.3.3 Story Space Exploration

As mentioned above, a modern IDS system of say a Façade [Mateas, 2002] complexity produces hundreds, thousands or even millions of different stories. Sampling the resulting story space by hand is unfeasible. That is also one of the reasons why some of the IDS systems employ techniques that fairly limit the players capabilities of influencing the story – just to keep the system tractable when debugging. Moreover, as the IDS systems are often interactive requiring a human user in the loop, it makes the sampling of such systems hard requiring a lot of testers working for an extensive amount of time.

A solution to the problem here could be an introduction of an artificial player. The story space that needs to be explored would be specified (perhaps using the abstractions portrayed above) and the artificial player would traverse the story

[12]For example Jaro-Winkler distance does not even obey triangle inequality.

many times saving the progress for further analysis. The abstractions could be used in two ways – either to specify which directions the artificial player should follow or which parts of the story space should be avoided. The directed search is beneficial for the designer to see in a greater detail stories of a particular type, e.g. all the stories where the wolf did not eat Red in the Little Red Riding Hood story.

On the other hand, exploring parts of story space that deviate from a given set of stories might reveal previously unseen parts of the story space to the designer, e.g. to look in all of the stories where Red did not reach her granny (the scene that should be avoided here would be "Red meets her granny"), or "look into all the stories where the tension at the end is high". The artificial player implementation will differ based on the system the player is traversing. However, the general approach to explore unseen or to probe more already seen holds.

The resulting stories can be summarized by logs and graphs and can be further analyzed by the clustering method showing trends in the IDS systems (for example, all stories where the player does not meet the wolf share a low tension value).

4.3.4 Discussion

The main contribution of the solution above is putting together story abstraction methods with semi-automatic story space exploration techniques and using machine learning algorithms to estimate the level of the story similarity. The author of the thesis might have been also the first one who used k-means clustering for stories produced by an IDS system [Bída et al., 2013].

Moreover, to the knowledge of the author only little work has been done on story clustering with the exception of [Hawlitschek and Köppen, 2014] where they used a clustering approach similar to [Bída et al., 2013]. The main difference between is that their clustering approach is based on different features tailored for the game they studied[13]. However, their *k-medoids* approach [Kaufman and Rousseeuw, 2005] might be an interesting future work.

The tension curve was used in [y Pérez and Sharples, 2001] to judge how is the story interesting. In the solution here, the tension curve is used as a feature for clustering of the stories.

It was shown that except of tension (suspense) curve there can be other curves observed such as surprise or curiosity curves as proposed in [Brewer and Lichtenstein, 1982]. These might be good candidates for additional high-level story features. The question is how to extract those automatically.

The concept of judging stories based on their similarity was proposed in [Ontañón and Zhu, 2011]. A next step we could take could be to tag the "good" stories and let the system to compute the ones that are the most similar to those. The solution presented in this Chapter defines different *similarity* concept based on the tension curve distance and action string distance.

[Porteous et al., 2013] used the Levenshtein string distance for similarity analysis of virtual drama. In the solution proposed here, Levenshtein distance metric

[13]The features they used are: number of "view" actions, number of "take/press/open" actions, number of "combine", number of "dialogs", "time in dialogs", "time in game" and number of "scene changes".

is used to compare the similarity of action strings along with other metrics enabling to compare the metrics with each other.

[Wendel et al., 2014] presents an approach for automatic detection of situations in multi-player serious games. That is highly relevant to the solution here as sub-scenes are proposed as a high-level feature of the story as well. In the implementation here the used *sub-scenes* detection mechanism is based on all four of [Wendel et al., 2014] criteria. However, the situation is not defined in an explicit way as in [Wendel et al., 2014] but rather specified as the conditions that based on the four criteria define an "interesting" situation.

Talking about using of visualization tools for game logs analysis as discussed in [Sali and Mateas, 2011], when conducting analysis on SD One and SD Two the visual debugging tools defined by EWA architecture (see section 3.5.2) were used extensively. Those have been useful in: a) debugging of the behavior of IVAs, and b) identifying potentially interesting drama points in the story that helped when designing the solution proposed in this Chapter.

Also, note that relating to qualitative techniques mentioned in the Section 4.1, there were many informal play sessions with SD One and SD Two where it was studied how the games are played and the feedback was gathered.

Moving away from related work, another question is how to extract a tension curve from a system that does not have an explicit emotion model or a notion of emotions. For that system it is indeed harder but one possible solution could be to design an emotion model tailored for the particular system that would try to simulate emotions of the users watching the drama. That should be feasible for a system with a reasonable level of abstraction – e.g. for systems that define a notion of events or actions which is often the case in most of the IDS systems.

4.4 Implementation

This Section summarizes the implementation of the features extraction, the artificial player exploration and the clustering.

4.4.1 Action Strings Representation

Action strings were extracted for SD One and SD Two domains and the MOSS domain. In SD One, the action string are represented as a string of letters where one letter (A) represents an action that happened in the environment:

$$Action_{SD1} = A$$
$$A = ActionType$$
$$ActionString_{SD1} = [Action_{SD1}]+$$

"Thomas greets Barbara", "Barbara greets Thomas", "Thomas asks Barbara to go to the cinema" are three actions from a SD One story. That sequence is represented as "HHG"". Note that there were only two characters in SD One and the characters took turns typically – every action of one character was followed by a reaction of other character. That lead to the decision of omitting a letter representing the actual character that performed the action.

In SD Two, additional letter (*X)*) is added to action letter representation to represent the agent that performed the action:

$Action_{SD2} = XA$
$X = AgentName$
$Y = ActionType$
$ActionString_{SD2} = [Action_{SD2}]+$

"Thomas greets Barbara", "Barbara greets Thomas", "Thomas asks Barbara to go to the cinema" would be then represented as "THBHTG".

There were around 50 possible actions in SD One and around 80 in SD Two. See 4.5.2 and 4.5.2 for examples of action strings from SD One and SD Two domains.

In the MOSS system, the stories are represented as a list of facts. An alphabet was created that converts that list into letter representation in a way each fact has a letter assigned. If there are variables within the fact a letter is assigned to those as well. For example:

distress(dragon,becametrue(neg(is_free(dragon))),1)

represents that the dragon in the story at the time 1 experiences distress because it was captured. For every fact and entity in the formula a letter is assigned resulting in *"a,b,c,d,e,b,g,;"* where *a* represents distress, *b* dragon, *c* becametrue, *d* neg, *e* is_free, *b* dragon and *g* 1. The letters are then re-used for the same concepts found in another Prolog rules. When a new fact or a variable does not have a letter assigned, a new letter is generated. This mapping is then stored.

4.4.2 Sub-scenes Extraction

Sub-scenes were used to better analyze SD Two domain. In SD Two, sub-scenes were extracted as follows. One sub-scene in SD Two is represented by two, three or four letters – one letter represents a location of the story (e.g. P for park) and the consecutive letters represent characters in the sub-scene (e.g. T for Thomas; one letter per each character present). For example, the "TBR" string represents a sub-scene where "Thomas is with Barbara at the restaurant" and "TP" is "Thomas in the park". The sub-scene sequence string is simply a concatenation of strings representing the individual sub-scenes. Formally, if *X*, *Y*, *Z*, and *P* are letters then a sub-scene string sequence is defined as:

$SubScene_{SD2} = XYZP$
$XYZ = AgentNames$
$P = Place$
$SubSceneStringSequence_{SD2} = [SubScene_{SD2}]+$

Sub-scenes detection was not implemented for MOSS stories because the MOSS stories are already relatively short.

4.4.3 Tension Curve Extraction

To extract the tension curve the following approach was used. As the characters are equipped with OCC based [Ortony et al., 1988] emotion model which directly influences their decision making, the tension was defined as follows: Every 250 ms a snapshot of all characters' emotions is made. Then the sum of these emotions is computed in a way that every positive emotion is counted with a minus sign and every negative emotion is counted with a plus sign. The resulting number encoded the tension value at the moment. The *tension curve* is then simply the *piecewise linear function* defined by these values. See Formula 4.9 for the definition of the tension value at time t.

$$tension_t = \sum_{a,b \in Agents} \frac{feeling_t(a,b)}{N_{agents}} \tag{4.9}$$

In the MOSS system the emotions are also defined explicitly as a part of the generated stories. The MOSS emotions were categorized to positive and negative categories and the sum of these is then used as the tension value at the specific time point of the story.

4.4.4 String Metrics Implementation

As an implementation of string metric distance a Java toolkit [Alias-i, 2015] for language analysis was used. In the Levenshtein distance the cost of insert, delete and substitute operations was set to 1. In Jaro-Winkler a prefix of length 4 (l) was used with boost weight of 0.7 (p).

4.4.5 Artificial Player Exploration

The artificial player was implemented for SD One and SD Two domains. In SD One, the artificial player simulates key input normally performed by a human user by randomly pressing one of the keys that are triggering actions in SD One (class *UserSimulatorLevelOne*). The delay between two key presses is randomly selected from a specified interval (between 4 and 31 seconds). The actions that are triggered with the same probability are:

- A random positive action
- A random negative action
- Weird walking
- ActionJoke
- ActionCompliment
- ActionInsult
- ActionBlame
- A random neutral action

As SD Two game mechanics is more complex than SD One it was necessary to implement artificial player that takes into account game mechanics and is able to work with sub-scenes level representation (class *ScenarioExplorerUser*, Algorithm 6). The artificial player (controlling Thomas) extracts the sub-scene sequences from the given set of stories (*getListOfScenarios()*) and then tries to achieve

a different sub-scene sequence in the story (*planNextSubscene(listOfScenarios)*). For example, if the artificial player detects that most of the given stories started with characters at the restaurant, the game location in the story will change to a different one by inviting the character for example to the cinema and so forth for the second and the n-th sub-scene in the sequence.

Moreover, the artificial player has a simple domain-specific knowledge that limits the actions considered only to those contextually appropriate (e.g. Thomas does not try to cuddle a girl in the restaurant).

```
List<Scenarios> listOfScenarios = getListOfScenarios();
Subscene nextSubscene;
while scenario_time < MAX_SCENARIO_TIME do
    if nextSubscene == null then
        nextSubscene = planNextSubscene(listOfScenarios);
    end
    reachNextSubscene(nextSubscene);
    if subscene_reached(nextSubscene) then
        double max_subscene_time = planNextSubsceneTime();
        while time_in_subscene < max_subscene_time do
            if current_time > next_action_time then
                performNextAction();
                next_action_time = planNextActionTime();
            end
        end
        nextSubscene = null;
    end
end
quitTheScenario();
```

Algorithm 6: Artificial Player SD Two Pseudo-code

4.4.6 Clustering

To accommodate for non-standard metrics used in clustering a custom k-means clustering implementation was made (*KMeansClustering* class) allowing to determine a centroid of a cluster properly in case of string distance metrics.

In case of SD One and SD Two, clustering analyzes features that are extracted from game logs that are stored in XML files. The format of the files is the same for SD One and SD Two. In one XML file, a progress of one game is stored. In particular, following information are saved (see Section 3.5.2 for the visualization of some of these):

- Agent location (snapshot is made every 250 ms)
- Agent feelings towards other agents in the scenario (snapshot is made every 250 ms)
- Active goal of the agent at time (every 250 ms)
- List of agents that were visible at particular time (every 250 ms)
- Agent ALMA mood development during the scenario
- List of actions the agent performed with time stamps
- List of events the agent experienced during the scenario with time stamps

The information above are enough to compute the abstract story features and hence conduct the clustering based on the distance metrics proposed in this Chapter. The actions are stored enabling to extract the string representation of the story and the feeling values are stored enabling the extraction of the tension curve.

It also quickly became apparent that to run clustering effectively in a timely fashion the distances between members being clustered need to be pre-computed. Hence, the clustering works in two stages:

1. All members (stories) that need to be clustered are loaded into memory and all metrics that are going to be used for clustering are pre-computed for every two members and the results are saved to a file[14].
2. Clustering loads the file and runs the algorithm based on the information in the file saving the computation time as the distances between the members are already computed.

A console tool implementing 1) and 2) for SD One and SD Two domains was created as one of the outcomes of this thesis.

Note that prior to the clustering all pairwise distances between stories have been normalized and standardized.

4.5 Evaluation

To evaluate the implementation of the solution proposed in this chapter four experiments have been conducted on SD One (Section 3.3), SD Two (Section 3.4) and MOSS [Sarlej, 2014] systems.

SD One is a simple 3D dating game where the goal of the user is to achieve that a couple – Thomas and Barbara – gets to the cinema. The game comprises

[14]XML files were used as the data format in the project.

Figure 4.1: Screenshots from SimDate3D showing interactions between characters.

a sketchy conversation through comic-like bubbles called emoticons and the user partially controls one of the characters actions. There are three possible endings of this story: a) characters get to the cinema safely, b) characters get angry and part and c) characters interaction is too positive, so they decide to skip the cinema and head home.

SD Two (Fig. 4.1) is an extended scenario, where Thomas is now dating two girls at the same time. All game sessions end with all three characters meeting and engaging in an argument. The outcome of this argument depends on previous user actions and there are four endings: a) Thomas staying with Barbara and breaking up with Nataly, b) Thomas staying with Nataly and breaking up with Barbara, c) both girls breaking up with Thomas and d) Thomas staying in the relationship with both girls.

The MOSS system [Sarlej, 2014] developed by M. Sarlej generates textual short stories with morals (e.g. greed, retribution, etc) in three domains (animals, family and fairytale). Each moral has its own emotional pattern that is used to generate stories with moral of a particular category. Internally the system uses Prolog abstraction to generate the stories, which is then translated to human readable text with Perl scripts. The internal Prolog representation of the stories was parsed and analyzed.

4.5.1 Overview

There were four experiments performed in total (Experiment 0, 1, 2, and 3).

Experiment 0. In the experiment, the goal was to evaluate the quality of k-means clustering based on story distance definitions. The quality of clustering in the Experiment 0 was determined by the stories in the cluster sharing the same

ending. The performance was tested on two domains – SD One and SD Two. SD One presented a simple domain and SD Two a complex domain. For SD One k-means clustering with two, three and four clusters (defining the resulting number of clusters) was run. SD Two clustering was run with k set to four, five and six clusters. The k was determined arbitrarily based on the number of different game endings in the domains. The goal of the experiment was to see how well the general features proposed capture a concrete feature of the domains – the game ending.

It is known that k-means clustering may converge to local optima. To account for this, the clustering was run 100 times for all the experiments and the results were averaged. To provide a simple baseline to the measurements, the clustering analysis was compared to assigning the stories to clusters at random. Once again an average of 100 random assignments is measured.

Experiment 1. In the experiment, a more thorough analysis of the similarity metrics regarding the game ending and the similarity of stories was conducted on the domain of SD Two. The story similarity is now measured also with regard to how much time various characters spend together in the story.

Experiment 2. In the experiment, it was investigated whether the clustering on top of the proposed features can distinguish between play sessions conducted by humans versus those generated by the artificial player. The idea here was to see whether the metrics capture the difference between the stories since the artificial player was programmed to play the game so the outcome differs from a set of input stories.

Experiment 3. In the experiment, it was investigated whether the features proposed can cluster together MOSS stories sharing the same moral. The motivation was to see whether the general features proposed scale to another domain.

Evaluation Methodology

As there is no generally accepted method for evaluating the quality of clustering independent of the application, an ad hoc method was used. The goal in this stage was to confirm whether the k-means algorithm with a specific distance metric is capable of detecting some of the meaningful features of the input stories, i.e. whether it would group stories with similar values of the feature in one cluster.

First and the most important feature of the story was defined by the story ending (both levels feature multiple endings). The question then was: "Do stories grouped in one cluster end the same when using a particular input distance metric?" This is represented by precision of the metric. To obtain precision, the most frequent ending within each cluster was identified. Then the cluster precision was defined as the fraction of the stories with this ending inside the cluster. This can be generalized as follows.

Evaluating the Clustering Quality. Intuitively, a clustering is good, if stories in the same cluster have many features in common. Let there be a feature function f:$S \rightarrow V$, where S is the set of all possible stories and V is a finite set representing possible values of a feature the designer might be interested in.

For a cluster $X \subset S$ the *precision with respect to f* is defined as the proportional size of its largest subset sharing the same value of the feature:

$$precision(X,f) = \frac{max\{|M| : M \subset X, \forall m,n \in M : f(m) = f(n)\}}{|X|} \tag{4.10}$$

In other words, precision of 0.62 means 62% of stories in the cluster produce the same value for f. The precision of the whole clustering is simply the average of per-cluster precisions. A system that clusters stories can be considered useful, if it provides high precision across multiple domains and multiple features.

In SD Two, it was also inspected how much time (in seconds) Thomas interacts with girls and how much time he stays alone (Experiment 0). If the clustering works well with respect to this feature, it would be expected the stories in the same cluster to share similar values, e.g. Thomas interacts with Barbara in each story from one group roughly the same time. To measure that, time deviations of a cluster were defined that are the standard deviations of the respective times gathered from stories in that cluster. E.g. a cluster with two stories where Thomas interacts with Barbara for 60 and 120 seconds has higher deviation than cluster with three stories where the respective times are 110, 120 and 130 seconds. The lower the deviation the more "similar" the stories are.

In Experiment 2, the feature for measuring the precision was whether the story is generated by a human or the artificial player.

In MOSS (Experiment 3), the feature for measuring the precision of the clusters was the moral of the story. Recklessness, retribution and reward morals were selected to keep the number of the target clusters roughly the same as for the experiments conducted on SD One and SD Two domains.

To summarize, the k-means algorithm was run with four story distance metric on three domains with different resulting number of clusters multiple times. Story distance metric for k-means were tension curve distance and Levenshtein [Levenshtein, 1966], Jaccard index [Jaccard, 1901] and Jaro-Winkler [Winkler, 1990] distance on action strings and sub-scene strings. It was compared how similar are stories in resulting clusters according to four features – story ending (Experiment 0 and 1), time the characters spend together in the story (Experiment 0), whether the story is generated by human or artificial player (Experiment 2), and the story moral for the MOSS stories (Experiment 3). As a baseline, a random cluster assignment where the clusters were not determined by k-means but every story was assigned to a cluster at random was used. This was repeated 100 times in the Experiment 0, 1, 2 and 3 and the results were averaged.

Note that to provide a more robust evaluation of the methodology, it would be best to measure precision with respect to similarity of stories as perceived by humans. However, that poses multiple methodological issues. From the author's perspective, a biggest obstacle to human evaluation is finding a useful dataset. Since humans cannot effectively cluster more than a handful of stories, the dataset needs to be small, which is usually unsuitable for machine clustering as the algorithm can easily pickup artifacts in the data.

4.5.2 Experiment 0

This Section is based on [Bída et al., 2013] page 7-10 and presents revised results.

In this experiment, the quality of clustering was determined by the stories in the cluster sharing the same ending. The analysis was performed on two datasets. First dataset was 1135 play sessions of game SD One generated automatically with user input being simulated by a random algorithm (see Section 4.4.5 for more information about the artificial player). Second dataset was 41 human play sessions of game SD Two which exhibits more complex domain with more game endings.

SD One Results and Discussion

From 1135 play sessions of SD One, 608 stories ended with characters getting angry with each other and parting, 479 ended with characters being too positive and not reaching the cinema, 16 stories ended with characters getting to the cinema and 32 stories endings were undefined (because of some technical issues). The low number of "cinema" stories is not surprising as it is necessary to "steer" the conversation between agents intelligently to assure they will get to the cinema – a thing that the artificial player did not account for. Since there are three possible endings, the clustering was done into 2 – 4 clusters.

Two SD One stories with intimate ending
AABDBHHIMOIOIOJDFLCCDYHGLQPHPHPHPHPHPHPHPHPHPHPH-LHQDFGIODFGIOODYGLHIDMHNMNPHPHPHPHPHNHIOU-**UUUUVQWDP**
AABDBHHMNIOIODFGLLCCPHPHPHPDMHNU**UUUUVQWDP**
Two SD One stories with breakup ending
AACPDMADPDD**ZG00123**
AABDBHHMNIOMNJ6LCDMHMCMHNIOTLDTHMNMPHPHPHPHPHP-HPMHNbDTTM**ZG00123**

Table 4.1: Experiment 0 – SD One stories action strings examples. The figure shows action strings of two stories with intimate ending and of two stories with breakup ending. Notice that the last few actions are the same for intimate endings and breakup endings (bold).

The results indicate that all of k-means distance metrics were able to cluster the simpler domain of SD One with high cluster precision (Fig. 4.2). Quite surprising is the high value of precision for Jaccard index (average precision of 0.95 for all the clustering runs). Jaccard index compares only the number of same actions in both samples disregarding ordering (see Section 4.3.2). One of the causes of Jaccard index doing well is that two of the endings had pre-scripted final action sequences (Table 4.1) which biased the string distance metrics. Because of this, all pre-scripted actions at the end of the stories were removed for the experiments in the SD Two domain. Other reason of string metrics doing well in the scenario in general is the relative simplicity – more negative actions in the scenario meant the probability of negative ending increased and vice versa. These straightforward influences created a bias favoring the Jaccard Index in SD One.

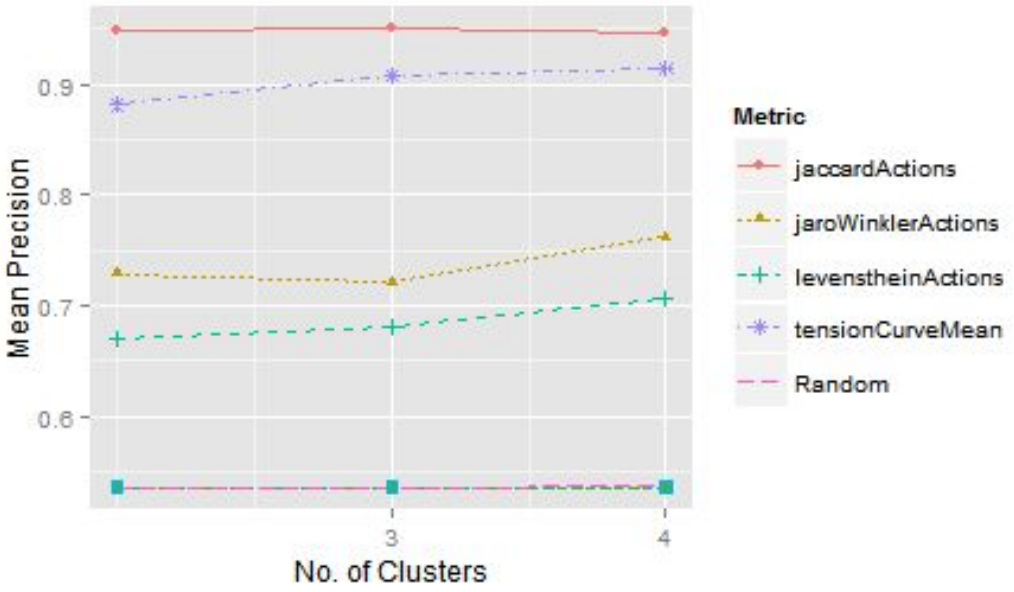


Figure 4.2: Experiment 0 – SimDate3D Level One clustering results. Cluster precision weighted averages can be seen for two, three and four clusters. The results are averaged over 100 clustering runs with different initial cluster positions.

The tension curve scored the second best with the average precision rating of 0.90 across all the clustering runs. Upon closer inspection of the tension values of the scenarios (see Fig. 4.3), it can be seen that the tension values of scenarios with various endings overlap. This is a side-effect of the endings definitions – two stories with similar beginning can end differently based on a rather short sequence of actions performed later in the scenario. Despite that, the tension curve clustered the stories with high precision.

All metrics were significantly better than random cluster assignment.

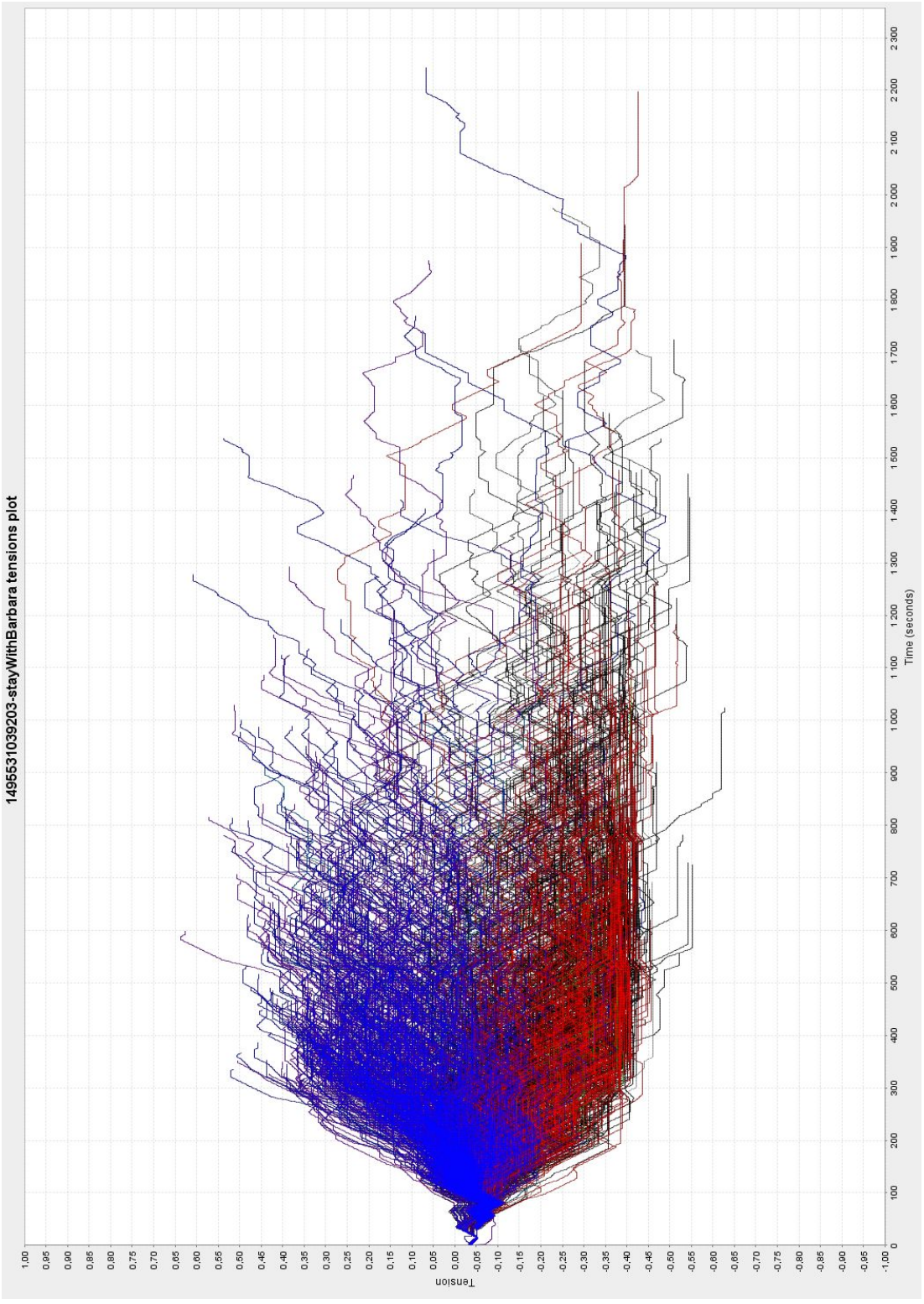


Figure 4.3: Experiment 0 – SimDate3D Level One Tension Values Visualized. Blue color corresponds to the scenarios ending with characters skipping the cinema, red color represents scenarios where the characters broke up and the green color represents scenarios where the characters made it to the cinema.

SD Two Results and Discussion

From 41 play sessions of SD Two, 14 stories ended with Thomas staying with Barbara, 21 stories ended with Thomas staying with Nataly, 4 stories ended with both girls breaking up with Thomas and 2 stories ended with Thomas staying in relationship with both girls. Since there are four possible endings, the clustering k was set to 3 – 5. More clusters were not tried because even with 5 clusters present some of them were almost empty due to lower initial number of stories. Example action strings for level two are shown in Table 4.2.

Two SD Two stories with "stay with Barbara" ending
ABACDBDEF4APDQDNAOD1AkDEDrACA9D5DQDNAOAPACDEA2D2-A2DQDNAPAOD1ACAKDKALDQDNAPDMACANDQDNAPAOACAPDQ-DNAOFQDNACAJAPDQDNAOFQDQF4D1
ABACDBDEF4DKAkDkAkDkAkDkDQDtAtAPABDBAGDEAPDQAGDED-lAlD1AID1AmDrAJD1A9FQFcF4
Two SD Two stories with "stay with Nataly" ending
DBDEABACF4APDQDnAnDBABDEAqD1AkDkAkDkAkDIAIDIAIDID-QDhAMAPD1A9DQDhAqA2AgAPDQDhAgFQDQDhAgFBFQDQAq-F4DEAPDL
F4ABACDBDEAmDJAkDQDtAtAPABDBAGDEAPDQAGDEDQDjAjAPA-BDBATDEAPDQATDEDJAmDmAlDlAlDlAlDlDQDhAMAPFQDQ-DhAgFBFQATDQAKF4

Table 4.2: Experiment 0 – SD Two stories action strings examples. Notice the sequences are longer than SD One sequences. There are two reasons for that: a) SD Two takes longer to complete and thus more actions are performed and that b) in level two each action is coded by two letters, first letter specifies the character that performed the action (D – Thomas, A – Barbara and F – Nataly), second letter specifies the action, e.g. "AB" marks Barbara performing action "set focus".

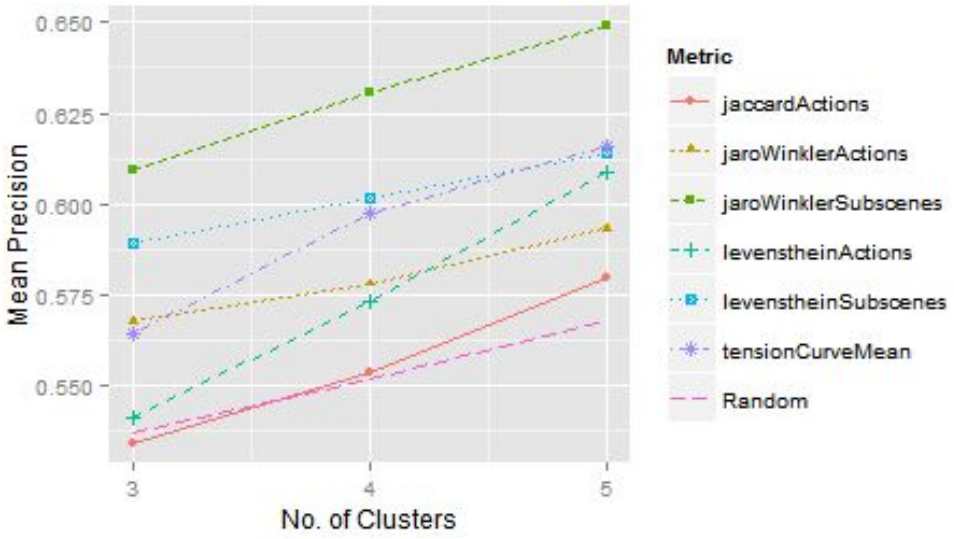


Figure 4.4: Experiment 0 – SimDate3D Level Two clustering results. The plot shows Jaro-Winkler on sub-scenes outperforming the other metrics. This suggests that sub-scene strings is an useful abstraction generalizing the stories well. The tension curve scored second and third place based on the number cluster generally outperforming string distance metrics based on actions. All metrics scored significantly better than the random cluster assignment.

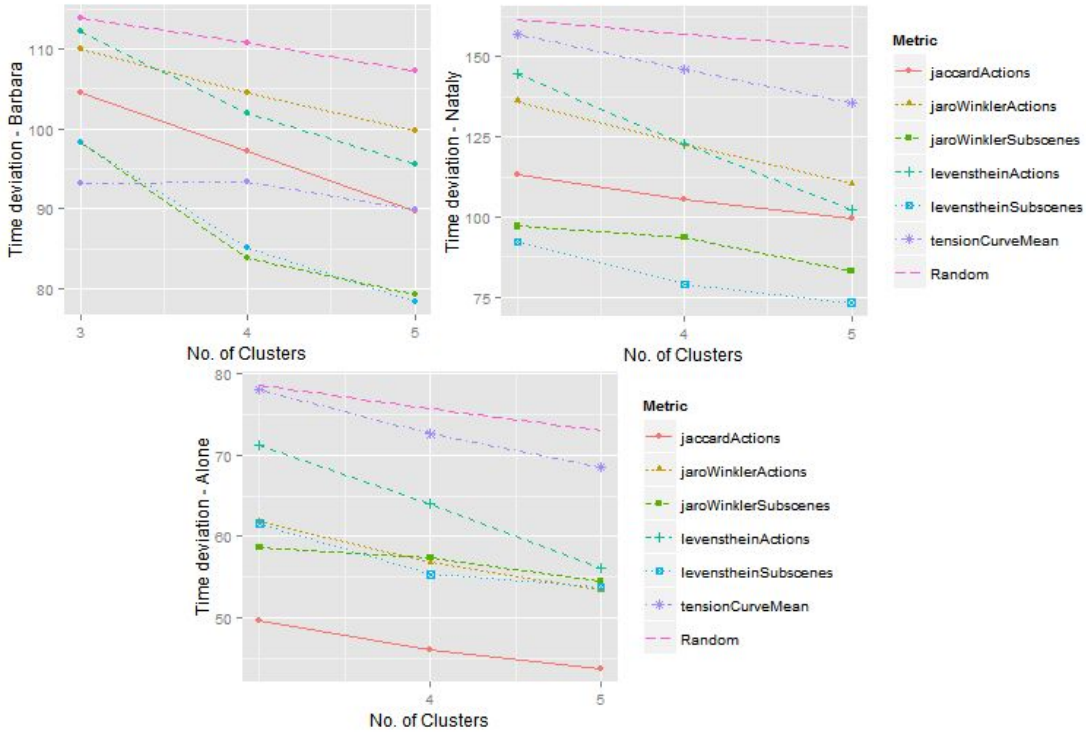


Figure 4.5: Experiment 0 – SimDate3D Level Two Time Deviations. Time (in seconds) deviation plots for Barbara and Thomas interaction (upper left), Nataly and Thomas (upper right) and Thomas alone (bottom) show that all metrics are doing better than a random assignment (lower deviation means the stories are closer to each other in this sense).

The preliminary results on SD Two show a drop in cluster precision of all metrics indicating the higher complexity of the domain. The Jaro-Winkler distance based on the sub-scene strings demonstrated the highest precision of 0.65 for five clusters. The mean precision of curve metric for five clusters was 0.62, while random clustering had precision of 0.57.

The time deviations of all metrics were lower than that of random cluster assignment (see Fig. 4.5), however, the ordering of the metrics differ based on the time deviation measured not allowing for any strong conclusions.

Summary

The string distance metrics applied to sub-scene strings scored the best suggesting the sub-scene string representation is a good abstraction of the stories and that it represents a meaningful high-level feature. The tension curve scored better than the string distance metrics applied to action strings suggesting it scales better. The problem of the string distance metrics on action strings might be that they do not make any abstraction of the story from the qualitative point of view and may be misled by action sequences that look different but are not different from the story perspective (e.g. talking about weather and talking about yesterday lunch both represent casual conversation but have different action sequences). Moreover, the action sequences contain actions that may not be directly related to story tension, e.g. move actions and set focus actions. One of the ideas how to improve the performance could be to prune actions like this from the sequence.

The Experiment 0 results indicate that the tension curve might scale to other domains but more experiments are needed to confirm this. The string metrics applied to action strings were able to cluster SD One domain reasonably well, however, they did not perform well applied to SD Two domain.

The sub-scene string representation turned out to be a promising direction as in SD Two domain it produced the best overall results (with Jaro-Winkler distance string distance). To get more insight into how the methodology performs, there were additional experiments conducted to shed more light onto this.

4.5.3 Experiment 1

This Section is cited from [Bída et al., 2015] page 3 with modifications.

In the experiment, a more thorough analysis of the similarity metrics regarding the game ending and the similarity of stories was conducted on the domain of SD Two. An extended dataset of 70 human play sessions of SD Two was analyzed using additional feature – condensed tension difference curve based on sub-scenes. Precision is measured with respect to the ending of the story. A graph of the results is presented in Figure 4.6.

It can be seen that the tension curve outperforms other approaches in mean precision (0.6 for three clusters to 0.63 for five clusters). The interesting observation is that the sub-scene string sequence (metrics marked as "Subscenes" on Figures 4.6, 4.8, 4.9) outperform almost all of action strings based methods (metrics marked as "Actions" on Figures 4.6, 4.8, 4.9) on this dataset which is in line with the results from the previous experiment.

This further promotes the notion that sub-scene sequence is a meaningful feature in SD domain, relevant to story ending. Also note that Jaro-Winkler

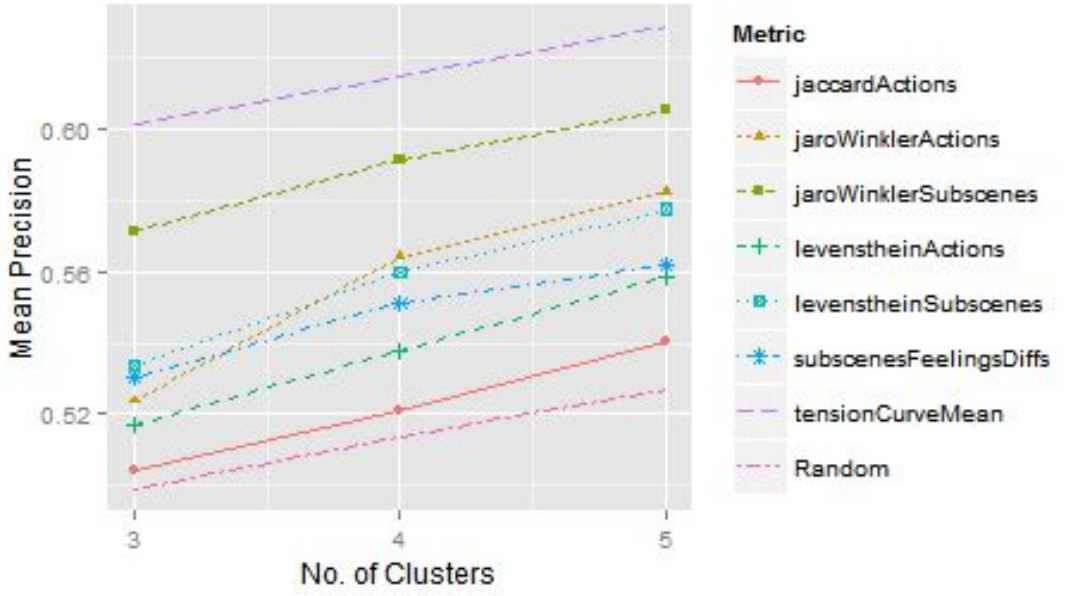


Figure 4.6: Experiment 1 – SimDate3D Level Two clustering results. Cluster precision weighted averages can be seen for three, four and five clusters (this is chosen arbitrarily based on that there are four possible endings). The results are averaged over 100 clustering runs with different initial cluster positions. The precision is calculated with respect to story ending.

distance on sub-scenes (average 0.59) slightly outperforms Levenshtein (average 0.56). This is somewhat unexpected as Jaro-Winkler distance could be regarded as a sub-par choice for clustering as it does not satisfy the triangle inequality. However, the distance gives more weight to differences between first four characters of the string. The good performance of Jaro-Winkler on sub-scene sequences may then be explained by a large impact of the beginning of the story on its ending. Assigning higher weight to story start and/or story end might be an interesting extension of the approach as it would reflect the way stories are perceived by humans[15].

The compressed tension difference curve (metrics marked as "subscenesFeelingDiffs" on Fig. 4.6) scored on par with action strings distance metrics (average 0.55), but did not match the uncompressed original tension curve.

All metrics scored significantly better than the random cluster assignment. However, compared to previous results in section 4.5.2 the addition of more stories did not result in big precision value changes for the tension curve and Jaro-Winkler distance on sub-scenes – in fact the precision of these even slightly increased. These might be an indication that the metrics scale well.

Examples of the stories from this dataset and their clustering can be found in Appendix E. Visualization of the tension values of the stories can be seen in Figure 4.7.

[15]For example, Kahneman [Kahneman, 2013] claims that humans use "peak-end" rule when evaluating their past memories putting more weight on the beginning, the end and the most emotionally intensive moment of the experience.

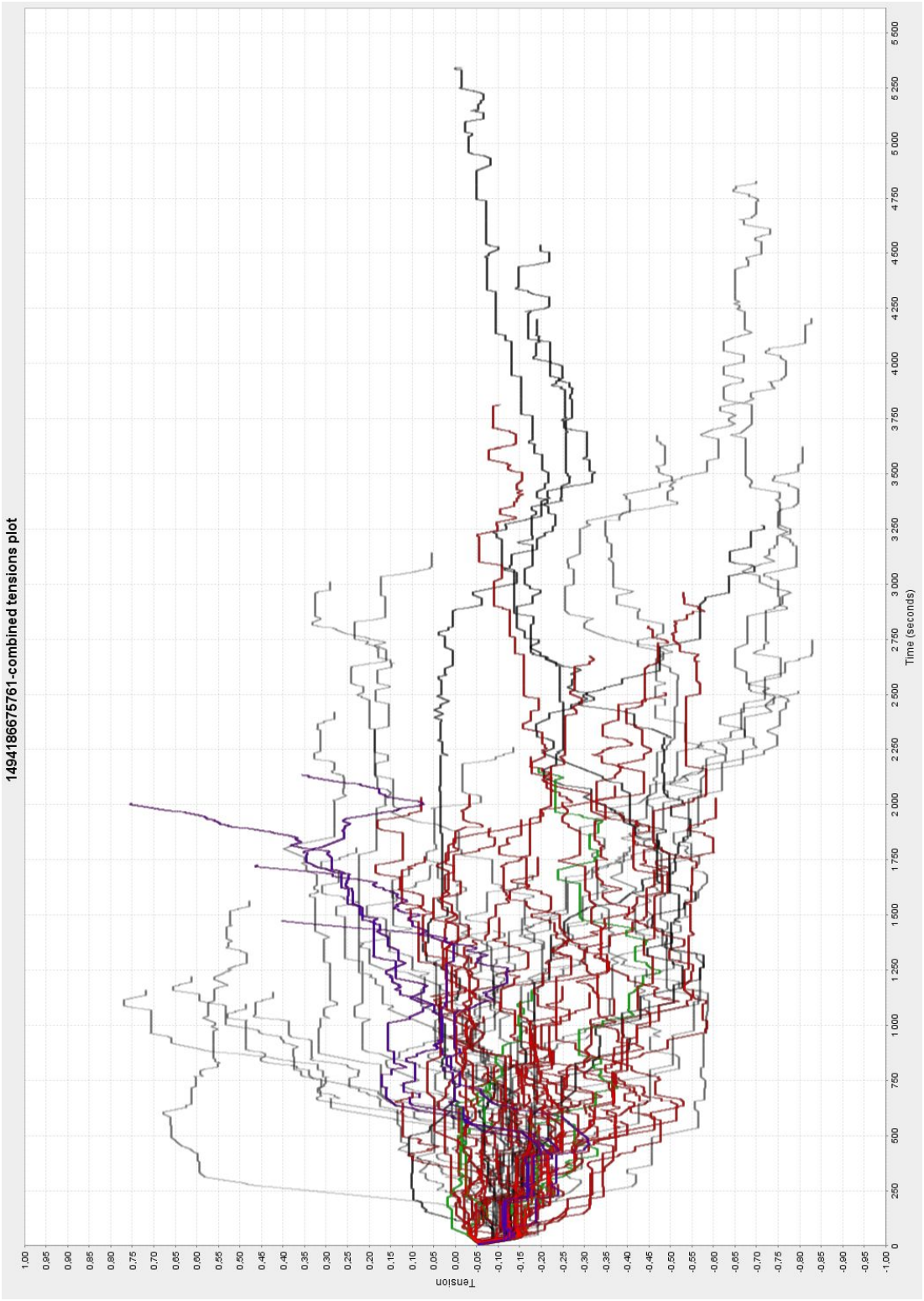


Figure 4.7: Experiment 1 – SimDate3D Level Two Tension Values Visualization. The scenarios ending with both girls leaving are visualized in blue. The scenarios where all three characters stay together are visualized in black. Green represents scenarios which feature Thomas and Nataly staying together and red feature Thomas and Barbara staying together. The tensions of different endings show some overlap.

4.5.4 Experiment 2

This Section is based on [Bída et al., 2015] page 3-4 with modifications.

The goal of this experiment was to see whether the general features proposed can be used to distinguish play sessions of human players (N=41) versus play sessions of the artificial player (N=60). Note that the SD Two implementation of artificial player was used (Section 4.4.5). The idea is to see whether the clustering is able to pick up the difference here, since the artificial player is set in a way to avoid the sub-scene sequences defined by the human play sessions. Hence, the precision of the clustering is measured with respect to the type of the user that generated the story. A graph of the results is presented in Fig. 4.8.

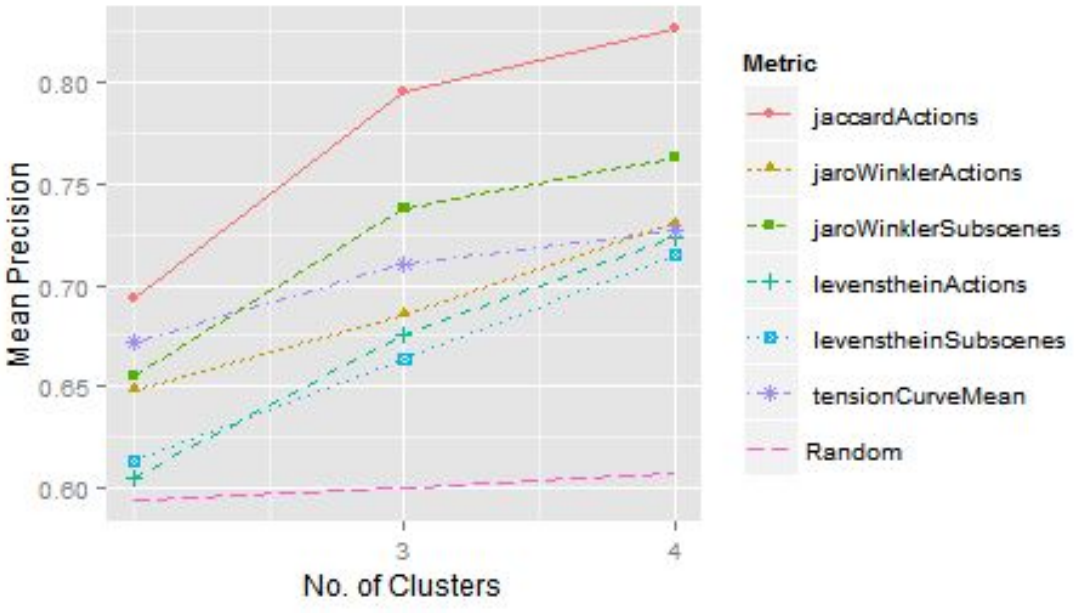


Figure 4.8: Experiment 2 – SimDate3D Level Two clustering results. Figure shows the average precision of clustering with respect to the users that created the stories as a function of a number of clusters.

The best metric for distinguishing between human and artificial player is the Jaccard Index on actions (with precision 0.69 on two and 0.83 on four clusters). Jaro-Winkler distance on sub-scenes scored the second best (with precision 0.66 on two and 0.76 on four clusters). The good performance of the Jaccard Index is probably caused by the way in which the artificial player chooses actions. The artificial player actions are chosen at random but with fixed random weights. Also, the choice of an action is influenced by the place the agents are at. This leads to the artificial player generated scenarios sharing larger amount of actions than human generated stories as the human's action choice is much less deterministic.

The performance of Jaro-Winkler on sub-scenes can be explained again by the feature of the algorithm putting more weight on the first characters of the string (first four in the settings used). The artificial player tries to achieve a different set of sub-scenes (see Table 4.3) – the first thing done is a change of the initial sub-scene to some other resulting in the different "beginnings" of the stories regarding the sub-scenes. This was picked up by Jaro-Winkler resulting in a good performance of the algorithm.

Three sub-scene strings from stories created by the human users
B0BCB0BRBN0
B0BCB0BN0
B0BC0N0NRN0BN0
Three sub-scene strings from stories created by the artificial player
0H20H1BH1B0BN0
0H2NH20B0BJ0BNRBN0
0NH2N0NH20H1BH1

Table 4.3: Example of sub-scene strings from SD Two stories. Notice the difference at the beginning of the sub-scene strings – the artificial player tries to elicit different sub-scenes than represented by the sequences of the human players.

The tension curve performed a bit worse than the sub-scenes based metrics on this task (average 0.7) but still ended up on a second or third place. This is understandable as different sub-scene sequences in the story may produce similar tension curves. Also, it seems that different metrics are able to represent different features of the stories well. Overall, it indicates that the problem of similarity of the stories is multi-layered and to grasp this properly a combination of features is likely to be required.

Discussion

Upon closer inspection of the clusters generated by the various distance metrics, it was found out that when the number of clusters is set above 2, one cluster is usually entirely dedicated to stories generated by the artificial player (Jaccard Index and Jaro-Winkler sub-scene metric seems to be the most robust clustering together stories with length above 2 minutes). Also, almost half of the stories (N=28) generated by the artificial player are rather short meaning the argument between the characters happened in the first two minutes of the scenario. These short stories might be harder to distinguish from those created by human players – shorter stories mean less data that can be picked by the features and the clustering.

4.5.5 Experiment 3

This Section is cited from [Bída et al., 2015] page 4 with changes.

In the Experiment 3, the goal was to see whether the methodology scales when used on a different domain – represented by the MOSS system. The MOSS system generates stories with morals represented by a set of Prolog facts. There are multiple domains – animals, fairy tales and family and multiple morals. Moreover, the MOSS stories can be generated with dramatic length two and four[16].

[16]Dramatic length corresponds to a number of actions in the story that are dramatically relevant – are moving the story forward, e.g. "a dragon kidnapped the princess" and "a knight slayed the dragon" might a be a story with dramatic length two.

For the purpose of the experiment here, six millions of stories were generated for the fairy tale domain – three million with dramatic length two and three million with a dramatic length four. From the three million – a million stories contained the recklessness moral, a million of stories contained the retribution moral, and a million of stories contained the reward moral. As this dataset was very large a sub-set of stories was selected at random. In total, 3000 stories were analyzed – 1000 stories for each moral. Half of the stories comprised of two dramatic actions, and the other half comprised of four dramatic actions. In both cases, the resulting stories contained about 30 atomic actions. The precision was measured with respect to the moral of the story. A graph of the results is presented in Fig. 4.9.

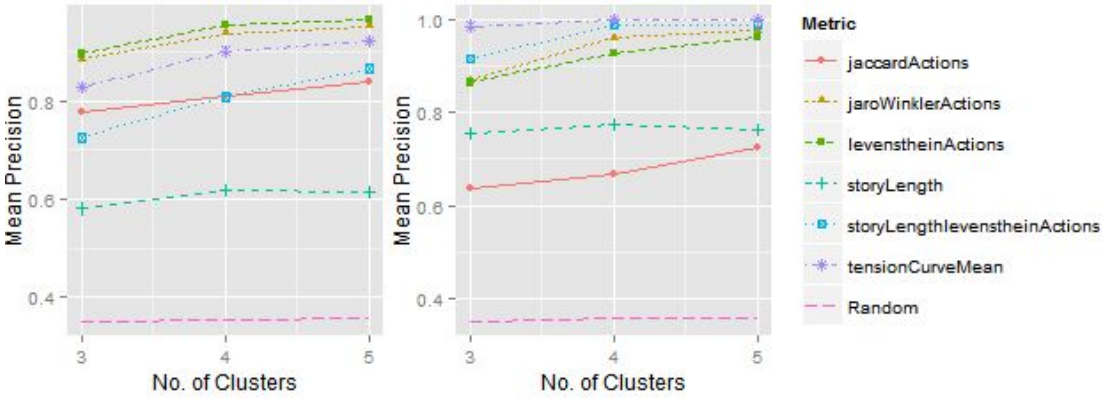


Figure 4.9: Experiment 3 – MOSS domain clustering results. On the left there are precisions of clustering for three, four and five clusters when distinguishing between stories of the dramatic length two with particular moral. On the right there is the same for stories with the dramatic length four. All results were averaged over 100 clustering runs.

It can be seen that the precision of clustering is very high for almost all clustering metrics. For MOSS stories of length four, tension curve achieved precision of 0.99 on three clusters. The sum of normalized story length and Levenshtein on action strings was the second best scoring 0.93 on three clusters. On MOSS stories with length two, these two metrics performed a bit worse. The best was Levenshtein on action strings which averaged on 0.94 and the tension curve with 0.88 precision on average. The story length metric was outperformed by almost all other metrics and it also did not bring significant improvements to the Levenshtein distance indicating that the MOSS generating process did not produce artifacts in story length.

Examples of the stories from this dataset and their clustering can be found in Appendix E.

Discussion

This overall good performance is caused by the fact that stories in MOSS are generated through templates that use emotional patterns. Stories in one domain exhibit the same or very similar emotional patterns resulting in similar tension curves. This is picked by the tension curve metric really well. The comparable

performance of string metrics on action strings is likely caused by the presence of emotional actions in the action strings. The overall slightly worse performance on stories with dramatic length two is probably caused by the fact that less dramatic actions in the story offer less space to distinguish the stories from each other (however the performance was still remarkably good).

4.5.6 Summary

Four experiments conducted in this Chapter were used to evaluate a solution for semi-automatic evaluation of interactive storytelling systems based on clustering of similar stories. Results from Experiments 0, 1, 2, and 3 showed that the methods can be transferred successfully to other domains. However, it needs to be taken with a grain of salt as the MOSS story generator abstraction was very favorable to the method as it uses emotional patterns to define the categories of the stories. On the other hand, there is a good chance that any complex IDS will contain an explicit model of emotions as discussed in this Chapter.

Next, we have used sub-scene sequence feature of the stories in the implementation of the artificial player designed to explore unvisited parts of the story space of SimDate3D domain and we have shown the performance of the method on a distinguishing artificial player from the human players. Some of the metrics were able to cluster together stories generated by the artificial player showing the metrics can, to some extent, pick up a different playing style. A combination of features could further increase the precision in this task. Alternatively, a modification of the artificial player code to circumvent short-length stories might prove helpful. Nevertheless, the semi-automatic exploration of the story space with artificial player proved useful and remains an interesting research direction potentially helping the developers understand large story spaces of IDS systems.

In line with the results from the first round of the evaluation (Experiment 0), the tension curve provided robust results across domains and feature functions and outperformed the other metrics in Experiment 1. A combination of tension curve and one of the string distances might prove useful. Other future work includes experiments with combination of distance metrics for the clustering algorithm (such as using k-medoids clustering) and further enhancements and additional experiments with an artificial player. It would be also beneficial to experimentally determine how humans would cluster some of these stories. However, first preliminary results indicate that it might be tricky to come up with a story abstraction enabling fast and precise clustering by humans[17].

The number of clusters was determined arbitrarily in the experiments above based on the number of target classes. However, there are methods such as hierarchical deterministic annealing [Rajagopalan et al., 1999] that can be used to determine the ideal number of clusters on the fly. It might be beneficial to try to combine said approach with the clustering methodology proposed in this Chapter.

[17] As a preliminary experiment, short comics representing the stories were experimented with. However, when it came to clustering, the presence of noise was higher than normal as the participants were not sure what exactly is happening in the story based only on the comic-like strips.

4.6 Discussion

This Chapter researched the problem of semi-automatic drama analysis. The outcome is a set of general story features that should help to cluster the stories according to their similarity. Clustering the stories help the developers to get a sense of the story domain of their system helping them to develop and debug the system, mitigating the challenges of the *wicked problem* of the IDS.

The approach presented in this Chapter was tested on three distinct domains showing that the performance of the methodology is, to some extent, able to grasp the notion of similar stories even though it is quite general in nature.

One of the potential problems that was encountered was the definition of similarity itself. In Experiments 0 and 1, similarity was defined as "the stories that end the same are similar". This definition might be problematic. The way the game mechanics was set up leads to a potential case where two stories can be completely similar to a large extent differing only in the last few actions. However, that can lead to the stories having different endings despite being very similar. Nevertheless, the overall results indicate that while this probably affected the precision values the metrics were still able to cluster the space reasonably well.

Another issue is the story length – this feature might be useful in general but applied to story endings it yielded a little gain as the story length usually does not determine the story ending (certainly not for SD Two domain).

Apart from using the story length as first class general feature of the stories, there are additional opportunities. For example, a finer grained tension curve that would be generated per agent, e.g., three agents in the environment would mean three distinct tension curves, each generated from emotions of a particular agent. Even more finer grained feature might be the emotions of the characters themselves. Using emotions as features might provide a richer and a more precise abstraction of the stories, resulting in higher precision of clustering in general.

Another improvement of the methodology might be to generate the tension not from the character's emotions but directly from human emotions. This should provide more precise tension values. Technically, it is possible to train emotion recognition algorithms using facial expressions[18] or by combining data from multiple biofeedback sensors such as breathing frequency and depth, heart rate, skin conductance and EEG and EMG[19] [Koelstra and Patras, 2013], [Clerico et al., 2016] – although, there are many technical obstacles that need to be addressed.

The general nature of the methodology opens up another potential research direction which is to conduct drama analysis on the text of real stories written by human authors. With the advancement of sentiment analysis technologies that are now able – to some extent – to extract emotional features of the text [Veselovská and Tamchyna, 2014], it is possible to use that data to obtain the tension curves directly from the written stories and a) analyze them according to these features, and/or b) cluster them together based on these features. There

[18]There are publicly available services such as Google Vision API at cloud.google.com/vision [27.12.2016] or Microsoft Emotion API at www.microsoft.com/cognitive-services/en-us/emotion-api [27.12.2016].

[19]EEG stands for Electroencephalography where the electric activity of the brain is measured and EMG stands for Electromyography where the electric activity of muscles is being measured. Both of these measurements can be related to emotions.

are large online corpuses of stories written by human such as Fanfiction[20] with millions of stories to analyze. Note that the analysis of these stories might not be limited to tension curve extraction but more basic language traits could be also analyzed such as the occurrence of adverbs. Fanfiction stories are rated by users which makes the analysis even more interesting – for example by answering the question "What makes a story an interesting read?".

Overall, a promising research direction turned out to be the artificial player. The artificial player yielded stories with different sub-scenes sequences in the Experiment 2 which was picked up by the proposed metrics. This confirms that the artificial player can be used for exploring the story space of IDS systems. Helping developers to automatically test a particular scenario within a prepared environment might be a useful approach mitigating the challenges when testing systems with large-scale story domains.

[20] Available at URL: www.fanfiction.net [27.12.2016].

Conclusion

The world is changing and the stories within are changing as well. This thesis researched the problematics of stories and Interactive Digital Storytelling (IDS). It discussed how the stories evolve with respect to new media such as computer games and new advances in the IDS field. We have introduced the reader to the field of IDS, defined how is it connected with the problematics of *believability* and affect simulation for IVAs and showed some examples of working systems.

The *wicked problem* of IDS was defined and a *good enough* solution with regards to medium-sized drama with Emohawk Agent Architecture (EWA) was presented. EWA was used in SimDate3D Level One and SimDate3D Level Two that were developed as part of this thesis. The thesis also extended the Pogamut platform by connecting the virtual environment of UE2 with a peaceful graphical content of EmohawkVille city including dozens of different virtual characters. This allows the developers to easily create IVAs in non-violent daily life scenarios[21].

All of the above steps were necessary to tackle the problematics of semi-automatic drama analysis of narratives produced by IDS systems in the last Chapter of this thesis. General features for story abstractions and mechanism for clustering of similar stories were proposed and evaluation on three IDS systems – SD One, SD Two and MOSS was conducted. It was shown that tension curves, sub-scenes and action strings are a useful high-level abstraction of stories and that the proposed methodology scales to multiple domains.

In particular, the following goals were set and achieved in this thesis:

G1. *Design a minimalistic agent architecture that aims to create a believable behavior of the NPCs in computer game-like environments.*

G2. *Find and use a 3D virtual environment that would allow us to experiment with NPCs engaging in social situations.*

G3. *Implement an IDS system on top of the agent architecture that will contain NPCs living in the 3D environment and that will produce stories that can be analyzed.*

G4. *Design and implement a methodology for estimating the quality of virtual drama produced by a game or an IDS system.*

The following contributions were made in this thesis pertaining to the goals above:

C1. EWA – a minimalistic agent architecture with an integrated emotion model and a navigation manager allowing for plausible navigation of characters in the virtual environment. This achieves the Goal 1 of the thesis.

C2. Connection of UE2 environment to the Pogamut platform, broadening the range of virtual environments that are available in the Pogamut platform. This was done by the author of this thesis in scope of the StoryFactory project. This achieves the Goal 2 of the thesis.

C3. SD One and SD Two – two narrative games in 3D environment using EWA for controlling the NPCs featuring two characters (SD One) and three (SD

[21] Pogamut is normally used to prototype IVAs in action first person shooter game Unreal Tournament 2004 [Epic Games, 2004].

Two) characters in social situations. This reaches the Goal 3 of the thesis and allowed to pursue the Goal 4.

C4. A methodology for estimating the quality of virtual drama that is based on abstracting the important story features that can be further analyzed, e.g. by methods of clustering. This reaches thesis Goal 4.

C5. As a secondary contribution, StoryFactory was implemented – a tool enabling creation of short movies in 3D game environment that can be used to mock up dramatic situations in 3D environment with IVAs and should promote the education in the field of IVA and IDS. StoryFactory contributed to Goal 2 and 3.

Advantages of C1 are that it can serve as an convenient starting point for computer game developers interested in IDS. Unlike other approaches, C1 is minimalistic and uses concepts that closely resemble the concepts used in the game development community such as Finite State Machines or Behavior Trees. C1 can be also used to quickly prototype medium-sized IDS systems helping the IDS community to test various approaches to the IDS development.

C2 and C3 extend the Pogamut platform with a new environment with peaceful graphical content that can be used to further promote quick prototyping of simple games taking place in the virtual city of EmohawkVille.

Moreover, C1, C2 and C3 enabled further research leading to C4. C4 presents a novel approach for semi-automatic drama analysis, combining mechanisms for abstracting the stories by tension curves, action strings and sub-scenes with clustering techniques that categorize the stories into clusters of similar stories. C4 can potentially help the developers to make sense of large story spaces of IDS systems, improve their scalability and enable the development of larger and richer IDS systems.

C5 is useful when prototyping dramatic situations in virtual environment or for teaching the basics of virtual agent curricula – such as movement and animations.

We are entering a new era of AI systems, a new era of media and a new era of games. What will happen in the following years is written in the stars but the author believes that with the recent advancements in AI and virtual reality technology, there is much to look forward to when it comes to digital entertainment, computer games and the IDS field.

Acknowledgements

The research related to EWA was supported by the Ministry of Education of the Czech Republic (Res. Project MSM0021620838), by a project P103/10/1287 (GACR), by a student grant GA UK No. 0449/2010/A- INF/MFF, and partially supported by SVV project number 263 314. The name “Emohawk” is inspired by Emohawk: Polymorph II, an episode of Red Dward VI (BBC). The graphical content was created by I. Diosi and Z. Krulich using Mayang’s Free Textures library: http://mayang.com/textures/.

StoryFactory development was supported by the project CZ.2.17/3.1.00/31162 that is financed by the European Social Fund and the Budget of the Municipality

of Prague. The research related to this application was also supported by the project P103/10/1287 (GACR), by a student grant GA UK No. 0449/2010/A-INF/MFF and partially supported by SVV project number 263 314. StoryFactory graphical content was created by Ivor Diosi, Zbyněk Krulich and Michal Červenka using Mayang's Free Textures library: http://mayang.com/texturesMayang [1.11. 2011] and RocketBox 3D Animations Package: http://rocketbox.de [3.3.2012]. We also thank to Markéta Tomková (formerly Popelová), Jakub Tomek, Jan Vyhnánek, Jan Havlíček, Cyril Brom and Edita Bromová (formerly Dufková) for their contributions.

Steering research was partially supported by project P103/10/1287 (GACR), by a student grant GA UK No. 0449/2010/A-INF/MFF, by SVV project number 263 314 and the research project MSM0021620838 of the Ministry of Education of the Czech Republic.

The virtual quarrel research was supported by the project P103/10/1287 (GACR), by a student grant GA UK No. 0449/2010/A-INF/MFF, by student grant GA UK No. 655012 and partially supported by SVV project number 265 314.

First battery of experiments in drama analysis section (Experiment 0) was partially supported by the student research grant GA UK 559813/2013/A-INF/MFF, by the SVV project number 267 314 and by the grant P103/10/1287 from GAČR.

Second battery of drama analysis experiments (Experiment 1, 2 and 3) was partially supported by SVV project number 260 224 and by a grant GA UK No. 559813/2013/AINF/MFF.

Apart from the grant projects above, many other people participated in the research endeavor surrounding this thesis and supported me during the years. I would like to thank my supervisor Cyril Brom for giving me the opportunity to work on not-exactly-mainstream projects and research. I would like to thank Jakub Gemrot for his drive and inspiration that lead to Pogamut 3 which started everything.

I would like to thank Rudolf Kadlec for all of his cynical comments that helped me to move forward – Ruda, sharelatex is great. I would like to thank the rest of the Pogamut team – thanks guys, we have achieved something. I would like to thank Markéta Popelová for her enthusiasm and research effort – it was a pleasure to supervise your thesis. I would like to thank Martin Černý, Veronika Stankovianska, Jolana Žižková and Martin Labský for proof reading and all their comments how to make the thesis better. I would like to thank my parents for their constant support and belief in me. I would like to thank Eva for all those beautiful years and her patience.

Last but not least, I would like to thank Jan Kleindienst and Martin Labský from IBM Watson Research Lab in Prague – without your belief and your support I would not be able to finish this thesis.

Bibliography

[Alias-i, 2015] Alias-i (2015). LingPipe 4.1.0. Java toolkit for text processing. URL: http://alias-i.com/lingpipe [22.3.2017].

[Aylett, 1999] Aylett, R. (1999). Narrative in Virtual Environments - Towards Emergent Narrative. In Mateas, M. and Sengers, P., editors, *Narrative Intelligence, Papers from the AAAI Fall Symposium, no. FS-99-01 in AAAI Fall Symposium Series*, pages 83–86, Menlo Park, CA. AAAI Press.

[Aylett, 2000] Aylett, R. (2000). Emergent Narrative, Social Immersion and "Storification". In *Proceedings of the 1st International Workshop on Narrative Interaction for Learning Environments*, Edinburgh.

[Aylett et al., 2009] Aylett, R., Kriegel, M., and Lim, M. (2009). ORIENT: interactive agents for stage-based role-play. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, 2:1371–1372.

[Aylett and Louchart, 2008] Aylett, R. and Louchart, S. (2008). If I were you - Double appraisal in affective agents (Short Paper). In *Proceeding AAMAS '08 Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 3*, pages 1233–1236.

[Aylett et al., 2007] Aylett, R., Vala, M., Sequeira, P., and Paiva, A. (2007). FearNot! – An Emergent Narrative Approach to Virtual Dramas for Anti-bullying Education. In *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, volume LNCS 4871, pages 202–205. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Baars and Franklin, 2009] Baars, B. J. and Franklin, S. (2009). Consciousness Is Computational: the Lida Model of Global Workspace Theory. *International Journal of Machine Consciousness*, 01(01):23.

[Babor, 2012] Babor, P. (2012). *Three virtual characters in a quarrel: an interactive model*. Bachelor thesis, Charles University.

[Bartneck, 2002] Bartneck, C. (2002). Integrating the OCC Model of Emotions in Embodied Characters. *Proceedings of the Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges*, pages 39–48.

[Bída et al., 2011a] Bída, M., Brom, C., and Popelová, M. (2011a). To date or not to date? a minimalist affect-modulated control architecture for dating virtual characters. *Intelligent Virtual Agents*, LNCS 6895:419–425.

[Bída et al., 2011b] Bída, M., Brom, C., Popelová, M., and Kadlec, R. (2011b). StoryFactory–A Tool for Scripting Machinimas in Unreal Engine 2 and UDK. *Interactive Storytelling*, LNCS 7069:334–337.

[Bída et al., 2013] Bída, M., Černý, M., and Brom, C. (2013). Towards Automatic Story Clustering for Interactive Narrative Authoring. *Interactive Storytelling*, LNCS 8230:95–106.

[Bída et al., 2015] Bída, M., Černý, M., and Brom, C. (2015). Follow-up on Automatic Story Clustering for Interactive Narrative Authoring. In *Proceedings of the 51st Convention of the AISB*, pages 37–41.

[Bída et al., 2012] Bída, M., Černý, M., Gemrot, J., and Brom, C. (2012). Evolution of gamebots project. *ICEC*, LNCS 7522:397–400.

[Bída et al., 2007] Bída, M., Kadlec, R., and Brom, C. (2007). Význam emocí pro umělé bytosti (Relevance of emotions for artificial beings). In Kvasnička, V., Trebatický, P., Pospíchal, J., and Kelemen, J., editors, *Mind, intelligence and life.*, chapter 2, pages 158–172. Vydavateľstvo STU, Bratislava, 1 edition.

[BioWare, 2003] BioWare (2003). Knights of the Old Republic. PC/Console Game. URL: http://www.starwars.com/games-apps/knights-of-the-old-republic [22.3.2017].

[BioWare, 2007] BioWare (2007). Mass Effect. Video Game. URL: http://masseffect.bioware.com/ [22.3.2017].

[BioWare, 2011] BioWare (2011). Star Wars: The Old Republic. MMORPG Game. URL: http://www.swtor.com [22.3.2017].

[Bizzocchi and Tanenbaum, 2012] Bizzocchi, J. and Tanenbaum, J. (2012). Mass Effect 2: A Case Study in the Design of Game Narrative. *Bulletin of Science, Technology & Society*, 32(5):393–404.

[Bordini et al., 2006] Bordini, R. H., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Gomez-sanz, J. J., Leite, J., O'Hare, G., Pokahr, A., and Ricci, A. (2006). A Survey of Programming Languages and Platforms for Multi-Agent Systems 1 Introduction 2 Declarative Languages. *Informatica*, 30(1):33–44.

[Bordini et al., 2007] Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons.

[Brewer and Lichtenstein, 1982] Brewer, W. and Lichtenstein, E. (1982). Stories are to entertain: A structural-affect theory of stories. *Journal of Pragmatics*, 6(5-6):473–486.

[Brom et al., 2012] Brom, C., Babor, P., Popelová, M., and Bída, M. (2012). Controlling Three Agents in a Quarrel: Lessons Learnt. *Motion in Games*, LNCS 7660:158–169.

[Brom et al., 2009] Brom, C., Bída, M., Gemrot, J., Kadlec, R., and Plch, T. (2009). Emohawk: Searching for a "good" emergent narrative. *Interactive Storytelling*, LNCS 5915:86–91.

[Brooks, 1991] Brooks, R. (1991). Intelligence Without Representation. *Artificial Intelligence*, 47:139–159.

[Bryson, 2001] Bryson, J. J. (2001). *Intelligence by Design : Principles of Modularity and Coordination for Engineering Complex Adaptive Agents*. Phd thesis, University of Bath.

[Campbell et al., 2002] Campbell, M., Hoane, A. J., and Hsu, F.-h. (2002). Deep Blue. *Artificial Intelligence*, 134:57–83.

[Campo Santo, 2016] Campo Santo (2016). Firewatch. PC Game. URL: http://www.firewatchgame.com/ [22.3.2017].

[Case, 2014] Case, N. (2014). Coming Out Simulator. Online Game. URL: https://ncase.itch.io/coming-out-simulator-2014 [22.3.2017].

[Cavazza et al., 2007] Cavazza, M., Lugrin, J.-L., Pizzi, D., and Charles, F. (2007). Madame bovary on the holodeck. In *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, pages 651–660, New York, New York, USA. ACM Press.

[Černý et al., 2014] Černý, M., Gemrot, J., and Brom, C. (2014). An AI System for Large Open Virtual World. *Proceedings of Tenth Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 44–51.

[Černý et al., 2015] Černý, M., Plch, T., Marko, M., Gemrot, J., Ondráček, P., and Brom, C. (2015). Using Behavior Objects to Manage Complexity in Virtual Worlds. *IEEE Transactions on Computational Intelligence and AI in Games*.

[Champandard, 2007] Champandard, A. J. (2007). 10 reasons the age of finite state machines is over. *AIGameDev.com*, Available Online. URL: http://aigamedev.com/open/article/fsm-age-is-over/ [22.3.2017].

[Champandard, 2017] Champandard, A. J. (2017). Behavior Trees for Next-Gen Game AI. *AIGameDev.com internet presentation. URL: http://aigamedev.com/insider/presentations/behavior-trees (2.3.2017).*

[Chatman, 1978] Chatman, S. (1978). *Story and Discourse - Narrative Structure in Fiction and Film*. Cornell University Press.

[Cheong et al., 2008] Cheong, Y.-g., Jhala, A., Bae, B., and Young, R. (2008). Automatically generating summary visualizations from game logs. *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 167–172.

[Činčera et al., 1967] Činčera, R., Roháč, J., and Svitáček, V. (1967). Kinoautomat.

[Clerico et al., 2016] Clerico, A., Chamberland, C., Parent, M., Michon, P.-e., Tremblay, S., Falk, T. H., Gagnon, J.-C., and Jackson, P. (2016). Biometrics and classifier fusion to predict the fun-factor in video gaming. In ., editor, *IEEE Conference on Computational Intelligence and Games (CIG'16)*, pages 233–240, Santorini, Greece.

[Colledanchise and Ögren, 2016] Colledanchise, M. and Ögren, P. (2016). How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees. *IEEE Transactions on Robotics*, 33(2):372–389.

[Cowling et al., 2013] Cowling, P. I., Buro, M., Bida, M., Botea, A., Bouzy, B., Butz, M. V., Hingston, P., Muñoz-Avila, H., Nau, D., and Sipper, M. (2013). Search in Real-Time Video Games. In Lucas, S. M., Mateas, M., Preuss, M., Spronck, P., and Togelius, J., editors, *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 1–19. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

[Crawford, 2012] Crawford, C. (2012). *Chris Crawford on Interactive Storytelling*. New Riders, 2 edition.

[Damasio, 2003] Damasio, A. (2003). *Looking for Spinoza: Joy, Sorrow, and the Feeling Brain*. Harvest books. Harcourt.

[Damasio, 2010] Damasio, A. (2010). *Self Comes to Mind: Constructing the Conscious Brain*. Pantheon Books, New York, 1 edition.

[Damian et al., 2011] Damian, I., Endrass, B., Huber, P., Bee, N., and André, E. (2011). Individualized Agent Interactions. *Motion in Games*, LNCS 7060:15–26.

[Dias et al., 2014] Dias, J., Mascarenhas, S., and Paiva, A. (2014). FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework. In *Emotion Modeling*, volume LNCS 8750, pages 44–56. Springer International Publishing.

[Dontnod Entertainment, 2015] Dontnod Entertainment (2015). Life is Strange. PC/Console Game. URL: http://www.lifeisstrange.com [22.3.2017].

[Ekman and Friesen, 1971] Ekman, P. and Friesen, W. V. (1971). Constants across cultures in the face and emotion. *Journal of Personality and Social Psychology*, 17(2):124–129.

[El-Nasr et al., 2000] El-Nasr, M. S., Yen, J., and Ioerger, T. R. (2000). FLAME—Fuzzy Logic Adaptive Model of Emotions. *Autonomous Agents and Multi-Agent Systems*, 3(3):219–257.

[Electronic Arts, 2014] Electronic Arts (2014). The Sims 4. PC Game. URL: http://www.thesims.com/ [22.3.2017].

[Endrass et al., 2011] Endrass, B., Rehm, M., and André, E. (2011). Planning Small Talk behavior with cultural influences for multiagent systems. *Computer Speech & Language*, 25(2):158–174.

[Entertainment Software Association, 2016] Entertainment Software Association (2016). 2015 Annual Report. Annual Report Available Online. URL: http://www.theesa.com [22.3.2017].

[Epic Games, 2002] Epic Games (2002). Unreal Engine 2. Game Engine. URL: https://docs.unrealengine.com/udk/Two/UnrealEngine2Runtime22262002.html [22.3.2017].

[Epic Games, 2004] Epic Games (2004). Unreal Tournament 2004. First Person Shooter Game. URL: https://www.epicgames.com/ [22.3.2017].

[Epic Games, 2012] Epic Games (2012). Unreal Engine 4. Game Engine. URL: http://www.unrealengine.com [22.3.2017].

[Fairclough, 2004] Fairclough, C. (2004). *Story Games and the OPIATE System*. Phd thesis, University of Dublin, Trinity College.

[Ferrucci et al., 2010] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J., Nyberg, E., Prager, J. M., Schlaefer, N., and Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79.

[Fónagy, 2001] Fónagy, I. (2001). *Languages within language: An evolutive approach*, volume 13. John Benjamins Publishing.

[Forbus et al., 1995] Forbus, K. D., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205.

[Freytag and MacEwan, 1968] Freytag, G. and MacEwan, E. J. (1968). *Freytag's Technique of the Drama; an Exposition of Dramatic Composition and Art. an Authorized Translation from the 6th German ed.* New York And London, Benjamin Blom, New York, NY, U.S.A., 6 edition.

[Friedemann et al., 2014] Friedemann, S., Meier, K., and Jantke, K. P. (2014). Here's Looking at You, Player - The Potential of Eye Tracking Analysis for Player-centered Learning Game Design. In *Proceedings of the 6th International Conference on Computer Supported Education*, volume 1, pages 532–538. SCITEPRESS - Science and and Technology Publications.

[Friedman et al., 2001] Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

[Gebhard, 2005] Gebhard, P. (2005). ALMA – A Layered Model of Affect. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, pages 29–36, New York, New York, USA. ACM Press.

[Gemrot et al., 2013] Gemrot, J., Havlí, J., Bída, M., Kadlec, R., and Brom, C. (2013). yaPOSH Action Selection. *Intelligent Virtual Agents*, LNAI 8108:458–459.

[Gemrot et al., 2009] Gemrot, J., Kadlec, R., Bída, M., and Burkert, O. (2009). Pogamut 3 can assist developers in building AI (Not only) for their videogame agents. *Agents for Games and Simulations*, 5920:1–15.

[Gervás, 2013] Gervás, P. (2013). Propp's Morphology of the Folk Tale as a Grammar for Generation. *Proceedings of the 4th Workshop on Computational Models of Narrative (CMN'13)*, 32:106–122.

[Guy et al., 2010] Guy, S. J., Lin, M. C., and Manocha, D. (2010). Modeling Collision Avoidance Behavior for Virtual Humans. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 2:575–582.

[Halliwell, 1987] Halliwell, S. (1987). *The Poetics of Aristotle: Translation and Commentary*. The University of North Carolina Press, 1 edition.

[Harel, 1987] Harel, D. (1987). Statecharts: a visual complex systems. *Science of Computer Programming*, 8:231–274.

[Harrigan and Wardrip-Fruin, 2007] Harrigan, P. and Wardrip-Fruin, N. (2007). *Second Person: Role-Playing and Story in Games and Playable Media*. MIT Press.

[Hawlitschek and Köppen, 2014] Hawlitschek, A. and Köppen, V. (2014). Analyzing Player Behavior in Digital Game-Based Learning : Advantages and Challenges. *Proceedings of the 8th European Conference on Games Based Learning*, pages 199–206.

[Hunter, 2015] Hunter, S. (2015). Google Self-Driving Car Project. Technical Report May.

[Ierusalimschy et al., 1993] Ierusalimschy, R., Celes, W., and de Figueiredo, L. H. (1993). Lua. OpenSource Scripting Language. URL: http://www.lua.org [22.3.2017].

[Jaccard, 1901] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579.

[Jan and Traum, 2007] Jan, D. and Traum, D. R. (2007). Dynamic movement and positioning of embodied agents in multiparty conversations. In *Proceedings of the Workshop on Embodied Language Processing*, pages 59–66, New York, New York, USA. ACM Press.

[Jaro, 1989] Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420.

[Kadlec et al., 2013] Kadlec, R., Čermák, M., Behan, Z., and Brom, C. (2013). Generating Corpora of Activities of Daily Living and towards Measuring the Corpora's Complexity. In Dignum, F., Brom, C., Hindriks, K., Beer, M., and Richards, D., editors, *Cognitive Agents for Virtual Environments*, volume LNCS 7764, pages 149–166. Springer Berlin Heidelbergr.

[Kahneman, 2013] Kahneman, D. (2013). *Thinking, fast and slow*. Farrar, Straus and Giroux, 1st edition.

[Karamouzas et al., 2009] Karamouzas, I., Heil, P., van Beek, P., and Overmars, M. H. (2009). A Predictive Collision Avoidance Model for Pedestrian Simulation. *Motion in Games*, LNCS 5884:41–52.

[Karamouzas and Overmars, 2010] Karamouzas, I. and Overmars, M. (2010). Simulating the local behaviour of small pedestrian groups. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology - VRST '10*, volume 1, pages 183–190, New York, New York, USA. ACM Press.

[Karimaghalou et al., 2014] Karimaghalou, N., Bernardet, U., and DiPaola, S. (2014). A model for social spatial behavior in virtual characters. *Computer Animation and Virtual Worlds*, 25(3-4):505–517.

[Kaufman and Rousseeuw, 2005] Kaufman, L. and Rousseeuw, P. J., editors (2005). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[Kirby, 1991] Kirby, J. (1991). Mimesis and Diegesis: Foundations of Aesthetic Theory in Plato and Aristotle. *Helios*, 18(2):113–128.

[Kleinginna and Kleinginna, 1981] Kleinginna, P. R. and Kleinginna, A. M. (1981). A categorized list of emotion definitions, with suggestions for a consensual definition. *Motivation and Emotion*, 5(4):345–379.

[Koelstra and Patras, 2013] Koelstra, S. and Patras, I. (2013). Fusion of facial expressions and EEG for implicit affective tagging. *Image and Vision Computing*, 31(2):164–174.

[Korstanje et al., 2016] Korstanje, R., Brom, C., Gemrot, J., and Hindriks, K. V. (2016). A Comparative Study of Programming Agents in POSH and GOAL. *Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, 2(Icaart):192–203.

[Krämer et al., 2014] Krämer, K., Bente, G., Kuzmanovic, B., Barisic, I., Pfeiffer, U. J., Georgescu, A. L., and Vogeley, K. (2014). Neural correlates of emotion perception depending on culture and gaze direction. *Culture and Brain*, 2(1):27–51.

[Laird, 2012] Laird, J. E. (2012). *The Soar cognitive architecture*. MIT Press.

[Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

[Lin et al., 2012] Lin, J., Spraragen, M., and Zyda, M. (2012). Computational models of emotion and cognition. *Advances in Cognitive Systems*, 2:59–76.

[Louchart, 2007] Louchart, S. (2007). *Emergent Narrative – towards a narrative theory of Virtual Reality*. Phd thesis, University of Salford.

[Louchart and Aylett, 2004] Louchart, S. and Aylett, R. (2004). Narrative theory and emergent interactive narrative. *International Journal of Continuing Engineering Education and Lifelong Learning*, 14(6):506–518.

[Loyall, 1997] Loyall, A. B. (1997). *Believable Agents: Building Interactive Personalities*. Phd thesis, Carnegie Mellon University.

[Marsella et al., 2010] Marsella, S., Gratch, J., and Petta, P. (2010). Computational models of emotion. *A Blueprint for Affective Computing-A sourcebook and manual*, 11(1):21–46.

[Marsella and Gratch, 2009] Marsella, S. C. and Gratch, J. (2009). EMA: A process model of appraisal dynamics. *Cognitive Systems Research*, 10(1):70–90.

[Mateas, 2002] Mateas, M. (2002). *Interactive drama, art and artificial intelligence*. Phd thesis, Carnegie Mellon University.

[Mateas and Stern, 2002] Mateas, M. and Stern, A. (2002). Towards integrating plot and character for interactive drama. *Socially Intelligent Agents*, 3:221–228.

[Mateas and Stern, 2003] Mateas, M. and Stern, A. (2003). Façade: An Experiment in Building a Fully-Realized Interactive Drama. In *Game Developers Conference, Game Design track*.

[Mateas and Stern, 2004] Mateas, M. and Stern, A. (2004). A Behavior Language: Joint Action and Behavioral Idioms. In Prendinge, H. and Ishizuka, M., editors, *Life-Like Characters*, pages 135–161. Springer Berlin Heidelberg.

[Mateas and Stern, 2005a] Mateas, M. and Stern, A. (2005a). Build It to Understand It: Ludology Meets Narratology in Game Design Space. In *Proceedings of the 2005 Digital Games Research Association Conference (DiGRA)*, Vancouver, Canada.

[Mateas and Stern, 2005b] Mateas, M. and Stern, A. (2005b). Structuring content in the Façade interactive drama architecture. In *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 93–98.

[McCoy et al., 2013] McCoy, J., Treanor, M., Samuel, B., and Reed, A. (2013). Prom Week: Designing past the game/story dilemma. *Proceedings of the 8th International Conference on Foundations of Digital Games: ACM*, pages 94–101.

[McCoy et al., 2014] McCoy, J., Treanor, M., Samuel, B., Reed, A. A., Mateas, M., and Wardrip-Fruin, N. (2014). Social story worlds with Comme il Faut. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(2):97–112.

[McCrae and John, 1992] McCrae, R. R. and John, O. P. (1992). An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175–215.

[McKee, 1997] McKee, R. (1997). *Story: Substance, structure, style and the principles of screen writing*. Regan Books, New York.

[Mehrabian, 1996] Mehrabian, A. (1996). Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament. *Current Psychology*, 14(4):261–292.

[Mehta et al., 2007] Mehta, M., Dow, S., Mateas, M., and MacIntyre, B. (2007). Evaluating a Conversation-Centered Interactive Drama. *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 5:24–31.

[Minsky, 2007] Minsky, M. (2007). *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind*. Simon & Schuster.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. a., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Mojang, 2011] Mojang (2011). Minecraft. PC and Console Sandbox Game. URL: http://minecraft.net/ [22.3.2017].

[Mononen, 2009] Mononen, M. (2009). Recast. NavMesh Generating Library. URL: https://github.com/recastnavigation/recastnavigation [22.3.2017].

[Moore, 1965] Moore, G. (1965). Cramming More Components Onto Integrated Circuits. *Electronics*, 38(8):82–85.

[Moravčík et al., 2017] Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, page online preprint.

[Muñoz-Avila et al., 2013] Muñoz-Avila, H., Bauckhage, C., Bida, M., Congdon, C. B., and Kendall, G. (2013). Learning and Game AI. In Lucas, S. M., Mateas, M., Preuss, M., Spronck, P., and Togelius, J., editors, *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 33–43. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

[Nelson and Mateas, 2005] Nelson, M. J. and Mateas, M. (2005). Search-Based Drama Management in the Interactive Fiction Anchorhead. *Proceedings of the First Annual Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 99–104.

[Object Refinery Limited, 2014] Object Refinery Limited (2014). JFreeChart. Free Plotting Software. URL: http://www.jfree.org/jfreechart/ [22.3.2017].

[Ontañón and Zhu, 2011] Ontañón, S. and Zhu, J. (2011). On the role of domain knowledge in analogy-based story generation. *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 1717–1722.

[Ortony et al., 1988] Ortony, A., Clore, G., and Collins, A. (1988). *The Cognitive Structure of Emotions*. Cambridge University Press.

[Park et al., 2013a] Park, J. H., Rojas, F. A., and Yang, H. S. (2013a). A collision avoidance behavior model for crowd simulation based on psychological findings. *Computer Animation and Virtual Worlds*, 24(3-4):173–183.

[Park et al., 2013b] Park, S. I., Quek, F., and Cao, Y. (2013b). Simulating and animating social dynamics: embedding small pedestrian groups in crowds. *Computer Animation and Virtual Worlds*, 24(3-4):155–164.

[Pedica and Vilhjálmsson, 2008] Pedica, C. and Vilhjálmsson, H. (2008). Social perception and steering for online avatars. *Intelligent Virtual Agents*, LNAI 5208:104–116.

[Pedica and Vilhjalmsson, 2010] Pedica, C. and Vilhjalmsson, H. H. (2010). Spontaneous Avatar Behavior for Human Territoriality. *Applied Artificial Intelligence*, 24(6):575–593.

[Pedica et al., 2010] Pedica, C., Vilhjálmsson, H. H., and Lárusdóttir, M. (2010). Avatars in Conversation: The Importance of Simulating Territorial Behavior. *Intelligent Virtual Agents*, LNCS 6356:336–342.

[Pérez y Pérez, 2007] Pérez y Pérez, R. (2007). Employing emotions to drive plot generation in a computer-based storyteller. *Cognitive Systems Research*, 8(2):89–109.

[Peters and Ennis, 2009] Peters, C. and Ennis, C. (2009). Modeling groups of plausible virtual pedestrians. *IEEE Computer Graphics and Applications*, 29(4):54–63.

[Píbil et al., 2012] Píbil, R., Novák, P., Brom, C., and Gemrot, J. (2012). Notes on pragmatic agent-programming with Jason. *Programming Multi-Agent Systems*, LNAI 7217:58–73.

[Plato et al., 1992] Plato, Grube, G. M. A., and Reeve, C. D. C. (1992). *Republic*. Hackett Publishing Company, 2 edition.

[Plutchik, 2003] Plutchik, R. (2003). *Emotions and Life: Perspectives from Psychology, Biology, and Evolution*. American Psychological Association.

[Popelová, 2011] Popelová, M. (2011). *Steering techniques library for virtual agents*. Bachelor thesis, Charles University.

[Popelová et al., 2011] Popelová, M., Bída, M., Brom, C., Gemrot, J., and Tomek, J. (2011). When a couple goes together: walk along steering. *Motion in Games*, LNCS 7060:278–289.

[Porteous et al., 2010] Porteous, J., Cavazza, M., and Charles, F. (2010). Applying planning to interactive storytelling. *ACM Transactions on Intelligent Systems and Technology*, 1(2):1–21.

[Porteous et al., 2013] Porteous, J., Charles, F., and Cavazza, M. (2013). NetworkING: using character relationships for interactive narrative generation. *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 595–602.

[Posner et al., 2005] Posner, J., Russell, J. A., and Peterson, B. S. (2005). The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(3):715–734.

[Propp, 2010] Propp, V. (2010). *Morphology of the Folktale*, volume 9. University of Texas Press.

[Quantic Dream, 2010] Quantic Dream (2010). Heavy Rain. Video Game. URL: http://www.quanticdream.com/ [22.3.2017].

[Rabe and Wachsmuth, 2013] Rabe, F. and Wachsmuth, I. (2013). An Event Metric and an Episode Metric for a Virtual Guide. *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, 2:543–546.

[Rabin, 2013] Rabin, S. (2013). *Game AI Pro: Collected Wisdom of Game AI Professionals*. A K Peters/CRC Press, 1 edition.

[Rajagopalan et al., 1999] Rajagopalan, A. N., Jain, A., and Desai, U. B. (1999). Data clustering using hierarchical deterministic annealing and higher order statistics. *IEEE Transactions on Circtuits and Sysytems-II: Analog and Digital Signal Processing*, 46(8):1100–1104.

[Rank et al., 2014] Rank, S., Hoffmann, S., Struck, H.-g., Spierling, U., Mayr, S., and Petta, P. (2014). Creativity in Configuring Affective Agents for Interactive Storytelling. *Applied Artificial Intelligence: An International Journal*, 28(6):629–645.

[Rao, 1996] Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, LNCS 1038:42–55.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI Agents: From Theory to Practice. *International Conference on Multiagent Systems*, 95:312–319.

[Ravenet et al., 2016] Ravenet, B., Pecune, F., Chollet, M., and Pelachaud, C. (2016). Emotion and Attitude Modeling for Non-player Characters. In Karpouzis, K. and Yannakakis, G. N., editors, *Emotion in Games" Theory and Praxis*, pages 139–154. Springer International Publishing.

[Reisenzein et al., 2013] Reisenzein, R., Hudlicka, E., Dastani, M., Gratch, J., Hindriks, K., Lorini, E., and Meyer, J.-j. C. (2013). Computational Modeling of Emotion: Toward Improving the Inter- and Intradisciplinary Exchange. *IEEE Transactions on Affective Computing*, 4(3):246–266.

[Reynolds, 1987] Reynolds, C. W. (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34.

[Reynolds, 1999] Reynolds, C. W. (1999). Steering behaviors for autonomous characters. *Game Developers Conference*, pages 763–782.

[Ricks and Egbert, 2012] Ricks, B. C. and Egbert, P. K. (2012). More realistic, flexible, and expressive social crowds using transactional analysis. *The Visual Computer*, 28(6-8):889–898.

[Riedl et al., 2011] Riedl, M., Thue, D., and Bulitko, V. (2011). Game AI as Storytelling. In *Artificial Intelligence for Computer Games*, pages 125–150. Springer New York, New York, NY.

[Riedl and Stern, 2006] Riedl, M. O. and Stern, A. (2006). Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. *Technologies for Interactive Digital Storytelling and Entertainment*, LNCS 4326:1–12.

[Rishes et al., 2013] Rishes, E., Lukin, S. M., Elson, D. K., and Walker, M. A. (2013). Generating different story tellings from semantic representations of narrative. *Interactive Storytelling*, LNCS 8230:192–204.

[Rittel and Webber, 1973] Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4(2):155–169.

[Rockstar Games, 2010] Rockstar Games (2010). Red Dead Redemption. Open World Console Game. URL: http://www.rockstargames.com/reddeadredemption/ [22.3.2017].

[Roemmele and Gordon, 2015] Roemmele, M. and Gordon, A. S. (2015). Creative help: A story writing assistant. *Interactive Storytelling*, LNCS 9445:81–92.

[Rojas et al., 2013] Rojas, F. A., Park, J. H., and Yang, H. S. (2013). Group Agent-based Steering for the Realistic Corner Turning and Group Movement of Pedestrians in a Crowd Simulation. *Computer Animation and Social Agents (CASA)*.

[Roth and Vermeulen, 2013] Roth, C. and Vermeulen, I. (2013). Breaching Interactive Storytelling's Implicit Agreement: A Content Analysis of Façade User Behaviors. *Interactive Storytelling*, LNCS 8230:168–173.

[Russell and Norvig, 2010] Russell, S. J. and Norvig, P. (2010). *Artificial Intelligence: A Modern approach*. Prentice Hall, Upper Saddle River, New Jersey, 3 edition.

[Sali and Mateas, 2011] Sali, S. and Mateas, M. (2011). Using information visualization to understand interactive narrative: A case study on Façade. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, LNCS 7069:284–289.

[Sarlej, 2014] Sarlej, M. (2014). *A Lesson Learned: Using Emotions to Generate Stories with Morals*. Phd thesis, UNSW Australia.

[Sarlej and Ryan, 2013] Sarlej, M. and Ryan, M. (2013). Generating Stories with Morals. *Interactive Storytelling*, LNCS 8230:217–222.

[Scherer, 2001] Scherer, K. (2001). Appraisal considered as a process of multilevel sequential checking. *Appraisal processes in emotion: Theory, methods, research*, pages 92–120.

[Schoenau-Fog, 2011] Schoenau-Fog, H. (2011). Hooked!–evaluating engagement as continuation desire in interactive narratives. *Interactive Storytelling*, LNCS 7069:219–230.

[Seif El-Nasr et al., 2013] Seif El-Nasr, M., Milam, D., and Maygoli, T. (2013). Experiencing interactive narrative: A qualitative analysis of Facade. *Entertainment Computing*, 4(1):39–52.

[Si et al., 2010] Si, M., Marsella, S., and Pynadath, D. (2010). Importance of well-motivated characters in interactive narratives: An empirical evaluation. *Interactive Storytelling*, LNCS 6432:16–25.

[Si and Marsella, 2010] Si, M. and Marsella, S. C. (2010). Modeling Rich Characters in Interactive Narrative Games. In *GAMEON-ASIA, Shanghai, China*, pages 12–20.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[Smith and Lazarus, 1990] Smith, C. A. and Lazarus, R. S. (1990). Emotion and Adaptation. In Pervin, L. A., editor, *Handbook of Personality: Theory and Research*, chapter 23, pages 609–637. Guilford Press.

[Spierling et al., 2002] Spierling, U., Grasbon, D., Braun, N., and Iurgel, I. (2002). Setting the scene: Playing digital director in interactive storytelling and creation. *Computers and Graphics (Pergamon)*, 26(1):31–44.

[Spierling et al., 2010] Spierling, U., Hoffmann, S., and Szilas, N. (2010). Report on pre-scriptive narrative formalisms and creation methods in interactive storytelling (non-digital and digital). Technical Report D3.1, IRIS Network of Excellence, FP7-ICT-231824.

[Square Enix, 2016] Square Enix (1987-2016). Final Fantasy Series. PC/Console Game Series. URL: http://www.thefinalfantasy.com [22.3.2017].

[Stern, 2008] Stern, A. (2008). Embracing the combinatorial explosion: A brief prescription for interactive story R&D. *Interactive Storytelling*, LNCS 5334:1–5.

[Steunebrink et al., 2009] Steunebrink, B. R., Dastani, M., and Meyer, J.-J. C. (2009). The OCC Model Revisited. In *4th Workshop on Emotion and Computing*, pages 1–8.

[Stewart and West, 2007] Stewart, T. C. and West, R. L. (2007). Deconstructing and reconstructing ACT-R: Exploring the architectural space. *Cognitive Systems Research*, 8(3):227–236.

[Swartjes, 2010] Swartjes, I. (2010). *Whose story is it anyway? : how improv informs agency and authorship of emergent narrative.* Phd thesis, University of Twente.

[Szilas, 2003] Szilas, N. (2003). IDtension: a narrative engine for Interactive Drama. In Göbel, S., Braun, N., Spierling, U., Dechau, J., and Diener, H., editors, *Proceedings of Technologies for Interactive Digital Storytelling and Entertainment (TIDSE 2003)*, pages 187–203, Stuttgart. Fraunhofer IRB Verlag.

[Szilas and Ilea, 2014] Szilas, N. and Ilea, I. (2014). Objective Metrics for Interactive Narrative. *Interactive Storytelling*, LNCS 8832:91–102.

[Tanenbaum and Tanenbaum, 2010] Tanenbaum, K. and Tanenbaum, J. (2010). Agency as commitment to meaning: communicative competence in games. *Digital Creativity*, 21(1):11–17.

[Telltale Games, 2012] Telltale Games (2012). The Walking Dead. PC and Console Game. URL: http://telltale.com/series/the-walking-dead/ [22.3.2017].

[Tomaszewski and Binsted, 2007] Tomaszewski, Z. and Binsted, K. (2007). The Limitations of a Propp-based Approach to Interactive Drama. In *Intelligent Narrative Technologies, Papers from the 2007 AAAI Fall Symposium*, volume 1, pages 167–173.

[Traum et al., 2015] Traum, D., Jones, A., Hays, K., Maio, H., Alexander, O., Artstein, R., Debevec, P., Gainer, A., Georgila, K., Haase, K., Jungblut, K., Leuski, A., Smith, S., and Swartout, W. (2015). New Dimensions in Testimony: Digitally Preserving a Holocaust Survivor's Interactive Storytelling. *Interactive Storytelling*, LNCS 9445:269–281.

[Unity Technologies, 2005] Unity Technologies (2005). Unity. Game Engine. URL: http://unity3d.com [22.3.2017].

[Velásquez, 1998] Velásquez, J. D. (1998). When robots weep: emotional memories and decision-making. *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 70–75.

[Vermeulen et al., 2010] Vermeulen, I. E., Roth, C., Vorderer, P., and Klimmt, C. (2010). Measuring user responses to interactive stories: Towards a standardized assessment tool. *Interactive Storytelling*, LNCS 6432:38–43.

[Veselovská and Tamchyna, 2014] Veselovská, K. and Tamchyna, A. (2014). UFAL: Using Hand-crafted Rules in Aspect Based Sentiment Analysis on Parsed Data. *SemEval 2014*, pages 694–698.

[Ware et al., 2012] Ware, S., Young, R., Harrison, B., and Roberts, D. (2012). Four quantitative metrics describing narrative conflict. *Interactive Storytelling*, LNCS 7648:18–29.

[Wendel et al., 2014] Wendel, V., Bär, M.-a., Hahn, R., Jahn, B., Mehltretter, M., Gobel, S., and Steinmetz, R. (2014). Automatic situation recognition in collaborative multiplayer Serious Games. *Conference on Game-based Learning*, 2:610–619.

[Weyhrauch and Bates, 1997] Weyhrauch, P. and Bates, J. (1997). *Guiding interactive drama*. Phd. thesis, CMU.

[Winkler, 1990] Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Me- thods (American Statistical Association)*, pages 354–359.

[Wooldridge, 2000] Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press.

[y Pérez and Sharples, 2001] y Pérez, P. and Sharples, M. (2001). MEXICA: A computer model of a cognitive account of creative writing. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):119–139.

[Yannakakis and Hallam, 2005] Yannakakis, G. and Hallam, J. (2005). A generic approach for generating interesting interactive pac-man opponents. *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, pages 94–101.

[Yannakakis and Hallam, 2007] Yannakakis, G. N. and Hallam, J. (2007). Capturing player enjoyment in computer games. *Studies in Computational Intelligence*, 71:175–201.

[Yannakakis and Togelius, 2015] Yannakakis, G. N. and Togelius, J. (2015). A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):317–335.

[Ziegfeld, 1989] Ziegfeld, R. (1989). Interactive Fiction: A New Literary Genre? *New Literary History*, 20(2):341–372.

[Zwaan et al., 1995] Zwaan, R. a., Langston, M. C., and Graesser, a. C. (1995). The Construction of Situation Models in Narrative Comprehension: An Event-Indexing Model. *Psychological Science*, 6(5):292–297.

List of Figures

1.1 Freytag's Pyramid . . . 12
1.2 Beads on a String . . . 13
1.3 Firewatch 3D Game Environment. . . . 25
1.4 Trip and Grace greeting the player in Façade. . . . 27
1.5 FearNOT! Game Environment. . . . 29
1.6 FearNOT! Game Environment – Advice Window. . . . 29
1.7 The Prom Week game. . . . 30

2.1 SimDate3D Level Two Characters. . . . 37
2.2 The EWA Architecture Overview. . . . 43
2.3 Basic decision making flow of the EWA architecture. . . . 45
2.4 Simplest Behavior Example. . . . 45
2.5 Hierarchical Goal Example. . . . 46
2.6 EWA Decision Making System Overview. . . . 50
2.7 Structural part of the OCC theory . . . 53
2.8 EWA Emotion Model Overview. . . . 57
2.9 General Overview of Steerings Architecture . . . 61

3.1 Pogamut 3 Architecture Overview. . . . 64
3.2 StoryFactory 3D Environment. . . . 65
3.3 StoryFactory GUI. . . . 66
3.4 SimDate3D Level One Overview. . . . 68
3.5 SimDate3D Emoticons. . . . 71
3.6 SimDate3D Level One – Simple FSM Example. . . . 75
3.7 SimDate3D Level Two Overview. . . . 78
3.8 SimDate3D birds-eye map overview. . . . 79
3.9 SimDate3D Level Two – FSM Example. . . . 82
3.10 Triangular Steering Third Condition Visualization. . . . 82
3.11 Four Results of Triangular Steering Conditions. . . . 83
3.12 EWA Debugging Tool Overview. . . . 87
3.13 Log Visualization – Agent Goals. . . . 89
3.14 Log Visualization – Agent Positions. . . . 89
3.15 Log Visualization – Agent Focus. . . . 89

4.1 Screenshots from SimDate3D . . . 112
4.2 Experiment 0 – SimDate3D Level One clustering results. . . . 116
4.3 Experiment 0 – SimDate3D Level One Tension Values. . . . 117
4.4 Experiment 0 – SimDate3D Level Two clustering results. . . . 119
4.5 Experiment 0 – SimDate3D Level Two Time Deviations. . . . 119
4.6 Experiment 1 – SimDate3D Level Two clustering results. . . . 121
4.7 Experiment 1 – SimDate3D Level Two Tension Values . . . 122
4.8 Experiment 2 – SimDate3D Level Two clustering results. . . . 123
4.9 Experiment 3 – MOSS domain clustering results. . . . 125

B.1 Debugger – General Agent Overview. . . . 155
B.2 Debugger – Agent Affect Overview. . . . 156
B.3 Debugger – Agent Settings Window. . . . 156

B.4 Log Visualization – Feeling Plot. 157
B.5 Log Visualization – Mood Plot. 157
B.6 Log Visualization – Tension Plot. 157

List of Tables

2.1 Feeling emotions. 57

4.1 Experiment 0 – SD One stories action strings examples. 115
4.2 Experiment 0 – SD Two stories action strings examples. 118
4.3 Experiment 2 – SD Two stories sub-scene strings examples. 124

A.1 ALMA moods. 153
A.2 ALMA emotions. 154

C.1 SD One key controls. 158

D.1 SD Two general key controls. 159
D.2 SD Two action key controls. 159
D.3 SD Two proposal key controls. 160
D.4 SD Two walking key controls. 160

Acronyms

ABL A Behavior Language. 23, 24, 41, 62, 85

AI Artificial Intelligence. 4, 5, 9, 17, 19, 21, 22, 24, 33, 35, 36, 39, 41, 58, 59, 63, 74, 92, 130

BDI Belief–desire–intention software model. 36, 38, 39, 42, 44, 62

ECAs Embodied Conversational Agents. 58

EEC Emotion Eliciting Condition. 55, 56, 58, 87, 91

EWA Emohawk Agent Architecture. 7, 8, 18, 21, 22, 30, 35, 36, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 58, 59, 60, 61, 63, 67, 68, 69, 70, 74, 75, 76, 77, 81, 83, 84, 85, 86, 87, 88, 90, 91, 92, 101, 107, 129, 147, 163

FSMs Finite State Machines. 39, 75, 81, 91

IDS Interactive Digital Storytelling. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 31, 33, 34, 35, 38, 40, 41, 43, 51, 61, 64, 65, 77, 84, 86, 87, 88, 90, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 126, 128, 129, 130

IVA Intelligent Virtual Agent. 4, 7, 9, 16, 17, 18, 19, 21, 30, 35, 36, 37, 38, 42, 43, 44, 45, 46, 48, 52, 54, 58, 59, 60, 63, 66, 81, 90, 91, 92, 107, 129, 130

MBTs Modular Behavior Trees. 22

MOSS Moral Storytelling System. 8, 107, 108, 109, 111, 112, 113, 114, 124, 125, 126, 129, 147

NLG Natural Language Generation. 32

NLU Natural Language Understanding. 18, 27, 70

NPC Non-player Character. 4, 6, 7, 14, 16, 19, 21, 22, 23, 36, 39, 40, 41, 58, 59, 60, 61, 62, 129

OCC Ortony, Clore and Collins Emotion Theory. 20, 21, 22, 53, 54, 55, 56, 57, 58, 91, 97, 108

RPG Role Playing Game. 21

SD Two SimDate3D Level Two. 7, 8, 36, 37, 43, 61, 62, 63, 64, 67, 72, 75, 77, 81, 83, 84, 85, 86, 90, 91, 92, 98, 101, 102, 103, 107, 108, 109, 110, 111, 112, 113, 114, 115, 118, 120, 123, 127, 129, 149, 159, 163

SD One SimDate3D Level One. 7, 8, 36, 43, 61, 62, 63, 64, 67, 69, 70, 72, 74, 75, 76, 77, 80, 81, 83, 84, 86, 90, 91, 92, 98, 103, 107, 108, 109, 111, 112, 114, 115, 118, 120, 129, 149, 158, 163

UE2 Unreal Engine 2. 7, 63, 65, 70, 71, 74, 75, 129, 163

Appendices

A. ALMA Emotions and Moods

See the overview of emotions and moods defined by ALMA in the tables below.

Exuberant	P+, A+, D+	Bored	P−, A−, D−
Dependent	P+, A+, D−	Disdainful	P−, A−, D+
Relaxed	P+, A−, D+	Anxious	P−, A+, D−
Docile	P+, A−, D−	Hostile	P−, A+, D+

Table A.1: ALMA moods. Table is showing eight possible moods states in ALMA along with their quadrant depiction defined by three dimensions pleasure (P), arousal (A) and dominance (D). Table based on a table from [Gebhard, 2005].

Emotion	**Mood Octant**
Admiration	P+ A+ D− Dependent
Anger	P− A+ D+ Hostile
Disliking	P− A+ D+ Hostile
Disappointment	P− A+ D− Anxious
Distress	P− A− D− Bored
Fear	P− A+ D− Anxious
FearsConfirmed	P− A− D− Bored
Gloating	P+ A− D− Docile
Gratification	P+ A+ D+ Exuberant
Gratitude	P+ A+ D− Dependent
HappyFor	P+ A+ D+ Exuberant
Hate	P− A+ D+ Hostile
Hope	P+ A+ D− Dependent
Joy	P+ A+ D+ Exuberant
Liking	P+ A+ D− Dependent
Love	P+ A+ D+ Exuberant
Pity	P− A− D− Bored
Pride	P+ A+ D+ Exuberant
Relief	P+ A− D+ Relaxed
Remorse	P− A+ D− Anxious
Reproach	P− A− D+ Disdainful
Resentment	P− A− D− Bored
Satisfaction	P+ A− D+ Relaxed
Shame	P− A+ D− Anxious

Table A.2: ALMA emotions. Here is a list of all emotions specified by ALMA and their corresponding mood quadrant. P stands for pleasure, A stands for arousal, D stands for dominance. Table based on a table taken from [Gebhard, 2005].

B. Debugging Tool Windows Overview

The Chapter describes the three debugging windows the designer can use to overview the behavior of an agent: General Agent Overview Window (Fig. B.1), Agent Affect Overview Window (Fig. B.1) and Agent Settings Window (Fig. B.3.

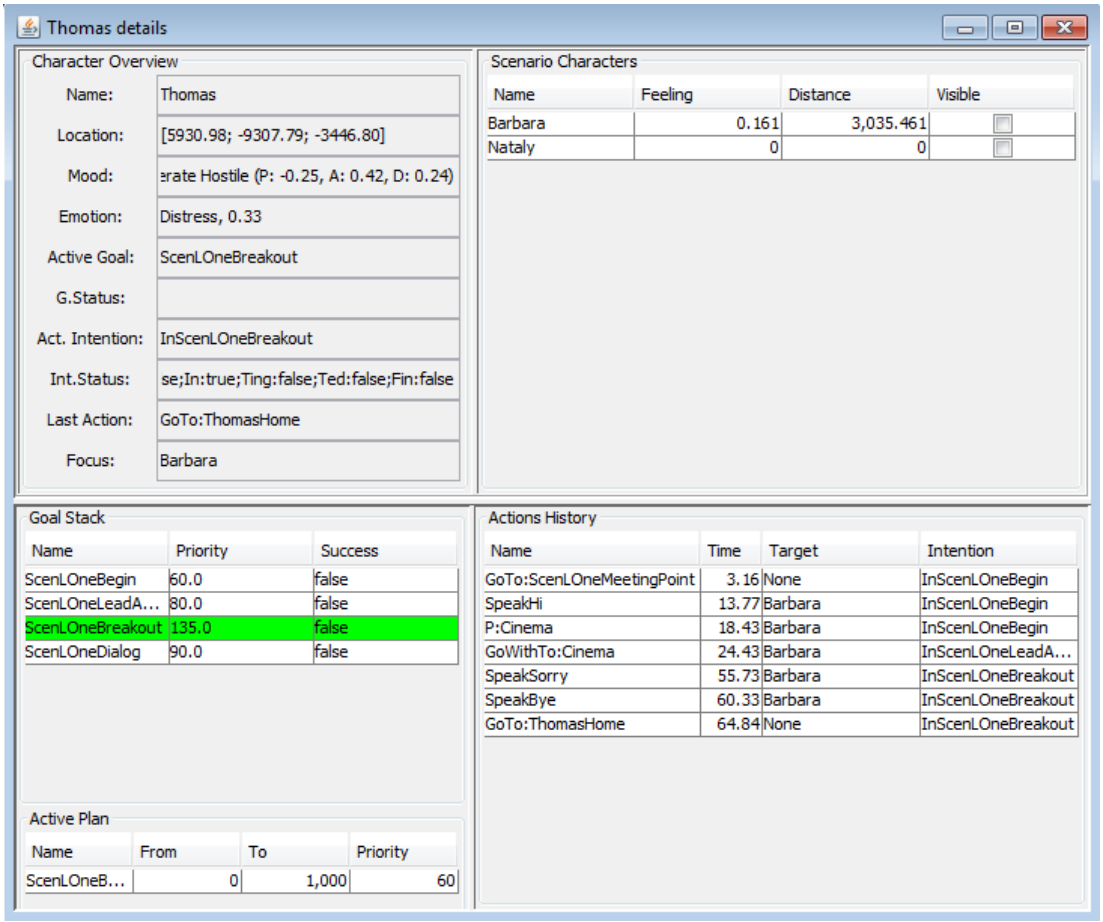


Figure B.1: Debugger – General Agent Overview. Main debugger agent overview window is showing the basic info about the agent in upper left corner along with the status of current intention. List of all active goals in the goal stack is shown in lower left corner along with currently active goal. List of all actions that were performed by the agent is shown in lower right corner. Upper right corner is showing all the other characters in the scenario with associated feeling values.

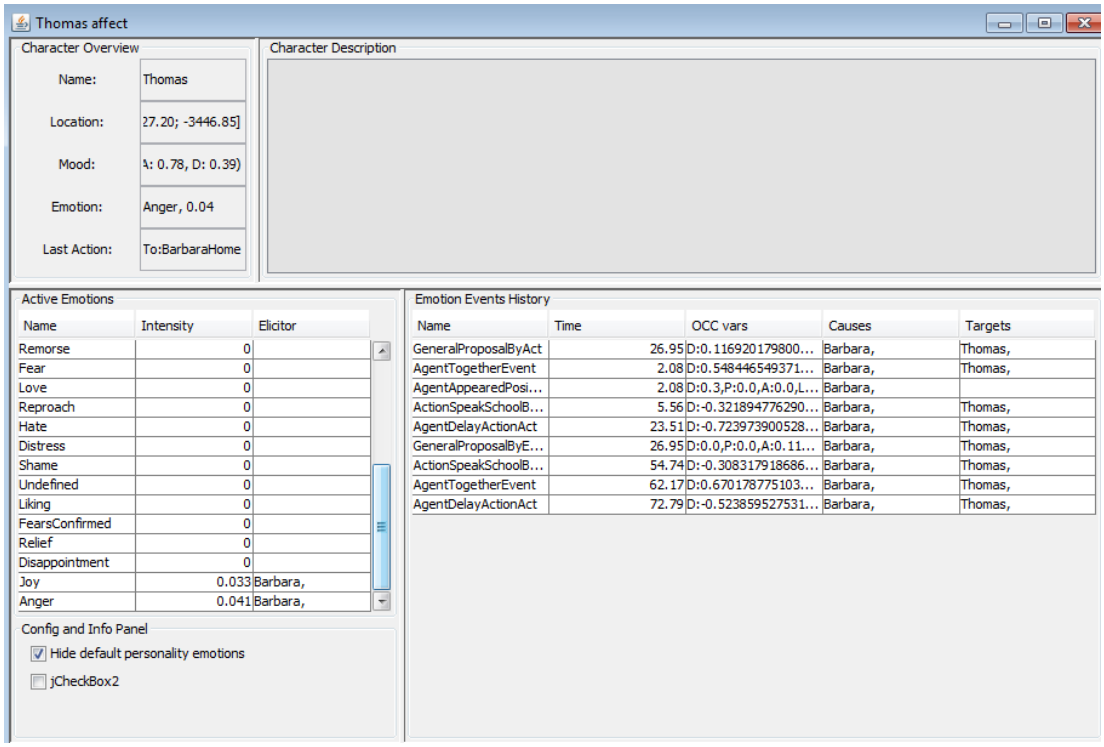


Figure B.2: Debugger – Agent Affect Overview. Window showing the affect overview of the agent. Basic information with the mood value is shown in upper left corner, list of all active emotions and their values is shown in lower left corner. Lower right corner is showing all emotion events detected by EEC factories with their OCC variable values.

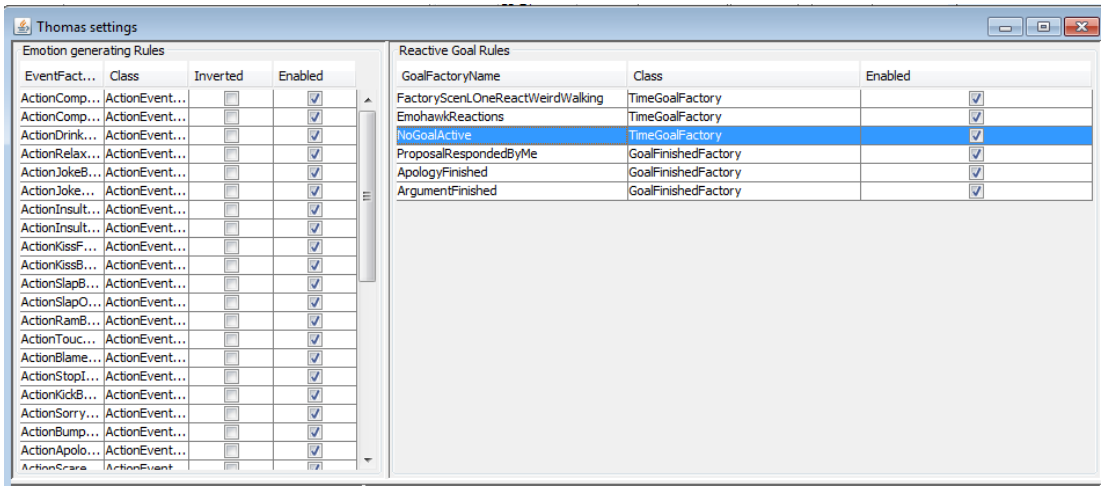


Figure B.3: Debugger – Agent Settings Window. Agent settings window is allowing to dynamically turn on or off reactive factories of the agent (pane on the right). Also EEC factories can be controlled by the UI (pane on the left).

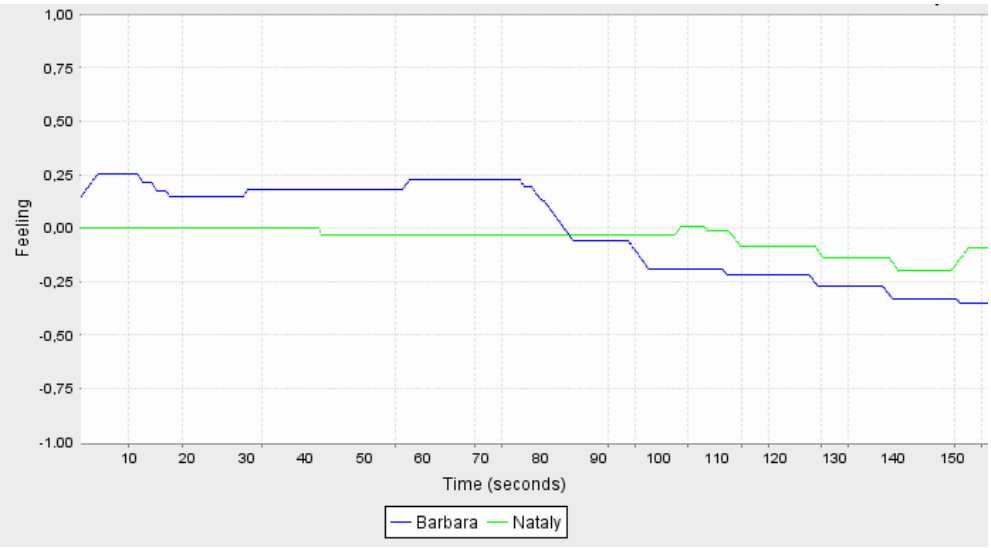


Figure B.4: Log Visualization – Feeling Plot. Plot visualizes feeling value changes of Thomas towards Barbara (blue) and Nataly (green) in the scenario.

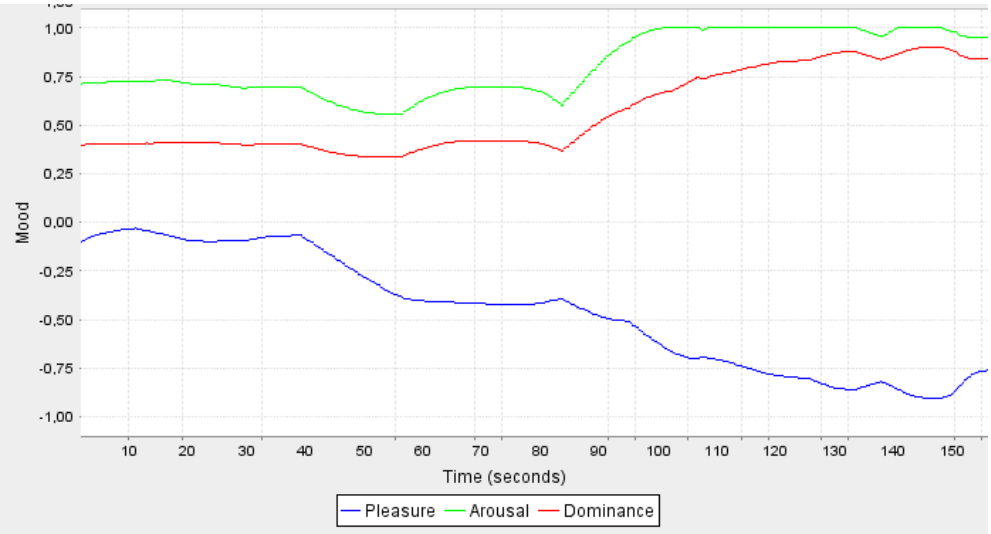


Figure B.5: Log Visualization – Mood Plot. Plot visualizes the changes in ALMA mood dimensions – pleasure (blue), arousal (green) and dominance (red) of Thomas during the scenario run.

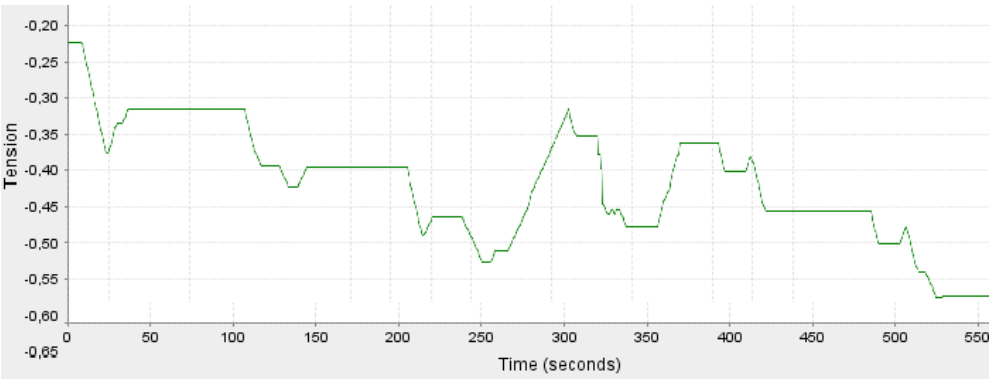


Figure B.6: Log Visualization – Tension Plot. Plot visualizes the tension value change in the scenario run.

C. SimDate3D Level One Game Controls

See the overview of key controls in SD One game in the table below.

Key	**Action**	**Cost**
1	triggers random positive action	250
2	triggers random negative action	250
3	weird walking triggered	250
4	normal walking triggered	250
Z	action joke triggered	350
X	action compliment triggered	350
V	action insult triggered	350
B	action blame triggered	350
N	random neutral action speak triggered	300
9 and 0	increase/decrease distance between characters	50
H, U, J, K	change the angle the girl is following the boy at	100
5	starts the game when the game is not running	N/A
6	ends the game	N/A

Table C.1: SD One key controls. List of the key controls in SD One.

D. SimDate3D Level Two Game Controls

See the overview of key controls in SD Two game in the tables below.

Key	Action
=	Cycle game speed – 1, 2 or 4. Recommended speed is 2. Setting game speed to 4 may cause the characters to "twitch" – the engine stops to handle the game well at this speed. This depends on the HW of the computer. The speed 4 intended use is to skip longer sequences when the character(s) are only going somewhere.
Q	Show/Hide in-game Help
5	Starts the ext game session if not running (characters appear in the game again in their initial positions and emotions)
6	Ends the game if not running. Use this key only when something goes wrong or to skip the argument at the end of the story.
M	Cycles through map modes. Those are: off, small map, large map. On the map, the player sees positions of the places and characters in the city.
P	Plays next audio track.
F9	makes a screenshot

Table D.1: SD Two general key controls. List of the general key controls in SD Two. For numeric keys do not used numpad.

Key	Action
Z	Tells a joke
X	Tells a compliment
B	Boy is being bossy to the girl
C	Boy tries to impress the girl
V	Boy insults a girl
N	Tells random neutral action

Table D.2: SD Two action key controls. List of the general key controls in SD Two.

Key	Proposal
1	Propose a kiss – girl has to agree (successful kiss raises score). Works only at certain places.
2	Propose "cuddling" action – girl has to agree (successful cuddling action raises score). Works only at certain places.
F1	Propose to go to the central park
F2	Propose to go to the jogging park
F3	Propose to go to the restaurant
F4	Propose to go to the cinema
F5	Propose to go to Barbara home (blonde girl)
F6	Propose to go to Nataly home (dark hair girl)
F7	Tells goodbye to current girl (the girl and Thomas will head home). Usefull when you want to start dating the other girl.

Table D.3: SD Two proposals key controls. List of the proposal key controls in SD Two. For numeric keys do not use numpad.

Key	Action
3	Makes the character to walk weird
4	Makes the character to walk normal
9 and 0	increase/decrease distance between characters
H, U, J, K	change the angle the boy is following the girl

Table D.4: SD Two walking key controls. List of the walking key controls in SD Two. For numeric keys do not use numpad.

E. Example Stories

This Chapter presents several examples of stories from SimDate3D and MOSS domains and show examples of the clustering of stories using the tension curve metric. In both cases we provide simple handcrafted natural language representations of the actions in the story.

E.1 SimDate3D Domain

Story 1: Thomas went with Barbara to the cinema. After the movie, he was rude to her. They have parted ways. Thomas went to Nataly's home to pick her up. They went out for a walk, but they did not speak much. Thomas insulted Nataly. They met Barbara. An argument started and both girls left Thomas.

Story 2: Thomas went with Barbara to the cinema. After the movie, he was rude to her. They have parted ways. Thomas went to Nataly's home to pick her up. Thomas was rude to Nataly. They went out for a walk and Thomas was rude to Nataly. They met Barbara. An argument started and both girls left Thomas.

Story 3: Thomas spent a long time with Barbara in the cinema, then he was very rude on her. Nataly was in the restaurant alone. Then Thomas and Barbara got very angry on each other, but continued talking. Nataly noticed them on their way from restaurant and she run towards them. An argument started and Thomas ended up with Nataly.

Stories 1, 2 and 3 get clustered together in most cases. Stories 1 and 2 are extremely similar and end the same, while story 3 is an example of a story that is relatively similar to the other two, but does not end the same.

Story 4: Thomas and Barbara were on a way to cinema. Thomas asked Barbara to kiss him and to cuddle, she refused. Then they've run into Nataly, argument started and Thomas ended up with Barbara.

Story 5: Thomas and Barbara were going to the cinema. Thomas was making jokes on the way. Before they've get to the cinema they've run into Nataly, argument started and Thomas ended up with Barbara.

Stories 4 and 5 on the other hand are also very similar and end the same but were almost never clustered together.

E.2 MOSS Domain

Story 1: A wizard gets hungry. He picks up a rose. A troll kidnaps a princess. The troll also kidnaps a dwarf. A knight rescues the princess from the troll. (Generated as an example for recklessness)

Story 2: A wizard gets hungry. He picks up a rose. A troll kidnaps a princess. The troll also kidnaps a dwarf. A dragon gives a treasure to the dwarf. (Generated as an example for recklessness)

Story 3: A dwarf kills a princess. A troll kidnaps the dwarf. A dragon tries to kidnap a unicorn, but fails. Fairy gives magical dust to the dragon. Dragon gives the dust back to the fairy. (Generated as an example for retribution)

While it is clearly visible, how stories 1 and 2 are extremely similar, story 3 seems very different as it is from a different retribution domain.

F. Content of the DVD

The attached DVD contains:

- A text of this thesis in the pdf format.
- Documentation of EWA, SD One and SD Two in the form of an overview document and attached Javadoc.
- Documentation of the StoryFactory tool in the form of an overview document and attached Javadoc.
- Documentation of the GameBots project used to connect UE2 environment to the Pogamut that was used in SD One, SD Two and StoryFactory.
- Source codes of SD One, SD Two and EWA that are all part of the EmohawkScenario project.
- Source codes of the StoryFactory tool and source codes for the GameBots project.
- Installers for SD One, SD Two, StoryFactory and UDK game content for StoryFactory UDK version.
- Videos showing the game-play of SD One and SD Two, an overview of the StoryFactory tool, working implementation of the GameBots project on UE2 and several other videos from the virtual quarrel simulator that are relevant to this thesis.
- Experiments containing all the experimental data, the results and guidelines how to replicate these data.
- Screenshots showing an overview of all implemented systems.

The layout of the DVD is discussed in detail in the *readme.txt* file in the DVD root folder.

The attached documentation outlines how to build and install the projects. However, the easiest way to try SD One, SD Two and StoryFactory is to use the attached installers on the DVD. The author recommends to watch the attached game-play videos first. Another resources are available online. For the StoryFactory project refer to `www.storyfactory.cz` and for SD One and SD Two to `pogamut.cuni.cz/pogamut-games`.