To construct believable flow and nuanced dialogues in a BDI framework

An architecture for the development of interactive narratives

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science (MSc), 45 ECTS
March 6, 2015

Student:	David Isaacs Paternostro
Number:	3370879
University:	Utrecht University
Faculty:	Science
Department:	Information and Computing Sciences
Research Group:	Intelligent Systems
Master Programme:	Technical Artificial Intelligence
Track:	Intelligent Agents
Organization:	&ranj Serious Games
Supervisor &ranj Serious Games & third reviewer:	dr. ir. I. Swartjes
Supervisor Utrecht University & first reviewer:	prof. dr. J.J.C. Meyer
Second reviewer:	dr. M.M. Dastani
Period:	04/2014 - 02/2015

Universiteit Utrecht

Abstract

There are many games with narrative aspects in which the player will have to speak with non-playable characters (NPC). These games exist both in the commercial gaming industry and the serious gaming industry. Serious games are games that are created to train or educate. The dialogues that take place in those games are often well-written. Yet, the characters are seldom open for much interaction. The reason for this, is that the dialogues in games are most often fully scripted and interaction would quickly cause a combinatorial explosion of possible storylines.

We would like to increase interaction with the NPC's in serious games while still retaining some of the expressive control currently provided by scripting dialogues, because we believe it will offer new opportunities for training and education. Especially for games that educate on bad news reports, sales tactics, interrogating witnesses or suspects, and other similar situations in which eloquence and an understanding of the nuance of human natural language are key to success. Current approaches for increasing the interactivity of NPC's do not offer the artists enough expressive control over the agents for this kind of games to be used as business product.

We argue that it is essential for an interactive character to be believable and social and that those aspects should permeate any architecture designed for the realisation of such an agent. That is why we propose to combine a Belief, Desire, Intention framework (BDI), with scripting. BDI offers us generality and modularity. Scripting offers us expressive control, because we can author specific parts of a dialog by hand.

The agent's action-selection mechanism is a partial implementation of JADEX (a BDI reasoning engine) into ActionScript®3. The agent's action-expression mechanism consists of two tools, a tool called OZ Dialog Editor, in which scripts can be authored, and a tool called OZ Dialog Player, which can run the scripts. The agent selects plans based on its beliefs, desires, and intentions. The execution of a plan consists of at least one action. An action opens a script as the expression of that action.

The architecture we designed is a first step towards creating believable and nuanced flow in a BDI framework. In game studios, programmers program the NPC's and game designers write the dialogues. They can do this separately. We aim to sustain this task-segregation between programming agents and writing dialogues. This task-segregation bears restrictions that shape our architecture.

The architecture was tested with two game designers at &ranj serious games by paper-prototyping. The game designers performed an exercise in which they had to use certain script building blocks to write scripts. After this exercise we conducted an interview where we discussed the functionality of the building block and the architecture in general. The shape and function of the building block was adjusted based on the results of the interview. In general, the tests indicated that our method to creating interactive narratives could work. However, we argue that there is a large gap between theory and practice in AI research. Often issues will only be addressed if the method is actually put to work. Therefore, we suggest future researchers to build a game using our architecture.

Contents

1 Introduction **7**
- 1.1 Interactive characters . . . 7
- 1.2 Motivation . . . 7
- 1.3 Current practices in the field of serious games . . . 10
  - 1.3.1 Scripted dialogues and storylines . . . 10
  - 1.3.2 BDI . . . 13
- 1.4 Combine BDI and scripting . . . 14
  - 1.4.1 Why dialogue-based games? . . . 14
- 1.5 GOFAI/Interactionist & Expressive AI . . . 15
- 1.6 Socialness and believability: what affords interpretivity? . . . 16
- 1.7 Approach and validation . . . 16
- 1.8 Practical choices, limitations, and building blocks . . . 17
  - 1.8.1 Strict segregation between action-selection and action-expression . . . 17
  - 1.8.2 Building scripts . . . 18
  - 1.8.3 Combinatorial explosion of context-independent scripts . . . 18
  - 1.8.4 Connecting scripts . . . 18
- 1.9 A look back . . . 19

2 Background and related work **20**
- 2.1 Philosophy of AI . . . 20
  - 2.1.1 GOFAI vs Interactionist . . . 20
  - 2.1.2 Expressive AI . . . 21
- 2.2 Desired properties of virtual agent-behaviour . . . 22
  - 2.2.1 Believability . . . 22
  - 2.2.2 Socialness . . . 25
  - 2.2.3 How are socialness and believability related? . . . 28
  - 2.2.4 Turn-taking . . . 28
  - 2.2.5 Transitions in behaviour . . . 29
  - 2.2.6 Summary . . . 32
- 2.3 Methodologies . . . 32
  - 2.3.1 A Belief, Desire, Intention paradigm . . . 32
  - 2.3.2 Language processing and generation . . . 34
  - 2.3.3 Combining scripted dialogues and BDI . . . 35

3 Method **37**
- 3.1 Designing the initial architecture . . . 37
  - 3.1.1 Analysing existing games . . . 38
  - 3.1.2 Translating game concepts to an agent and scripts . . . 38
  - 3.1.3 Considering the task-division and process of game-development . . . 39
- 3.2 Evaluation of the initial design . . . 39

4 Design **40**
4.1 An overview . . . 40
4.2 Design-choices and restrictions . . . 40
4.3 Initial Design . . . 41
4.3.1 The naive agent-script model . . . 42
4.3.2 Script building block . . . 43
4.3.3 Possible state structures . . . 44
4.3.4 Context . . . 50
4.3.5 Transitions between scripts . . . 54
4.3.6 Final model . . . 56
4.4 Second Design . . . 56
4.4.1 Building the actions . . . 57
4.4.2 The script building block's function . . . 57
4.5 Reusability . . . 59
4.6 Conclusive remarks . . . 60

5 Conclusion **61**

6 Discussion **66**
6.1 Future research . . . 66
6.1.1 Context . . . 66
6.1.2 Script building block . . . 66
6.1.3 Transitions . . . 67
6.1.4 Turn-taking . . . 67

Bibliography **68**

Appendix A: JADEX to ActionScript®3 **73**

Appendix B: Example used for evaluation **74**

Appendix C: Filled in script building block from second design **77**

List of Figures

1 Glengarry Glen Ross . . . 8
2 OZ Story Editor . . . 11
3 OZ Dialog Editor . . . 12
4 Dialogue example in fire fighter simulation . . . 36
5 The naive action-selection-action-expression model . . . 42
6 The initial script building block . . . 43
7 Possible player choice states . . . 46
8 Non-consequential choice example . . . 47
9 Consequential choice example . . . 48

10 Consequential end choice example 49
11 Context-dependent actions in model 51
12 String replacement 52
13 String replacement example 52
14 Conditional content 53
15 Conditional content example 53
16 Transitions in model 54
17 Transition mechanism 55
18 Final model 56
19 Script building block after evaluation 59

Acknowledgements

My gratitude goes to Ivo Swartjes and John-Jules Meyer, for the excellent support I have had. Whatever the reason was for me being stuck at times - a lack of inspiration, personal issues, et cetera - Ivo always was open to talk and patiently guided me through the whole process of this research.
John-Jules, your wisdom, enthusiasm and smiling throughout every meeting was really comforting and motivated me to keep on working. Thank you!
I would like to thank Thijs Schippers, intern at &ranj, for advice on how to test the architecture.
I would like to thank interviewees Bruno Jordaan and Paul van der Meer, head of game design and game designer respectively, at &ranj, for their feedback on our design.
Lastly, I would like to thank everyone at Ranj for having a pleasant environment to work in.

1 Introduction

We start to explain the field of serious gaming. Then, we explain why we are working on this matter and introduce our new approach to creating games. We briefly explain our philosophical standpoint. We introduce desired properties of characters in a virtual environment. We explain our method to gaining an understanding of our approach. Finally, we argue design choices and their implications to our architecture.

1.1 Interactive characters

There are many games with narrative aspects in which the player will have to speak with non-playable characters (NPC). These games exist both in the commercial gaming industry and the serious gaming industry. Serious games are games that are created to train or educate and this research aids in the development of serious games. The dialogues that take place in those games are often well-written. Yet, the characters are seldom open for much interaction. The reason for this, is that the dialogues in games are most often fully scripted and interaction would quickly cause a combinatorial explosion of possible storylines. Additionally, complex dialogue structures can be hard to represent with current authoring tools. We would like to increase interaction with the NPC's in serious games, because we believe it will offer new opportunities for training and education. Moreover, very interactive characters in commercial games could greatly enhance gaming experience. We intend to create such interactive characters.

1.2 Motivation

In the field of creating interactive narratives, or interactive characters in general, there is a gap between what we would like to have and what we are able to get. Next we will give three examples of what we eventually aspire to. Followed by an example of what we currently have. The first example describes the eloquence we wish for, the second example depicts the world we want to be in, the third example explains the reality we could want, and the last example is what we currently have.

The eloquence of a killer liar:
You caught a serial killer who manipulated other people into killing too. At least, there are many signs which indicate that this is the truth. No proof yet though. You take him to the interrogation room and start asking him about what he did at this and that time. It's not working. He is lying and manipulating you in such a way that the conversation tends to move away from where it should go to prove his guilt. His real intention lurks behind the ambiguity. Can you figure out how to outsmart him?

Game of Rings:
Imagine yourself in Tolkien's glorious world of Lord of the Rings. Free to fight Orcs, Ogres and every other abomination of Middle-earth together with your companions. You're able to speak to every character in this world as if you're really there and alter

the future with an infinite amount of possible scenarios. However, you would need to be a skilled negotiator to become king of Middle-earth or you might end up in a world where you and all your friends have perished and evil has taken over the world.

Hostages in hostility:
Be the commander-in-chief leading a crew trying to rescue hostages held in a bank. You would have to send commands to your forces, negotiate with the hijacker, find out what really drives him, and request information from the police or secret services centre on the hijackers and hostages. Their future lives depend on you.

Glengarry Glen Ross:
You're a real estate agent. The goal is to convince a possible buyer of buying the house. You're able to discuss about the properties of the property. The agent is able to express his opinion about you, about the house and you are able to respond with facts about the house, opinions about the house or questions toward the possible buyer.

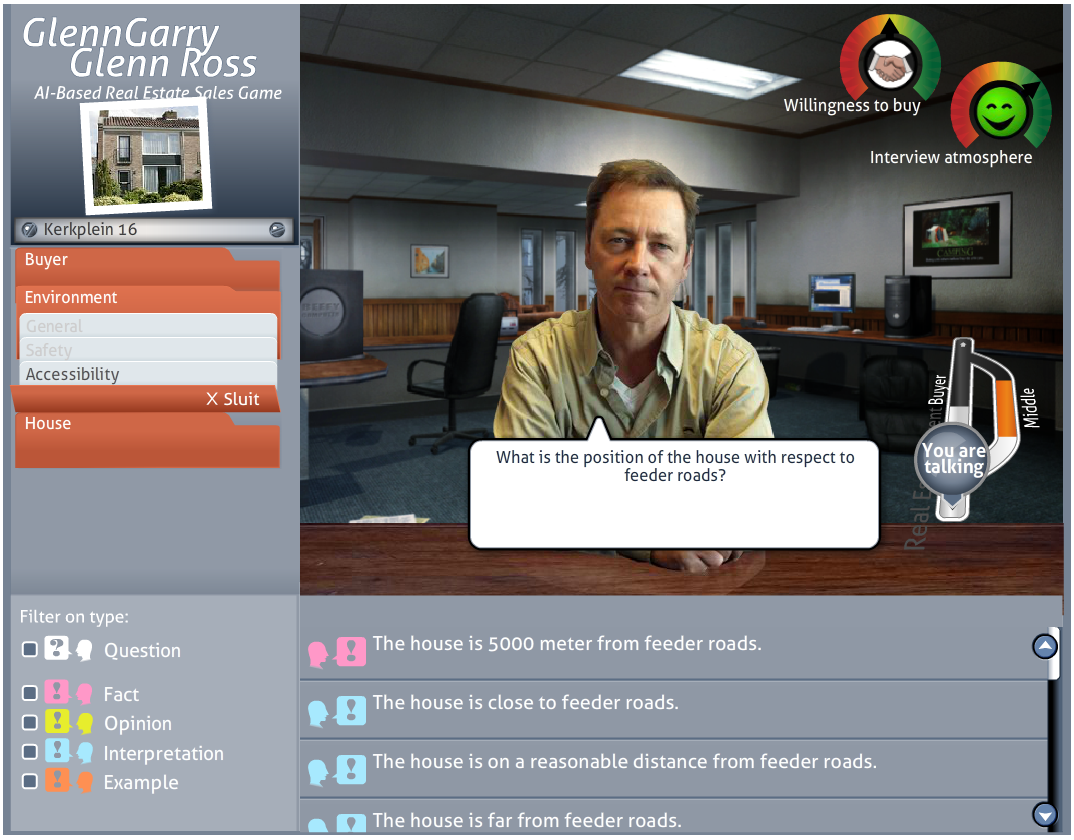

Figure 1: Screenshot of GGR [1]

So what we have is an agent simply stating his opinion X about Y, requiring object K to have property L within a very limited domain. Interactivity is limited to one where, even though a lot of subjects can be handled, the questions and answers always are somewhat of the same form. Moreover, eloquence is absent.

Sadly, we have not yet been able to create a believable interactive narrative as complex as the ones described in the first three examples. Current approaches to create a detective-like game as described above lack either scalability or expressive control [1, 2]. Scripting complete storylines refrains us from writing highly branching stories and autonomous agents often lack expressivity. In a later section we will elaborate on this matter. The same holds for the beautifully named Game of Rings. Here we are the ones who can freely choose how to try to influence the characters in it, the characters should believably change their emotions, expressions, social roles, opinions, et cetera, according to your actions. Also, long-term consequences should be that the story changes correspondingly because of the altered behaviour of the characters in it.

Neither have we been able to create something as realistic and practical as a fully organized scenario-based training (SBT) without any human supervision [3]. SBT is a real-life simulation aimed at education. For example, firefighters often exercise by burning a building with dummies in it, which they then have to rescue.

We would like to have immersive games which train the negotiating skills of players or unsupervised virtual SBT's which could replace regular SBT's. We can create stories which have appealing characters, a sensible sequence of events, wondrous environments and intriguing dialogues. The issue however, in making one of the worlds presented earlier, is creating interactive characters which are social and believable. It is 'interactive' which makes the difference. Interactive means it can respond to actions of the player or other NPC's. In this regard you could call almost any character interactive. Yet, for a character to be socially and believably interactive in one of the worlds presented above, it needs to be able to respond to a wide range of activities and, some sort of autonomous reasoning, albeit a mere illusion, has to be given. For a player to experience agency - a sense of control - a lot of actions should branch to a different storyline and the amount of possible storylines would no longer be graspable by storyline editors to be predetermined [4]. Moreover, a very interactive and swayable character could eventually have goals that conflict with the desired storyline. This conflict is called the "narrative paradox" [5] and buoyantly tailgates everyone who ventures in the world of interactive narratives. There are responses to the issue of combinatorial branching of possible storylines in an interactive narrative [4] . However, these always include restricting the freedom of in-game characters or restricting the freedom of possible scenarios to which a story can go.

We call an interactive character an agent. We would like to design a method where we have more expressive control over an agent while still retaining scalability, reusability and proactiveness. This way we hope to come closer to creating the, currently perhaps overly ambitious, games exemplified above.

1.3 Current practices in the field of serious games

Since 1999, &ranj (Formerly named Ranj) has been working on the gamification of tasks and the creation of serious games. Gamification refers to "*a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation.*" [6] To put it in simple words: "put game elements in tasks to make them more interesting".
The goal of a serious game is to educate or train the player. Because the training or education is in the form of a game, it is believed people are often more interested; where would you rather train your negotiation skills: from a book or in a world resembling that of Lord of the Rings? We think that people who are interested generally perform better. Also, in a game the authors have control over the flow of the game. People who experience a nice flow are fully involved in the present moment [7]. A task should never be too simple; people will get bored and lose interest. A task should never be too hard either; people will be demoralized and give up [8]. With serious games we also have the possibility to teach skills which would not be learned in regular training because certain events would normally be very rare to occur or are for some reason physically unworkable. SBT's, for example, are very time consuming, hard to perform and financially troubling [3].

1.3.1 Scripted dialogues and storylines

Currently, almost all consumer dialogue-based games are created with authoring tools. These are used for producing dialogue content without programmer intervention; we call this scripting. Scripting gives artists of the game full control. However, as stories get complex, artists will have difficulty maintaining a good overview on the whole to create a sound story. Even if they are brilliant story editors, it quickly becomes too much to comprehend. Secondly, determinism is in many cases, like in the examples given in the motivation, not desired. Lastly, it does not leave much room for some form of initiative, like changing the topic halfway a dialogue or a turn-taking agent.

At &ranj[1], games are developed for education, training, recruitment, communication and personal health care. Until now, games are fully scripted. Scripts are written with an authoring tool called the OZ Dialog Editor, to which we will come back later this subsection.

The general structure of the game is implemented with the OZ Story Editor (see figure 2). The Oz Story Editor works a bit like a trigger system. In the example presented below a dialog file gets opened if interventieEmilioBonusToegepast (which means that you decided to give Emilio a bonus) is true in week 2. When the dialogue is finished and 5 milliseconds have passed, it checks whether interventieIedereenBonusToegepast (you gave every employee a bonus) is true or not. If it is true, an image of a cash register opens; if not, then an image is opened where the other employees look unjustified.

[1]Formerly called Ranj

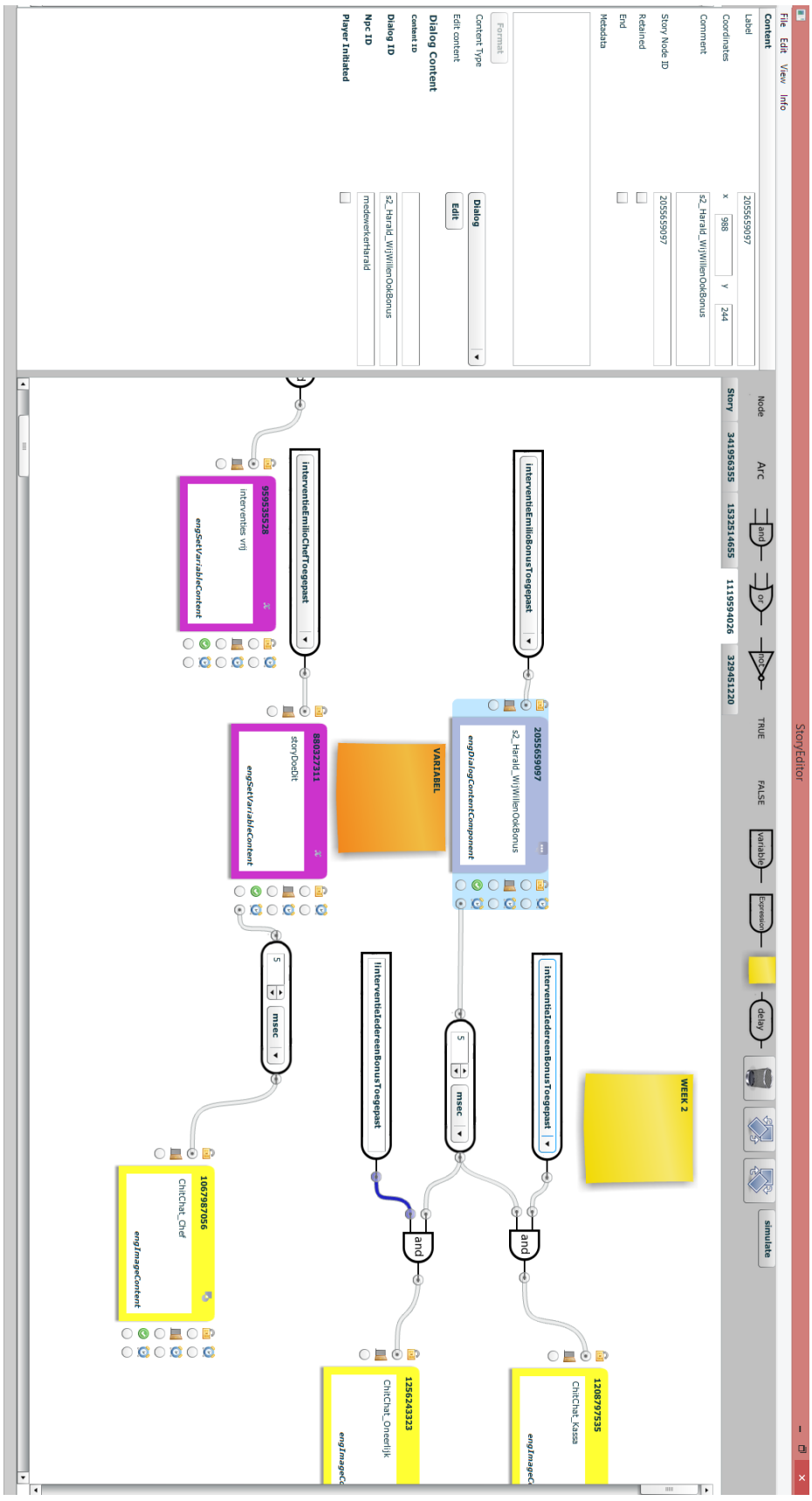

Figure 2: Screenshot of a OZ Story Editor file.
In the light blue (selected) box, a dialog from the OZ Dialog Editor is opened, in the fuchsia boxes (straight below the selected one and the most left one) a variable is set, and in the yellow boxes (the remaining) images are opened. This way a game gets its structure with the OZ Story Editor.

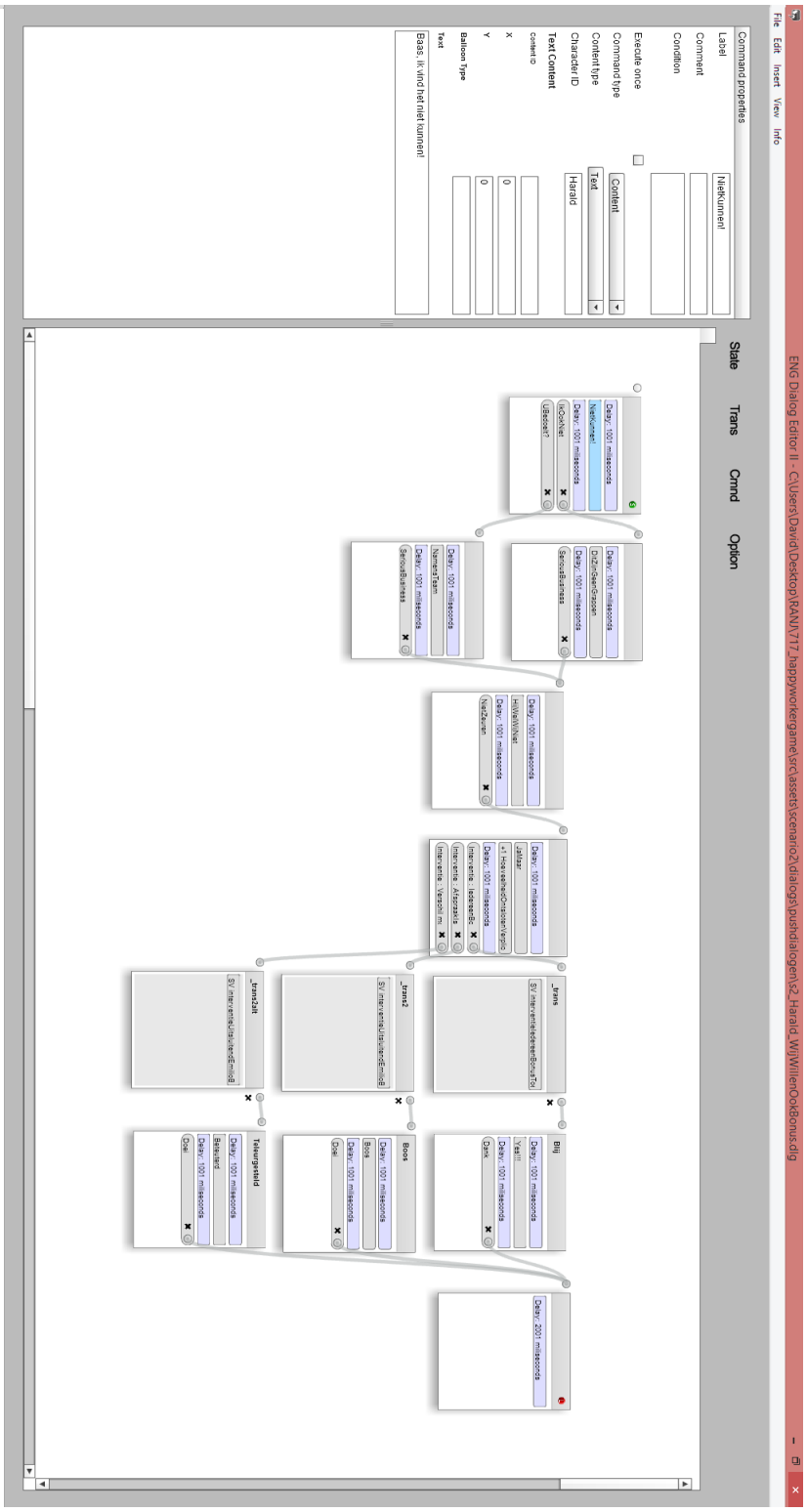

Figure 3: Screenshot of a OZ Dialog Editor file.
A script consisting of 12 states. The first state contains 5 commands: a Delay (waiting time), followed by an utterance of an employee “Baas, ik vind dit niet kunnen” (dutch for: “Boss, I find this unacceptable!”), followed by another delay. The last two commands are 2 options, the player can choose to be a wiseguy and say “Yeah, I agree. This is unacceptable!” or it can ask “How do you mean”?. As can be seen, even though a different state will be opened for the different answers, no variable is set during either of the states. Moreover, after either of those states the dialogue transits to the same successive state. Only in the 4th state, relevant choices are made. One of the three possible interventions will be chosen and these will change variables which in the OZ Story Editor. This will have an influence on which dialogues shall be opened later on in the game.

A dialogue from the OZ Dialog Editor is a finite state machine (see figure 3). This means that anything from one utterance by one participant in the game to a complete game with action and reaction by several players could possibly be put in one script.

A finite state machine is not always ideal. Imagine a case where when we are at one of the states A1,..,A10 we can say 3 different things and saying one of them at the right time from one of those states will cause us to go to a different part of the story, let's call it B, which is only reachable through this action at this moment. Once part B is finished we have to go back to the state from which we came to finish this state. However, the finite state machine has no memory of where it came from so here we already feel the limits of scripting with the OZ Dialog Editor. Of course, we could use the OZ Story Editor to catch those procedures which, like the ones described above, are very inconvenient to model with finite state machines. However, even quite simple situations already cause combinatorial explosions, as is also made apparent by other work [9]. Bartish and Thevathayan have shown that complexity measured as a function of the number of behaviours, was linear for agents and quadratic for finite state machines [10]. Script authoring tools do not necessarily have to use finite state machines. Still, in any case scripting implies relatively simple storylines, because in any case the vastly branching tree representing our possible narrative paths has to be contrived. Since scripting alone is insufficient for creating the interactive characters we desire, we look for different approaches to creating an NPC.

1.3.2 BDI

The Belief, Desire (goal), Intention(plan), paradigm of Bratman [11] is based on folk-psychology. When people explain their reasoning to others, it feels natural for them to do this by describing their beliefs, desires and intentions.

Beliefs represent what is believed to be true in the world. Desires are what is desired to become true in the future. Intentions are pursued desires, often because a specific desire has a corresponding plan of which the agent believes can be successfully completed within marginal time. Reusability, scalability and proactiveness are some of the advantages of BDI [1, 10, 12, 13]. It offers us the possibility to create complex storylines and agents which can actively take the turn to act.

Glengarry Glen Ross

In a fully scripted game, every possible dialogue in the game is written beforehand and every possible sequence of dialogues is predetermined. This results into nicely written dialogues but also refrains them from writing very complex storylines. Moreover, it does not allow for real-time turn-taking.[1]

TNO - the Netherlands Organisation for Applied Scientific Research - together with Utrecht University and RANJ, created a game called Glengarry Glen Ross inspired by the equally named movie [1]. They used a BDI-framework with language generation

[1] Turn-taking refers to a mechanism which enables both the player and NPC to act at certain moments in time.

through using standard templates in which you had to fill in the variable words to create a fully autonomous quasi-emotional agent, which also had the capability of initiating its own turns [1]. Here it became evident that, even though the agent performed well in general, language generation often results into unnatural sentences and poorly expressed dialogues. Nevertheless, the project still showed that BDI could be used for interactive serious games. More specifically, if we could find a way to bring more dynamics and nuance in the use of language it could also be used for training court cases, interrogating suspects and witnesses, or in any game which focuses on training negotiating skills for that matter. What these games have in common is that their focus is not on a strong story, but rather strong autonomy. In these games it does not matter much which topic comes when or how it ends. The focus lies on games where the player can try different communicative tactics to observe the reaction of the agent and evaluate his or her tactics.

1.4 Combine BDI and scripting

To achieve more believable flow and nuanced dialogues, Ranj decided to integrate the JADEX reasoning engine [14] to ActionScript®3 [15]; proceeding their research on the use of BDI for serious games. We intend to connect scripted dialogues to certain actions of an NPC. The NPC will be a Belief, Desire, Intention Agent (BDI), which will select actions at a high level - this could be something like "express my feelings to her". Actions will be a result of its intention; the action chosen - "express my feelings to her" - will then be expressed through a scripted dialogue. A first move on combining scripting and BDI has already been made [16]. They connected actions from the agent to the OZ Dialog Editor and expressed them with the OZ Dialog Player. However, not enough work has been done to encounter the technical and conceptual difficulties that arise in doing so. This is where we continue.

We propose to combine scripted dialogues with a Belief, Desire, Intention framework by coupling the pre-written dialogues to higher-level actions of an agent. The expression of a high-level action could be very nuanced and detailed, but the agent chooses when it is applicable.

RQ1: Can we gain an understanding of the process of creating a game with our approach of BDI and scripting?

RQ 2: Can we retain merits of BDI, like reusability, scalability and proactiveness, when combining it with the narrative control of scripted dialogues?

1.4.1 Why dialogue-based games?

Keep in mind that any piece of script like "kiss" could easily be realised in a game with a physical environment by showing the animation or let the gamer act the animation via controls on their controller. This indicates that the use of this research is not limited to dialogue-based games. We opt for dialogue-based games because building a dialogue-

based demonstration game would require less work compared to building a interactive narrative with believable animations.

1.5 GOFAI/Interactionist & Expressive AI

In contrast to the standard way of creating an AI, GOFAI or Interactionist, we embrace the expressive AI way for our research. The good old fashioned, or symbolist, AI-ists work to create an intelligent mind. A mind for a general solution. The Interactionists work from a point of view where intelligence must be embodied and can only be created reactive and specific to their environment. While standard AI is focused on Task Competence, Objective Measurement, Generality, and Realism, cultural production is focused on Poetics, Audience Perception, Specificity, and Artistic Abstraction [17]. Games cover both the field of AI and cultural production. An AI like in the examples in the motivation, should play with ambiguity and have several layers of meaning to them; they should have a story. The success of this approach can not be objectively measured. Not until the audience is able to participate in the poetics defined by the artist, will success be achieved. In this sense it is not objectively measurable, but completely dependent on audience perception. General AI tries to achieve some general knowledge. In arts, artists try to let the audience experience a specific process. Similarly, in serious games we want to educate the audience; bring across a certain message. Moreover, to bring across this message we do not necessarily have to create realistic situations. Often, some artistic abstraction can suffice. Thus, we arrive at expressive AI.

> *"The concern is not with building something that is intelligent independent of any observer and their cultural context. Rather, the concern is with building an artifact that seems intelligent, that participates in a specific cultural context in a manner that is perceived as intelligent."* [17]

In our process to reach the goal of developing social and believable agents we work from an approach where it is more important that an agent acts *as if* it is social and intelligent according to the situation than whatever "intelligent" the reasoning behind its actions is. Two terms stand at the base of expressive AI: 'interpretive affordances' and 'authorial affordances'.
Interpretive affordances are signs which indicate which actions can be taken.

> *"Interpretive affordances support the interpretations an audience makes about the operations of an AI system[...]Interpretive affordances provide resources both for narrating the operation of the system, and additionally, in the case of AI-based interactive art, for supporting intentions for actions that an audience may take with the system."* [17]

We are creating agents for an environment where it has to communicate with players. Hence, we argue that it is important that the agent shows interpretive affordances such that the player knows what is happening and what it could or should do. Players will try that which was afforded and experience cause and effect relationships. Secondly, we

try to find a way in which the artist can express itself through the agent. These are the authorial affordances the framework offers.

> *"The authorial affordances of an AI architecture are the "hooks" that an architecture provides for an artist to inscribe their authorial intention on the machine."* [17]

By adding more expressive control by combining BDI with scripting, we hope to offer the authorial affordances to the artists, so that in turn, they can offer the interpretive affordances to the audience.

1.6 Socialness and believability: what affords interpretivity?

To be able to create an agent that affords interpretivity, we need to get some understanding of what being believable and social means. For example, as a character, the agent should overly express its feelings. Additionally, it should act as if it is aware of past actions and environment. As a game master, it should know which parts of the narrative are fit at any moment. Our architecture should offer the authorial affordances such that the artists can inscribe these desired properties in the characters of the game. In general, we say a believable and social character allows for suspension of disbelief, which literally means what it says. As long as a person can suspend its disbelief, it believes in what he sees as a coherent character or story. This does not mean that this person finds the character or story realistic, honest or convincing. It just means that this person accepts the character or story as it is, and allows for himself to be emphatic with the character or be immersed in the story. We emphasize this because this is often overseen by people and we have to keep in mind that this believability is imperative. An unbelievable character is uninteresting.

1.7 Approach and validation

In this thesis, we try to get an understanding of the process involved in the development of an interactive narrative with our proposed approach. Which steps have to be made before a game is created? First, we create a BDI agent. This agent has beliefs about the world and some reasoning about his goals and intentions, which causes him to try to execute certain actions. If actions are expressed with scripting, how do we translate the action of a BDI to a script? As an example, the agent has the intention to lie about his whereabouts of the previous night. Now we have some action lieAboutLastNight and some script in which this dialogue is expressed. There is a gap between the action and the script, and we try to fill in this gap. What kind of information is required before an action can be translated to a script? We try to encapsulate this in building blocks. What kind of authorial affordances does scripting offer? We try to encapsulate this in design patterns. We aim to formulate certain design patterns and to create paper-prototypes of the building blocks we think are crucial for the game designers and programmers to construct an interactive narrative. We design an exercise for employees of &ranj. In this

exercise a game designer will have to work with the information given by us; exclusively using the building blocks presented to them. We will have direct feedback on what is salient and what is missing. By using more than one game designer this will be an iterative process where we adjust our view on both the function and the shape of the elements of our architecture.

1.8 Practical choices, limitations, and building blocks

With our approach of combining BDI and scripting, we envision that artists often get the choice whether to break down one high-level action into several lower level actions, together covering the same concept, or to keep the high-level action as a whole; thus limiting interactivity, but increasing control. Additionally, they can determine whether some player's choice within a script influences the agent's state (beliefs, desires and intentions), or whether the player's sense of agency is an illusion. The latter refers to the case where the player's choice does not influence the agent's state, even though it may seem so, these differences are described in section 4.3.3. We would like to formulate rules of thumb, as advice to take a certain choice. In the end anything is possible and it is all mainly subjective. The latter can easily overwhelm the artists. Making any choice at all can be demanding if confronted with too many options [18].

1.8.1 Strict segregation between action-selection and action-expression

In the process of creating a game, visual designers, game designers, and programmers cooperate. Our proposed approach for creating interactive narratives covers both the task of game designer and the programmer. At &ranj (and other gaming studios), programmers and content authors (game designers, visual designers) can largely work separately. Ideally we would like to sustain this task-segregation. Therefore, we would like to keep action-selection segregated from action-expression. Programmers would design an action-selection mechanism and game designers work on reaching the audience.

We attempt to sustain the strict segregation of the tasks of the programmer and the game designer, respectively assigning them the task of action-selection and action-expression

This means that once a script is opened, we do not change the agent state until the script is closed again. To change the agent state during a script would mean to give a game designer a programmers task. Additionally, the idea of expressing actions with scripting is lost when you fiddle with the agent during scripts, we further substantiate this statement in section 4.2. The latter however does not infer that it is impossible to send information from the agent to a script. It simply means that once a script is opened we can not then decide to retrieve more data from the agent to the script.

We can not update the agent state during a script and we are not able to retrieve data from the agent during a script. Can we draw the consequences?

The architecture we intend to design should incorporate the development process and task-division of the production of games. At which stages of the development are the actions of the game defined? Who define(s) them? A game designer? A programmer? Both? The answers to these questions influence the shape of the architecture. In section 4.2, we address this issue in more detail.

1.8.2 Building scripts

We focus at how, once an action is chosen, it can be expressed by a script. Also equally important, we focus at how we define actions. As stated earlier, in theory we could fit an entire game in one script and the action would be "play game". This means that at any moment we have to think whether we want to script the next step of all possible narrative paths after a player has acted, or whether we want to use the agent to decide the possible narrative paths.

RQ 3: How should the size, or level-height, of an action be decided?

1.8.3 Combinatorial explosion of context-independent scripts

The expression of one action for pursuing a certain intention, can differ at different states of the world.[1] Here we encounter an issue: some actions are very context-dependent. Thus, we would like a way to use variable content in scripts. Otherwise we would have to create a different plan and script for every possible context. We make case for this statement in section 4.3.4.

Can we design a structure where information from the agent is sent as a list of arguments which can be used during the running of a script such that variable content can cover possible contexts?

1.8.4 Connecting scripts

Once a script is ended and the agent has selected a new action, which then again opens up a script, we need to provide the interpretive affordances which make the audience understand the relation between those two actions. Our expectations for the need of some explanation for changes in behaviour, are supported by work of Bruner.

> *"If we look at someone doing something, we are actually trying to figure out why he is doing it, not what he is doing".* [19]

Given that we want the agent to express the reasoning behind its actions, is it important for the agent to express the reason behind its mental state transitions as well? If so, how

[1] With a different state, we mean that the logical formula describing the world is non-equivalent to the one describing the other.

can we model these transitions in such a way to find a good balance between expressivity and both generality and modularity?

1.9 A look back

We started to explain the field of serious gaming. Then, we explained why we are working on this matter and introduced our new approach of combining BDI and scripting. We briefly explained our philosophical standpoint. We introduced desired properties of characters in a virtual environment. We explained our method to gaining an understanding of our approach. Finally, we argued design choices and their implications to our architecture.

The research questions in this thesis are:

- RQ 1: Can we gain an understanding of the process of creating a game with our approach of BDI and scripting?
- RQ 2: Can we retain merits of BDI, like reusability, scalability and proactiveness, when combining it with the narrative control of scripted dialogues?
- RQ 3: How should the size, or level-height, of an action be decided?

2 Background and related work

This chapter is divided in three subsections. The first section, philosophy of AI, places our approach to creating an AI in contrast to more common approaches. This approach has had an influence on the reason for choosing our approach of combining BDI and scripting, as well as an influence on the eventual architecture, as can be read in the design chapter.

In the second section, desired properties of virtual agent-behaviour, we describe properties we think are necessary for creating dialogue-based serious games with strong characters.

Lastly, the methodologies section. Here we describe BDI, our agent framework of choice, and discuss language generation and processing and why we think it is currently not sufficient for creating dialogue-based serious games with strong characters.

2.1 Philosophy of AI

As stated above, in this subsection we discuss our philosophy to creating an agent in AI and compare it with the two most prominent approaches in AI.

In the introduction, we argue that a different approach to creating an AI could be more suitable for the development of serious games. We describe the GOFAI/Interactionist debate, which is a debate on what an AI should be. GOFAI supporters assume there can be some intelligent system which functions universally. Interactionists assume that functionality is always dependent on body and environment. We argue, that if an audience is to be impressed, it does not matter whether the agent is intelligent at all. The latter is in line with expressive AI [17].

2.1.1 GOFAI vs Interactionist

Mateas has phrased the GOFAI/Interactionist debate very concisely. Thus we quote:

> *"GOFAI is characterized by its concern with symbolic manipulation and problem solving. Its proponents draw a firm distinction between mental processes happening "inside" the mind and activities in the world happening "outside" the mind. GOFAI's research program is concerned with developing the theories and engineering practices necessary to build minds that exhibit intelligence. Such systems are commonly built by expressing knowledge in symbolic structures and specifying rules and processes that manipulate these structures. Intelligence is considered a property that inheres in symbolic manipulation "inside" the mind. This intelligence is demonstrated by the program's ability to solve problems. Where GOFAI concerns itself with mental functions such as planning and problem solving, interactionist AI is concerned with embodied agents interacting in a world (physical or virtual). Rather than solving complex symbolic problems, such agents engage in a moment-by-moment dynamic pattern of interaction with the world. Often there is no*

> *explicit representation of the "knowledge" needed to engage in these interactions. Rather, the interactions emerge from the dynamic regularities of the world and the reactive processes of the agent. As opposed to GOFAI, which focuses on internal mental processing, interactionist AI assumes that having a body embedded in a concrete situation is essential for intelligence. It is the body that defines many of the interaction patterns between the agent and its environment."* [17]

So, GOFAI supporters and Interactionists focus on an agent which functions, whether there is an audience or not. However, we want to focus on creating agents where there is an audience, and its perception is what matters.

2.1.2 Expressive AI

Where most AI research so far has focused on which actions to take, in this paper we also focus on how actions should be presented. The AI system is built in order to communicate the ideas of the artists to the audience with the idea that the general AI approach might not be very effective for creating interactive narratives. Tom Porter describes this problem, in contrast with action-selection, as the action-expression problem [20]: what can the agent do at every point in order to best communicate its actions and thinking to the user? As mentioned earlier, two concepts stand at the base of this approach: interpretive affordances and authorial affordances.

Interpretive affordances

Interpretive affordances are the 'hooks' supporting the audience interpretation of the operation of an AI system [17]. A coffee mug has a handle which indicates that it can probably be held, and more specifically, it can probably be held most efficiently at that particular place. Not all interpretive affordances have to be so obvious. In a dialogue between two humans a gaze towards the other human, after having made an utterance, indicates waiting for a reply [21]. It is a sign which says "Hey, I want you to take the turn to act now". If a person starts running after showing a scared face, we can deduce that he is probably fleeing from something. Likewise, if he says he is training for a marathon and starts running we can deduce he is most likely going to train. In a similar way, certain details in a conversation can suggest what is going on.

Thus, the agent signs why certain choices are made, as interpretive affordances. We have given an idea of the concept but what affords interpretivity? The agent needs to express itself in a believable and social manner in such a way that the player can use Theory of Mind [22] on the agent, in other words, interpret the agent and his actions. This allows the player to guess what drives the agent's actions and adjust his or her behaviour accordingly.

What it means to be believable and social will be elaborated in sections 2.2.1 and 2.2.2 respectively.

Authorial affordances

The authorial affordances of an AI architecture are the "hooks" that an architecture provides for an artist to inscribe their authorial intention on the machine [17]. The authorial affordances equip the programmers and game designers both with an arsenal for their individual asks as exclusive groups, as well as an arsenal which helps with the intercommunication that is equally essential for creating the interactive narrative.

We design mechanisms with which they can overcome issues like combinatorial explosion of plans which emerge from highly context-dependent plans. We propose design patterns for the game designers for their task of writing scripts. These are examples of the authorial affordances an architecture can offer. Also, not very coincidentally, these are authorial affordances of our architecture.

2.2 Desired properties of virtual agent-behaviour

Here, we describe properties we think are necessary for creating strong characters in dialogue-based serious games.

2.2.1 Believability

Have in mind that believability can be a confusing term. It does not imply realism, honesty or convincibility. For example, a realistic character would sulk just a little or even keep its sadness completely to itself whenever anyone would call him ugly. In contrast, believable agents would make overly exaggerated body movements to express their feelings solely for clarity towards the audience. Also, people in general are predisposed to believe in completely unrealistic characters like a flying elephant and raise disbelief whenever they look at a near-human animated character. The latter is also referred to as "The Uncanny valley" [23].[1] In general we could say that a believable character is one that allows for suspension of disbelief. As long as a person can suspend its disbelief, it believes in what he sees as a coherent character or story. This may sound like a rather tautological remark, but it is important to notice that suspension of disbelief does not have to be achieved through realism, honesty or convincibility.
To strengthen our point:

We have here an agent called Harry the retarded rock. He is always laughing and his only other ability is to say 'Hurry' while falling down an infinite slope which can be found behind the door to the dimension of infinite slopes. The other characters in the game can sometimes go to this dimension to visit Harry and "hang out" with him. They like him because he is a very happy rock and they will talk to each other about how awesome Harry is. In this context even Harry could be a believable character, while in no way Harry is realistic or convincing.

[1] The uncanny valley is a hypothesis in the field of human aesthetics which holds that when human features look and move almost, but not exactly, like natural human beings, it causes a response of revulsion among some human observers

Process of believing and threshold for breaking suspension

People can partially choose whether to suspend their disbelief. Nevertheless, this process is often subliminal. Some easier experience immersion because they suspend their disbelief more easily than others. Research has shown that in general people are open-minded towards unrealistic environments and characters [24]. For example, it would be expected that a lack of coherence between different aspects in a game would break suspension of disbelief. However, subjects in a test to find the thresholds for breaking suspension of disbelief, would not even notice physical changes like altering the height of a jump tenfold [24]. What exactly causes people to suspend their disbelief is not known, but we do have some intuition on it. The following paragraphs are describing some of these intuitions and how we have come to them.

What can we learn from existing animations?

Artists have been making believable characters for quite a while now. We can learn mostly, but not only, physical aspects for believability from Disney animators Ollie Johnston and Frank Thomas, who already showed us 12 basic principles of animation from their studies on all the characters made by Disney ranging from the early 30's to the moment they published their book "The illusion of life" in 1981 [25]. We will elaborate on the two most relevant to text-based game:

1. **Appeal** in a cartoon character corresponds to what would be called charisma in an actor. A character who is appealing is not necessarily sympathetic – villains or monsters can also be appealing – the important thing is that the viewer feels the character is real and interesting [25]. Characters' emotions need to be expressed broadly. That is, emotions must affect everything about the character: the way it moves, the way it talks, the expression on its face [25].

   > "If the character does not react emotionally to events, if they don't care, then neither will we. The emotionless character is lifeless, as a machine." [26]

2. **Anticipation** creates an opportunity for the audience to focus their attention to what is relevant. A character's eyes moving to the right changes the audiences attention to move to the right, and scary music evolving increases overall attention because something exciting is probably going to happen. For example, an utterance like "AAARHG" could draw people's attention and make them think someone is either a pirate or severely hurt himself; depending on the context of course.

What can we learn from existing narratives?

We can learn from previous research done for the Oz-project led by Joseph Bates [27]. This group focuses on creating interactive dramas and acts from a narrative perspective. They also listened to the artists and stated three fundamental lessons to learn from them [28]:

1. **Believable agents may not be intelligent:** "We don't want agent architectures that enforce rationality and intelligence. AI systems designed for these goals would be inappropriate for building believable agents."

2. **Believable agents may not be realistic:** "We don't want architectures that enforce realism. This means that we don't want systems that only generate externally realistic behavior. It also means that we are willing to use unrealistic internal processing – the goal is not to create cognitively plausible agents; it is to create good characters."

3. **Believable agents will have strong personalities:** "Personality should permeate the architecture and should not be constrained more than absolutely necessary."

Without strong personality no Theory of Mind

Especially the last lesson from the previous subsection is paramount. Having a strong personality is strongly correlated to expressiveness (remember we work from an expressive AI approach). The ability to understand that others have beliefs, desires and intentions that are different from one's own and to attribute mental states to oneself and others is called Theory of Mind. Whenever we look at someone and we try to use Theory of Mind on that person we are not actually looking at what he is doing, we are trying to figure out why he is doing it [19]. A strong personality and exaggerated character often pairs with clear motivations. We show this with some examples from the motivation section.

You ask a witness if he has seen anything suspicious. He says "ehhh no" with a firm face. While actually he is saying this because he is scared to get hurt. How should the player know about this, if emotions are not expressed?

Or if you're Aragorn, the supposed-to-be king and your future depends on proper negotiating skills. You would want other characters to stay in character or, next to the fact that it would raise disbelief and decrease immersion greatly, you could never predict the consequences of your actions and the game would be quite boring. NPC's randomly wandering off or getting angry can be frustrating. Instead, when you are negotiating with a corrupt king, which stays in character, you could take his personality into account to adapt your behaviour accordingly.

As the commander-in-chief in a hostage simulation, you would like the hijacker to stay in character and have a distinctive personality. The point of the training would be lost if the hijacker is on the phone telling he will not hurt hostages as long as he gets his airplane and money and, like some schizophrenic [29], suddenly decides to move all the hostages outside and randomly shoot background actors without any indication on why he changed his behaviour.

Believable agents should act aware of past actions

Johanssen [30] found that the most essential aspect of creating believable NPC's according to the respondents, was memory of previous interactions and emotions. So, not

only do we expect that we should be able to apply Theory of Mind on agents, but we also expect agents to apply Theory of Mind on us by making a model based on history and personality. This is something you would contribute to social agent behaviour. Thus, in narratives the agent needs to act as if it reasons about the past actions. One way is to keep track of what signs he sent to the viewer so far and decide what next expression should be made to best show his intentions [31].

Rules and norms of the environment

One could also argue that an agent should know the rules and norms of their environment. A character which finds himself within the same community as the player should act according to the rules and norms this community dictates. Of course, it can choose to deviate. But this would probably be done by an agent with a controversial personality. Unwritten social rules will also reduce the schizophrenia agents often show [29]. They automatically will act as transition rules between two or more actions. Often it is very rude to say nothing to explain your actions and just walk away from the one you were socializing with just one second ago.

2.2.2 Socialness

As humans we apply social models to practically every complex entity (yes, we also apply social models to cars, desktops and so on) in our world. As long as the entity's behaviour adheres to our social model the entity seems believable. Because this research will eventually be applied to human-like AI's we will mostly consider human socialness.

Cynthia Breazeal argues the following [32]:

> *"Right or wrong, people rely on social models (or fluidly switch between using a social model with other mental models) to make complex behaviour more familiar and understandable and more intuitive with which to interact. We do this because it is enjoyable for us, and it is often surprisingly quite useful. We argue that people will generally apply a social model when observing and interacting with autonomous robots. Autonomous robots perceive their world, make decisions on their own, and perform coordinated actions to carry out their tasks. As with living things, their behaviour is a product of its internal state as well as physical laws. Augmenting such self-directed, creature-like behaviour with the ability to communicate with, cooperate with, and learn from people makes it almost impossible for one to not anthropomorphize them (i.e., attribute human or animal-like qualities). We refer to this class of autonomous robots as social robots."*

Secondly, she argues that the previous statement is from a human's perspective. From a robot's perspective she divides socialness in four subclasses [32]:

1. Socially evocative agents: these are designed to encourage people to anthropomorphize in order to interact with it, like with the Tamagotchis. The human attributes

social responsiveness to the robot, but the robot's behaviour does not reciprocate.

2. Social interface agents: This subclass of robots uses human-like social cues and communication modalities in order to facilitate interactions with people. Because this class of robot tends to value social behaviour only at the interface, the social model that the robot has for the person tends to be shallow (if any) and the social behaviour is often pre-canned or reflexive.

3. Socially receptive agents: Whereas the benefits of socially communicative applications (e.g., those described in the previous paragraph) are dominantly measured from the human's perspective, socially receptive robots also benefit from interactions with people. Such examples often involve robots that learn from interacting with people through human demonstration (following a training model), such as acquiring motor skills, or a proto-language. They are socially passive, however, responding to people's efforts at interacting with them but not pro-actively engaging people to satisfy internal social aims.

4. Sociable agents: pro-actively engage people in a social manner not only to benefit the person (e.g., to help perform a task, to facilitate interaction with the robot, etc.), but also to benefit itself (e.g., to promote its survival, to improve its own performance, to learn from the human, etc.)

We differentiate three types of social:

1. Social as in this is a social act. This definition takes a social agent in its broadest sense. This refers to taking actions which have an effect on others within an environment.

2. Social as in he is a very social person. This holds that someone takes into account the norms and values and uses a Theory of Mind on other people to take their feelings into account. A sociable agent is equal to this except for that it can also choose to ignore norms and values and abuse someone's feelings for his own good.

3. Social from the definition of Breazeal as defined by the four subclasses described above [32]. Here, an AI is social from as early as when we tend to anthropomorphize them.

From now on when we write social we refer to the third definition of social. As our goal is to create sociable agents, we will use 'sociable' where it is appropriate.

What is being social/sociable?

Being sociable has several aspects to it. As explained by Johansson [30], being sociable is something that is defined by your surroundings and is very much related to acting believable. If you act as if you don't understand the social and law given rules you could behave unbelievably and it is hard to contribute some social model to your

behaviour which corresponds to one which coheres to a social model of someone which has an understanding of his environment and his interactions with it.

The other side of the definition of being sociable is being communicative and interactive through natural language; spoken, signed or written. Humans are most often being socially communicative in a spoken and signed way. As said earlier, we expect social agents to create a model of us. They should apply some form of Theory of Mind on us to be able to act sociable towards us. If they do not use a Theory of Mind they will not understand what actions to take.

Theory of Mind

We argued before that a strong personality is fundamental for a believable character. A strong personality is realised by adding emotions and expressions to actions. We then find some coherence between the actions, emotions and expressions to compose a social model. The social model is a subjective definition of the personality which stands before us. We then use this model to create a Theory of Mind on that personality. This Theory of Mind is used to predict the consequences of our actions. Obviously, this is not something that is always done correctly. So, we want an agent that can believably, and sometimes realistically, make mistakes with respect to using a Theory of Mind. Sengers [31] designed the Expressivator for this. A mechanism which remembers signals expressed towards others and uses this like a simplified form of Theory of Mind. We essentially do this by storing beliefs such as "discussedVillageLife", as can be seen in Appendix B. In the current philosophy there are two prominent accounts on human, adult Theory of Mind: Theory-theory and Simulation theory [22].

Theory-theory

This theory asserts that individuals hold a theory of psychology to infer mental states of others like beliefs, desires and intentions; it refers to the way humans think that they reason. This theory is used to understand the reasoning behind the others' actions and predict future behaviour. Learning here is by revising rules in your theory. Representation of knowledge here is clearly related to the BDI-framework as used in Muller et al. [1].

Simulation theory

Proponents of the simulation theory claim that Theory of Mind is the ability to project ourselves into another person's perspective, and simulate his or her mental activity with our own capacities for practical reasoning. So, we use our own reasoning and a full definition of the other's mental state is not required.

Joint activities and joint actions

As sociable agents have to interact, they sometimes have to engage in joint activities. Examples of joint activities are dialogues and sports competitions. Joint activities can be composed of several other activities or behaviours. These behaviours can both be joint or individual. For example, going into a horse race, the joint activity is both doing a

horse race; starting at the same time and place and ending at the same place. Agent one however, wants to go to its own horse just as agent two wants his horse and both want themselves to win and the other to lose. So only the main goal is joint, the subgoals are not joint. On the other hand, but similarly, the agent could have as joint goal to create a nice gaming experience for the player as an actor, while working on contradictory goals on a character level.

An agent could propose a joint behaviour to several other agents. One example of how such a joint goal could be initiated by Mateas and Stern [33]:

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

2. If a joint behaviour 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 behaviour (and behaviour children) to the ABT (active behaviour tree).

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

2.2.3 How are socialness and believability related?

The way we communicate and the way we express ourselves emotionally is very important for acting as a social being and the way we feel about someone is influenced by what we think about his social behaviour.

> *"First, social factors are very important in determining emotions. Many causes of emotion in people and in artistic characters arise from social factors. For instance, anger, love, hate, jealousy, and grief are often associated with other agents in the world. Without social knowledge and relationships with other agents, there would be a large gap in the types of emotions our agents could express."* [28]

You need to be social to be believable, emotions need to be expressed broadly to act believable, you need to be sociable to know which emotions to express, and you need to be believable to be socially interpreted. We could argue that these terms are almost inseparable.

2.2.4 Turn-taking

Turn-taking refers to the process of initiating an action during a joint action often, but not necessarily, referring to saying something in a dialogue. Sacks et al. [34] have worked on investigating turn-taking as a mechanism:

> *"Turn-taking is used for the ordering of moves in games, for allocating political office, for regulating traffic at intersections, for serving customers at business establishments, and for talking in interviews, meetings, debates, ceremonies, conversations etc.-these last being members of the set which we shall refer to as 'speech exchange systems'."* [34]

Turns, as they state, can be taken at a transition-relevance place (TRP) according to a set of rules [34, pp. 704] which are based on turn-allocation components of the form of (a) those in which next turn is allocated by current speaker's selecting next speaker; and (b) those in which a next turn is allocated by self-selection. Self-selection is a process which depends on the character of the entity that may take the turn. For example, the CEO of a company will be more assertive in taking the turn, or even interrupting someone, than the janitor. Prendinger et al. [35] took this idea and showed that social role awareness improves the believability of turn-taking agents. Turns are constructed by turn-construction units, which can be words, phrases, clauses, sentences, paragraphs etc. At the end of a turn there is a TRP. Prosody, facial expression and syntactic and semantic knowledge belong to the aspects which indicate to us whether we have reached a TRP.

In an interactive narrative, agents would need to communicate their goals and beliefs in some way to know whether a TRP takes place and whether it is fit for someone to take the turn. Joint activities describe exactly the mechanism which we could use for this issue.

Overlap in turn-taking is an element of realistic, and perhaps believable, speech exchange systems. Overlap can happen for several reasons. For example, one could interrupt the current speaker for some reason or give verification to the current speaker like "uh-huh". Tannen has come with three causes for overlap which could be seen as "not interrupting" [36]:

1. Cooperative sentence-building: the listener engages in what he or she perceives to be cooperative sentence-building, when the other does not seek help.

2. Requesting and giving verification: one of the participants requests verification during the other's turn (This is the uh-huh-case).

3. Choral repetition - This occurs when a listener anticipates and says what the other says.

2.2.5 Transitions in behaviour

We briefly described before that two utterances can be seemingly unrelated if there is no "glue" binding them together. Halliday and Hasan [37] already worked on the difference between a text and random unrelated sentences. They proposed a set of cohesive devices: reference, substitution, repetition, ellipsis, and conjunction. They indicate semantic relations between an otherwise incomprehensible sequence of sentences. Part of their cohesive devices (and, but, because, or etc..) are nowadays called discourse

markers. There is no consensus on the exact set of discourse markers, however with help of research done by several [38–40], we think we can conclude some essential discourse markers which are sufficient to glue our action expressions together. Or as Louwerse and Mitchell [40] state:

> *"Whether discourse markers are defined as devices for marking transition points in discourse [39], as devices cueing the hearer to a change in discourse structure [38], or as devices marking movement between two discourse units [41], they are the conversational glue that participants effectively use to hold the dialog together at different communicative levels."*

Schriffin is one of the first to mention the multidimensionality of discourse markers [39]. She mentions:

- The ideational structure, with relations between propositions (cohesion relation, topic relation or functional relation).
- The action structure, which describes the organisation and constraints on the use of speech acts.
- The exchange structure, which is 'the outcome of decision procedures by which speakers alternate sequential roles and define those alternations in relation to each other'.

Duly noted that one discourse marker in almost every case functions within more than one dimension. 'And' functions as turn-keeping element and conjunction in about every case [42].

The approach to quite arbitrarily choose some discourse markers which are 'sufficient' for some domain has been employed by Bickmore & Cassell [43]. They used discourse markers for their agent, which often switched between small talk and task relevant talk.

> *"Discourse markers include "so" on the first small talk to task talk transition, "anyway" on resumption of task talk from small talk, and "you know" on transition to small talk from task talk."*

This shows that even some crude transition mechanism based on discourse markers could improve and increase the interpretive affordances offered.

Aijmer suggests that there are few aspects of any language which reflect the culture of a given speech community better than its particles. They are ubiquitous and idiosyncratic [44]. This would imply that any modularity we would find for transitions in our demonstration are not generalizable when we translate the game to another language. Groen et al. [45] did their research on the effect of substituting discourse markers on their role in dialogue. Substituting discourse markers in their case meant that they translated English discourse markers to Dutch discourse markers and checked if the translation had an equivalent semantic meaning. The results indicate that there is no mechanism which gives the proper discourse marker in any language. Or as they quote:

> *"The results show that substitution has a differential effect on the localization and assessment of coherence and dialogue goals. Based on these results, it is recommended that care needs to be taken when substituting discourse markers because the functional relation between the marker and the marked stretch of dialogue could be compromised."* [45]

There are many discourse markers which indicate how an utterance is to be interpreted. For example "I think" means that the speaker does not take full responsibility for what he is about to say next. However, we are looking for those discourse markers that glue two otherwise independent utterances together. Even though most research is about finding meaning through discourse markers, we can reverse their results to create sensible flow in our dialogues. If discourse markers are categorised, the intentions, or plans, of our agent should be categorised the same way. If 'exactly' is an indicator for positive continuation on a previously made statement, acknowledging the interlocutors statement fully, and showing empathy towards the interlocutor, then 'exactly' functions as befitting glue for two intentions which belong to the exact same category.

The categorisation of Louwerse and Mitchell [40] constitutes the following discourse markers:

- **Exactly** shows positive continuity, acknowledgement and emphatics
- **Mhm** shows positive continuity, acknowledgement and no emphatics
- **Anyway/ but then again** shows positive continuity, partial acknowledgement and emphatics
- **Oh okay** shows positive continuity, partial acknowledgement and no emphatics
- **Kind of** shows neutral continuity, acknowledgement and emphatics
- **I guess** shows neutral continuity, acknowledgement and no emphatics
- **but then** shows neutral continuity, partial acknowledgement and emphatics
- **aw/ whoa** shows negative continuity, acknowledgement and emphatics
- **oh gosh** shows negative continuity, acknowledgement and no emphatics

Erkens & Janssen [46] used discourse markers to create an automated coding procedure for dialogue acts. Their exact categorisation can be found in their article. For the remainder of this research we took the catigorisation of Louwerse & Mitchell in our minds [40].

2.2.6 Summary

We summarize our desired properties:

- The architecture should be based on selecting actions which best express that which the audience should know, not on how to select a specific action which is optimal for the agent. This will often mean giving up generality and modularity [47].
- Agent needs to act believable and sociable
- For an agent to act believable it needs a strong personality
- For an agent to have a strong personality it needs to act emotional, social and expressive
- For an agent to act believably sociable it needs to act as if it is aware of the timing of its actions for this we need:
  1. The agent to act as if it is aware of social norms and values
  2. The agent to act as if it is aware of its physical, albeit conceptual, environment
  3. The agent to use some mechanism which takes into account past actions and events
- For an agent to act believable, emotions need to be expressed broadly
- A sociable agent should be able to engage in joint activities
- A social agent needs to be able to have several behaviours run in parallel
- It should be able to somehow express its change in behaviour

2.3 Methodologies

There are several approaches to creating an intelligent agent. Muller and Kamermans have already attempted to use the BDI approach to create an intelligent agent for interactive narratives [1, 16]. As this research progresses on their work we will describe this approach and mention how its merits and detriments are related to creating social and believable agents.

2.3.1 A Belief, Desire, Intention paradigm

The BDI approach originates from Bratman's folk psychology [11]. Folk psychology describes the naive way in which people tend to explain their own reasoning process and that of others. BDI is an abbreviation for beliefs, desires and intentions. As such, a BDI agent is characterized by its beliefs, desires and intentions. Its beliefs stand for what he believes is true in the world e.g. boy(Roy) and get updated each sense-action with belief update rules which, in 2APL, are coded like [48]:

– {oldstate} event {newstate}

– {boy(X)} percept(X = Roy) {boy(Roy)}

– {boy(at(x,y))} moveRight(boy) {boy(at(x+1,y))}

Desires, or goals, stand for the motivational state of the agent; everything the agent wishes to be true. For every desire it checks if there is a possibility to fulfill this desire through some set of actions. Intentions are actively pursued desires. An intention has a set of plans and for every intention the agent checks whether the plans are feasible. The intention of which plans are feasible given current context shall be actively pursued. These plans consist of a list of actions and these actions represent the agent's visible behaviour.

Because of its foundation in folk psychology it should not come as a surprise that this paradigm is quite fit for creating synthetic characters with human-like characteristics. Some merits are:

- They act proactive based on their desires, which makes them predictable in the long run
- They are defined using folk psychology, which makes it relatively easy to describe their reasoning to everyone, not only to programmers.
- They can adapt to their environment as long as their ontology covers it
- It does not enforce realism or intelligence
- It does not exclude focus on action-expression
- Easy to add norms and values to belief base
- Easy to store facts from environment over a series of sense-actions. This means it can keep track of time because it keeps a belief base
- It could create models of other agents by adding facts in the belief base [49]
- It could be used for creating a GUI agent which enables turn-taking [1]. However, we do not necessarily need a GUI agent as turn-taking could be implemented within the acting agent
- Transitions, or changes in behaviour, could be explained [50]

There are circumstances where the paradigm is overly complex for what is required. Sometimes, low level subconscious behaviours cannot be defined as beliefs, desires or intentions but they still have a strong influence on the overall behaviour of the character. However, subconscious behaviours can be implemented as plug-ins. These behaviours are not directly part of the reasoning process, but influence it because the character must adjust for the timing and errors that occur in the behaviours [51]. In other circumstances

the level of abstraction is too high, and a more detailed model is appropriate [51]. For example, where there are additional high-level conscious processes that have a significant influence, like learning or emotions or social modelling [51]. Nevertheless, the high-level conscious processes can also be described by folk psychology and could be added to the BDI framework. This facilitates a new framework, based on BDI, which incorporates the characteristic of interest. Marsella et al. have addressed the issue of emotion, as have Dastani and Meyer [52, 53]. Learning in BDI has also been addressed by several [54–56]. Additionally, while it is possible and easy to store facts about other agents, a BDI architecture conceptualises individual intentionality and behaviour. It says nothing about the social aspects of agents being situated in a multi-agent system. It is not hard to imagine how BDI could be extended for multiple agents with social models and agents using Theory of Mind. Work on the latter has been done and has led to promising results [49, 57].

Another difficulty is, given that an intention is adopted, how committed should the agent be to this intention? We do not want an agent to give up too quickly because sometimes certain goals can not be achieved with a continuous line of progress. However, we also don't want to let the agent be committed to its goals into infinity. Kinny and Georgeff have worked on this problem already in the early days of BDI [58]. An example of work done on this subject has led to P-goals (persistent goals). These goals are not dropped until deemed impossible or if fulfilled [59].

Also often mentioned, is the issue which is the gap between theory and practice. Early AI focused on being the computational study of rational behaviour. The goal was to align theory and practice. However, perfect rationality for instance does not exist in practice by definition. To select the perfect action computation time is required. As the environment will have changed during computation the action is never perfect unless the environment has not changed; static environments are not a point of interest for AI research. However, there are several programming languages constructed based on the BDI paradigm to fill this gap: AgentSpeak(L) [12], METATEM [60], Congolog [61], 3APL [62], 2APL [48]and JADEX [63].

At Ranj, previous work has been done on integrating JADEX with ActionScript® 3 with the purpose of integrating a BDI framework in their OZ Story Editor [15]. It is a simplified version of JADEX in which only functions that were crucial for creating the Glengarry Glen Ross game [1] were integrated [15]. Lastly, BDI does not tell us how agents should behave to be social and believable. This is why we first defined socialness and believability. We see now that, while BDI says nothing about this, it is possible to extend the framework with these properties.

An overview of the simplified version of JADEX integrated into ActionScript®3 can be found in Appendix A.

2.3.2 Language processing and generation

An agent which is capable of generating natural language and understanding natural language the same way, or better than we do, would obviously be nice. However, we argue that this (GOFAI) goal, for now is not achievable. At the time Alan Turing intro-

duced the Turing test [64], most thought language would be a relatively easy problem to solve compared to things like playing chess. However, to this date there is no system that is capable of understanding and generating language at the same level as we do. TNO - the Netherlands Organisation for Applied Scientific Research - together with Utrecht University and RANJ created The Glengarry Glen Ross game. They used a BDI-framework with language generation through using standard templates in which you had to fill in the variable words to create a fully autonomous agent [1]. As many other language generation applications however, this one seems to fall short on nuances.

> *"dialogues lacked conversational nuance, due to the fact that expressive aspects were not included in the models. In its present form, the approach is better suited for learning conversational strategies than for learning conversational nuance, such as training bad news conversations or training to deal with social pressure."* [1]

Not only does it lack certain dynamics we normally have in our use of language, it is also creating weird sentences which often are grammatically and semantically correct but have an unnatural feel about them. Members of the Oz-project have also worked on a broad agent architecture, which includes language processing and generation, named Tok [65]. Hap [66] was the reactive component and ABL [33] was generated from Hap specially designed for interactive agents with extra capabilities like joint goals and joint behaviours. This system has been used in their game Façade [67]. A complex and well-thought system functions behind the agents in this game. Nevertheless, the agents lack interpretive affordances and as such the player does not know what to say to the agents. If the player says something not within the domain of the agent the agent will simply not understand what is said or reply with some seemingly random utterance way out of context.

2.3.3 Combining scripted dialogues and BDI

For now, it appears that an autonomous agent with language generation is not always viable solution for creating social and believable characters, and scripting is a weak solution for very complex storylines. We argue that, given that we have a finite domain, we could make an agent which autonomously decides what kind of action he wants to take and fill in the nuances of that action with scripting. Several have used similar techniques for creating dialogue-based agents.

The use of templates to create autonomous agents resulted in poor nuance [1]. Oijen et al. [68] used scripts by connecting them to certain actions. However, their dialogues ran in an ongoing physical environment and focused on npc-npc interaction as well as player-npc interaction. They simulated a fire on a boat and your goal was to distinguish the fire. The player had to command agents and agents had to work together to perform the acts commanded. This meant that the agents had to communicate to each other. Sometimes having several dialogues running in parallel. Since agents could have several behaviours run in parallel, another agent can not simply assume that the intended interlocutor has their attention at any time. Commencing and termination rules were

required to ensure a valid open communication channel with the interlocutor [68]. In our case however, commencing and terminating is covered within the scripts. Moreover, they focused on functional dialogues, not poetic dialogues. Below we present an example of a dialogue in their project.

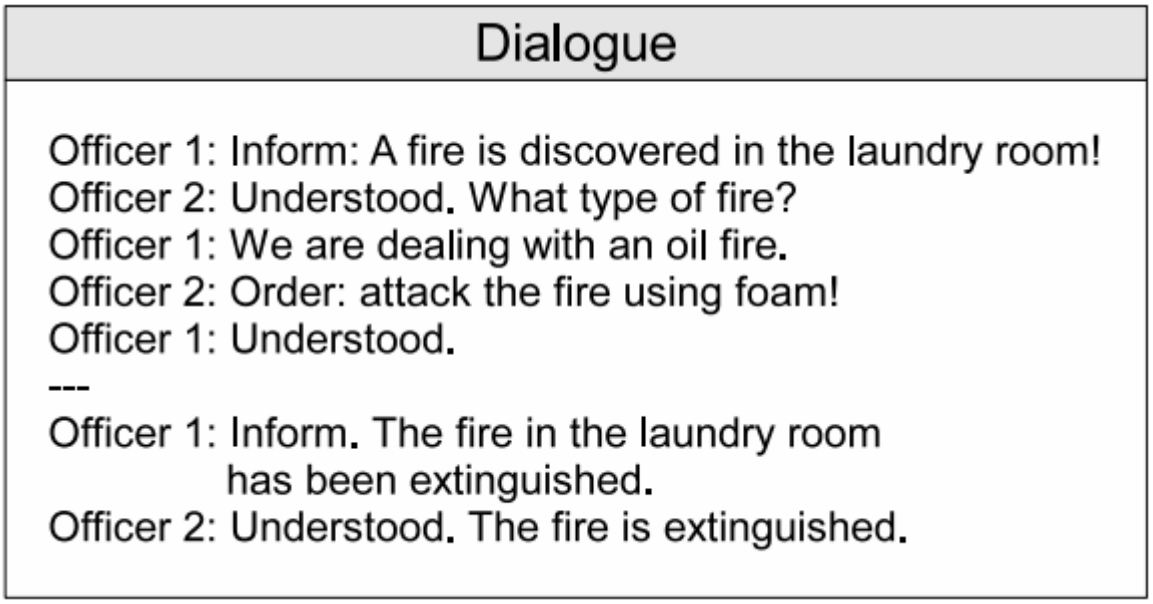

Figure 4: An example of a dialogue between two agents

So, they focused on clear communication between agents and players with low-level actions, such as informAboutFireAt(location). Instead, we focus on being able to be able to perform higher-level actions, such as askAboutVillageLife and have a complete conversation about the life of the NPC in its village. Obviously, we encounter different issues. We have to think about when interaction is relevant, whereas their low-level actions offer interactivity throughout the dialogue. So, even though both we and Oijen at al. intend to combine scripted dialogues and BDI, we encounter different issues as a result of different communicative and educational goals.

3 Method

We propose to combine scripting with BDI to get a new trade-off between interactivity and expressive control. Some have used similar techniques for creating dialogue-based agents [1, 68], but not with our combination of design and purpose. Our contribution here is to enable believable flow and nuanced dialogues in player-NPC-interaction. We think we can accomplish this because we take actions to a higher level than is normally done with language generation and express actions in their entirety. As far as we know this has not yet been achieved in interactive narratives.

We argue that it is not something trivial to combine BDI and scripting. Which parts of the game are covered by the BDI and which parts are covered with scripting? How does the BDI "communicate" with scripts? What kind of arguments are to be passed on to a script? Can scripts be categorised? What aspects influence the size of actions expressed by one script? Can we pinpoint rules of thumb for creating games with our approach. Can we reuse scripts for different agents and games? Et cetera. These are all questions that did not get an answer in previous work done by Kamermans [16] and van den Bogaard [15].

We restate our research questions:

- RQ 1: Can we gain an understanding of the process of creating a game with our approach of BDI and scripting?

- RQ 2: Can we retain merits of BDI, like reusability, scalability and proactiveness, when combining it with the narrative control of scripted dialogues?

- RQ 3: How should the size, or level-height, of an action be decided?

Next, we describe our method to find answers to the research questions.

3.1 Designing the initial architecture

The best way to get an understanding of the process of creating a game would be by creating one [69]. Even though building a full game falls out of the scope of this research, we still try to take a first step in doing so. Before we are able to create a game with our approach of combining BDI and scripting, we at least need any idea of what kind of system should be built. Thus, we played dialogue-based games at &ranj and analysed them in order to get an understanding of the structure of dialogues.[1] How does the game translate to BDI and scripting the actions of a BDI are at sentence level? How do they translate to BDI if the actions are at a higher level (whole dialogues)? Additionally,

[1] Game titles of &ranj Games: Engagement Game; DAF sales game; Internal Investigation Game; Heartware

we designed our own conceptual games, and analysed them with the same intention. This way we could think of concepts we thought were more suitable to create with our approach.

3.1.1 Analysing existing games

As stated in section 1.3, the game at &ranj are often games in which the player has to converse with NPC's. In the Engagement game (see footnote 5), the player has to learn how to cope with issues a manager of restaurant encounters. In the Internal investigation game, the player has to learn what steps should be taken to run a successful lawsuit. Here it is important that you learn who to talk to at what moment, what should be said and what should not, and whether something counts as evidence or not. The games all had almost linear dialogue structures. However, sometimes consequential choices were to be made. For example, in the dialogue shown in figure 3, the player has the choice to give either everyone or just one employee a bonus. So, we have some action discussBonusses, which opens the script discussBonusses. If the player chooses to give everyone a bonus, we want to store this in the agent (who for the sake of this example plays all NPC's). So, during the script a variable is set. After the script is closed, the variable should be processed to change a corresponding set of beliefs in the agent. This gives us a general idea of the process of the agent.

3.1.2 Translating game concepts to an agent and scripts

Secondly, we generated our own conceptual games.[1]

(1). A game where the player can talk to a mercenary in a Oblivion [2] type of world.[2] Rationally, your goal would be to convince him to join you in your cause for free, or perhaps, for a small fee. However, using different tactics could result in a fight to the death, different fees, or perhaps a new friend. In the process of translating this story to a BDI with scripting. We tried to figure out how a story translates to a BDI. This showed that the execution of an action is always very dependent of previous actions and events, since they could change the context.

(2). We created a second game where the player is a detective. His objective is to go to a village where it is believed drugs are produced and smuggled. In this village he should approach villagers and try to convince them to talk about the drug production as witness for the case. In Appendix B, a simplified version of the story can be found. Again, the influence context had on the scripts expressing the actions became apparent. Moreover, a second issue became apparent. Two subsequent scripts needed different transitions based on what was said in the first script. Also, we had to think carefully about the size of actions. If one of the actions of the agent was to mislead the detective about the whereabouts of the location and the player figures out the agent's intentions

[1] The reader can contact the author for files created. However, as they are quite sketchy, we decided not to include them in the appendix.

[2] In case the reader is not familiar with the game, check a description at : http://en.wikipedia.org/wiki/The_Elder_Scrolls_IV:_Oblivion

halfway the misleading action, we might want to offer the player the possibility to redirect the conversation. If the latter is opted for, then we need to think if we can cover this redirection within the same script or if we want to stop the script at a certain point and increase interaction by offering a whole new set of actions to the player. These kind of choices for the artists can be illustrated by design patterns, as part of the architecture.

3.1.3 Considering the task-division and process of game-development

Lastly, we considered what task-division would be most suitable for creating a game with our approach of combining BDI and scripting. For this we looked at the task-division currently present at &ranj. They stand by segregating tasks of programmers, game designers, and visual designers. With respect to the scope of this research, we looked at the tasks of programmers and game designers respectively, and we tried so sustain this task-segregation.

3.2 Evaluation of the initial design

We designed the initial architecture based on our experience we achieved through analysing the games and translating game concepts to an agent and scripts. This architecture can be found in section 4.3. We paper-prototyped the script building blocks presented in our initial design. We wrote a simplified version of game (2) from section 3.1.2. The general concept of the game, agents BDI and plans, and the filled in script building blocks can be found in Appendix B. Two experienced game designers at &ranj got a short presentation, separately, on the current research. After which we asked them to use filled in script building blocks to create scripts. After completion of the exercise, we conducted an interview. We aimed to pick up their thoughts on the whole architecture in general and, most importantly, the script building block. Did we use it correctly? Is the information given sufficient? Et cetera. In section 4.4 we clarify the adjustments made based on the perspective of the game designers. This type of evaluation bears no deductive proof, but it does bear an indication of the validity of our architecture.

4 Design

In this chapter we lay out the processes, design patterns, and building blocks that shape our architecture.

4.1 An overview

Creating a game with our combination of BDI and scripting is by all means not trivial and by offering a building block and design patterns we hope to provide some modularity. This modularity, in turn, allows us to create more complex games.

In section 4.2 we describe certain design-choices made. The design-choices made, follow from the process and task-division that takes place at the game development at &ranj. These design-choices bear restrictions that shape our architecture.

In section 4.3 we illustrate a process model describing the iterative process of the action-selection mechanism (the "agent-side" of the process) followed by the action-expression mechanism (the "script-side" of the process) and then back to the action-selection, et cetera. We start with a naive model in 4.3.1 and expand this in later subsections 4.3.4 and 4.3.5.

In section 4.3.2 we present the script building block. This building block was designed to be the medium for programmers to communicate with game designers. Programmers could fill in the building block and game designers would have enough information to write a script without any extra information.

In section 4.3.3 we describe design patterns, which illustrate some of the authorship the scriptwriters have. It is important to understand that we are not limited to these design patterns in creating scripts.

In section 4.4, we discuss the adjustments made based on the results of the interviews.

4.2 Design-choices and restrictions

If we build a game with a combination of BDI and scripting, what are our design choices? The creation of a game demands cooperation between programmers, game designers and visual designers. Our research covers both the task of the programmer and the game designer. Programmers generally are competent in creating action-selection mechanisms, while game designers generally are not. Additionally, game designers generally are competent in implementing the action-expression part of the agent, while programmers are generally not. So, we would like to sustain this task-segregation. The programmer's task is to fully design the action-selection mechanism such that the desired intentions are selected. These intentions open plans, which consist of actions. The game designers then have to write scripts which express the intentions behind the actions.

We give a crude idea of the task-division per step in the game development process [1]:

1. General concept of the game; game designers + programmers
2. General set up of BDI; game designers + programmers
3. Implementation BDI; programmers
   (a) Construct or adjust BDI reasoning engine
   (b) Set up of beliefs, desires, and intentions
   (c) Set up plans for intentions
   (d) Design of plans: defining the actions
4. Expressing the actions through scripts; game designers

This strict segregation between action-selection and action-expression bears restrictions, which then again influence certain design choices.

One of the restrictions is that we are no longer able to use and update beliefs for reasoning within the script. As updating and processing the agent's state within a script would imply giving the game designer the task of the programmer.

A second restriction is that we can not simply stop scripts halfway. An action as a concept is connected to a script. A half script means that the concept may not actually be expressed, and we would have to create some way to cope with interrupted actions. The latter is obviously possible, but being able to cope with all those half actions, we would quickly fall back to be able to cope with really low-level actions. We then lose one of the benefits of scripted dialogues: expressive control. Some crude interruption-mechanism could be implemented, where the agent simply decides that an action is not performed at all if interrupted. In games where a turn-taking mechanism is present, this crude mechanism could be opted for. This however, falls out of scope for this research. These two restrictions force us to think carefully about the size of actions.

4.3 Initial Design

First, we present a naive model. This model takes the strict segregation into account. However, it does not give us the opportunity to create context-dependent actions. Nevertheless, it will give us to opportunity to present some building blocks for scripts and design patterns for script authoring. The strict segregation poses us with a choice to make after a player has made a choice within a script. Does the choice influence the state of the agent? Does the choice influence the direct continuation of the dialogue? The answers to these questions correspond to design choices that have to be made. After this we propose a solution to the issue of highly context-dependent actions. Concepts of which the expression could be very context-dependent, would quickly result in a combinatorial explosion of different actions that all execute the same intention. So we create one plan which, based on current beliefs, executes an action with context-relevant beliefs

[1] It should become more clear during this section what these tasks are

as its arguments. These arguments can then be used in the script to release variable content corresponding to those arguments. We then propose two ways to cope with variable content within a script. This means that we give an idea of how the agent's action-selection part can communicate to the script, its action-expression part, which variable content should be activated. This shall be discussed in section 4.3.4. We then encounter another issue which we have already introduced. The issue of transitions between two different scripts. Two subsequent scripts, can sometimes lack meaning or be ambiguous if there is no cohesive device describing the semantic relation between the two scripts [37]. We insert a transition-mechanism between the action-selection and script-selection. As noted before, we elaborate on this in section 4.3.5.

4.3.1 The naive agent-script model

This model is the most simple case of a process from action-selection to action-expression and back to action-selection (see figure 5). In the first step of our process model, a plan gets selected by the BDI Reasoning engine, which as explained before is an implementation of JADEX to ActionScript3. The next step is the plan execution. An action is selected, which opens at least one script. The specific implementation of the expression (the script) is then filled in by the game designer. During the play of script, the execution of the player will influence the variables set; indirectly influencing the belief base of the agent. And the cycle starts again.

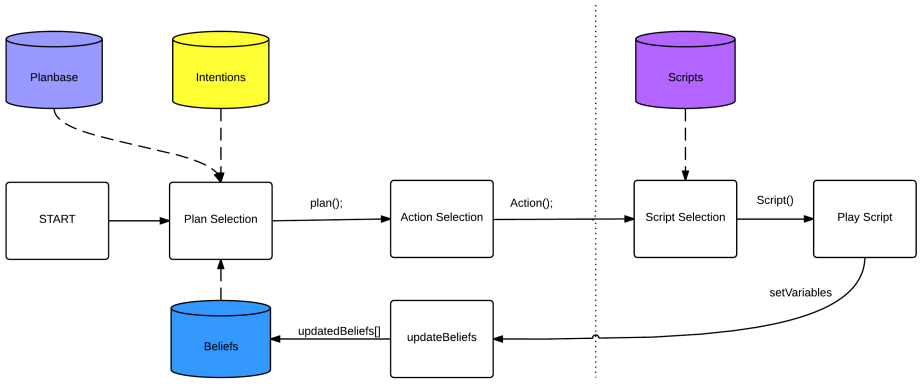

Figure 5: A plan is selected by the agent. It is chosen from the plan base because the conditions, which are beliefs, of a corresponding intention hold. The plan owns a function which selects an action. The agent has now chosen to take action based on its intention. From this point on comes the action-expression part. A database of scripts, for every action at least one, is available by the agent and finally one of the scripts, authored with the OZ Dialog Editor, is executed with the OZ Dialog Player. During the play of a script a list of variables that are set, based on the choices the player has made during the script, is stored. When the script is finished, this set is then sent back to the reasoning part of the agent. The set variables are converted to corresponding beliefs and based on its updated belief base the agent selects a new plan.

4.3.2 Script building block

Decent communication between programmers and game designers is vital for the success of creating a game in this architecture. The programmer and the game designer have to communicate what is crucial within the script to express the action.[1] Remember, what affords interpretivity? Therefore, we design a building block which is sent from the programmers to the game designers (see figure 6). Every known element is filled in and given by the programmers. They indicate who starts the script, who ends the script, what the context is for the transition between to scripts and other conditional content, what message the audience should get (see Appendix B for a filled in example).

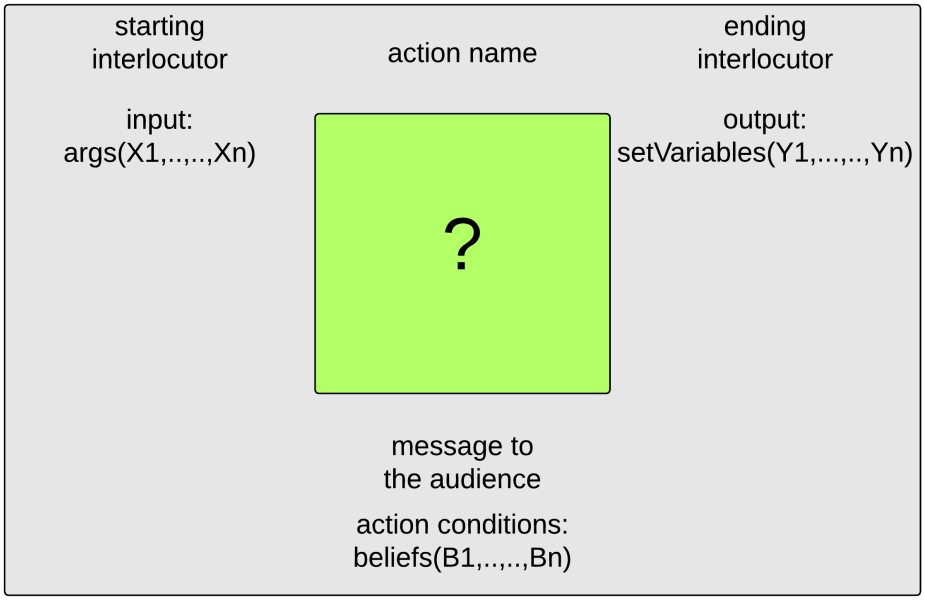

Figure 6: The starting/ending interlocutor is the character initiating/ending the conversation. The input arguments are the arguments indicating the conditional content, the first argument represents the semantic relation between the previously and currently chosen actions, each other argument is a context-relevant belief. The message to the audience is what part of the agent's intentions should be perceivable by the audience. The action conditions are the beliefs that triggered the agent to execute the action; these were supposed to aid the game designer in understanding the relation between the agent and the game. The variables that are set during the script in our case represent the agency of a player during a script. The choices a player makes influence the narrative path that is followed. Different paths during a script trigger different variables to be set which in the end change the state of the agent. The centered question mark box refers to the script

[1] This is NOT the final idea of the script building block, this was what we initially thought its function would be

4.3.3 Possible state structures

One of the main restrictions that arise from the strict segregation is the inability to request, update, and process beliefs during a script, as noted earlier in the introduction section. Once a script is run we can not send or retrieve information to or from the agent until the script is closed. This immediately gives rise to some possible structures (design choices for the game designer). Other structures are obviously allowed, these are just to give some insight on the way scripts can be constructed. [1]

When a reply by the player is given, the world for the agent has changed. Since we can not at that moment update and process beliefs for the conversation to continue, we can think of three different structures:

1. The choice the player has made is essentially meaningless to the agent. We call this a non-consequential choice (NCC)

2. The choice the player has made does have an influence on the state of the agent. However, it does not influence the continuation of the current script. We call this a consequential choice (CC)

3. The choice of the player instantly influences the continuation of the dialogue. This means that the script has to be cut off, and beliefs must be updated. The new state of the agent will trigger a new intention. We call this the consequential end-choice (CEC)

See figure 7 for an illustration on the possible states.

Non-consequential choice

So within a script we have three different possible states. Each state represents a different level of control and interactivity. Examples of when to choose for an NCC could be (see figure 8 for an example):

1. An introduction to the game mechanics

2. In serious games, occasionally some kind of evaluation takes place after the game has been played. In this case a piece of script, which had no influence at all on the game, can introduce a subject to discuss in the evaluation.

3. Apparent interaction: the player gets options which may lead to a different storyline. However, the agent's state does not get influenced by the choice the player makes. It simply leads to a different experience.

[1] Such as a recovery-structure. E.g. In state 1 a choice can be made, setting a variable based on the choice. However, later in the script, again a choice can be made, negating the variable set earlier in the script. So initially, it looks like a CC, but as you can recover, it was not necessarily consequential.

Consequential choice

In the CC some variable value is set. Again, since we do not have access to the agent's beliefs during a script we have to decide whether a player's choice instantly needs to be processed by the agent for the narrative to continue in a sensible manner, or whether it can wait. In the latter case we temporarily store the choice in a list of variables to be sent to the agent once the script is finished.

An example for this case can be found in a questionnaire. After every question an answer is given and every different answer sets different values to variables. The answers however do not have an influence on what question comes next. So, the intention can continue within the same script. Or you could ask how someone is feeling and the information given is used directly in the continuation of the dialogue. However, the game designer feels that the amount of options is overseeable as such it could be his or her design choice to include every possible continuation in the script instead of cutting it off (see figure 9 for an example).

Consequential end-choice

A CEC is essentially the same as a CC. The only difference is that after the CEC, the script is finished. If continuation of the story at a certain moment would branch in a lot of storylines, this is obviously a case where BDI has its advantage. A perfect example could be the agent asking the player what he wants to talk about (see figure 10 for an example).

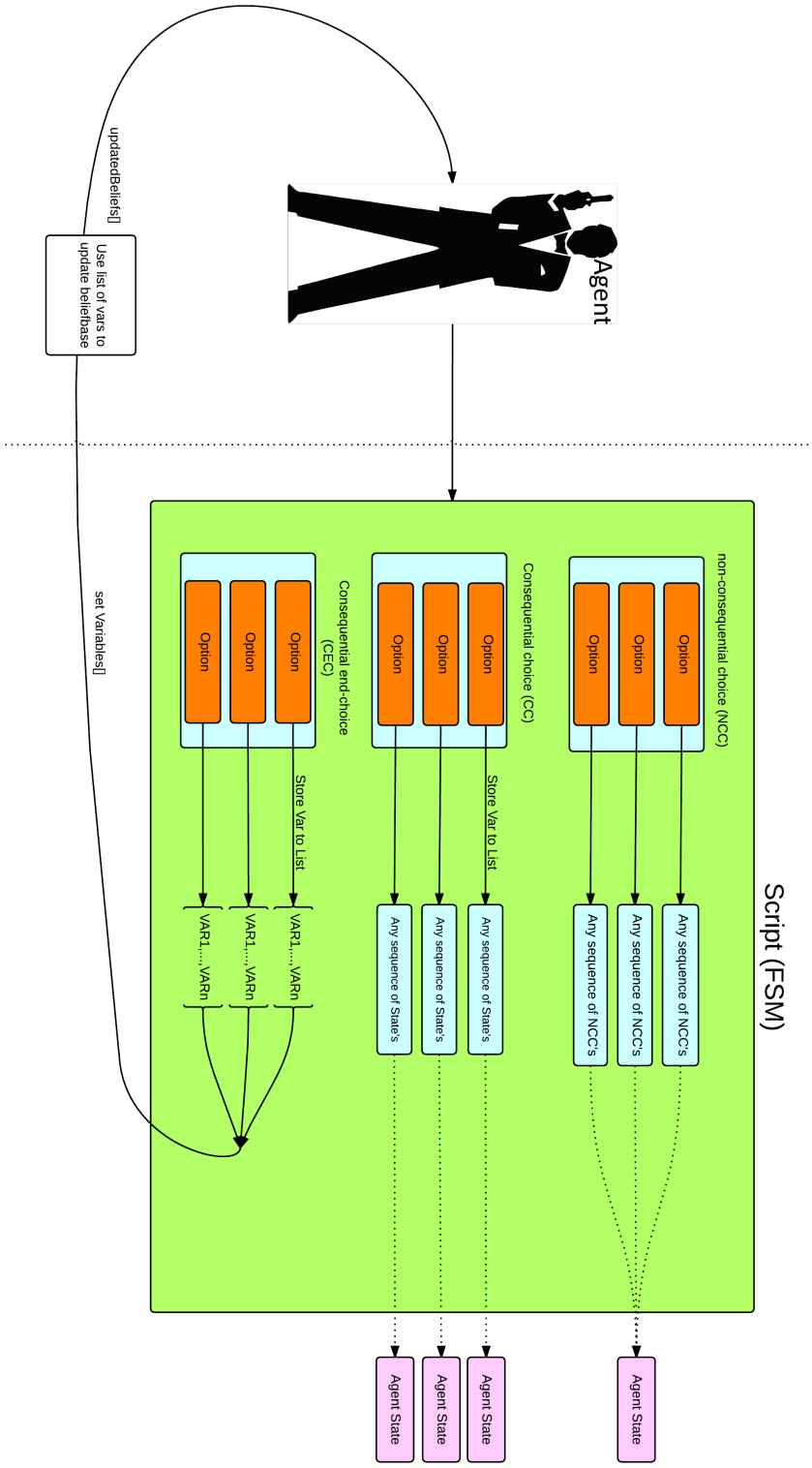

Figure 7: A triplet of NCC's will lead to the same agent-state no matter which choice was made. A triplet of CC's will lead to different agent-states, based on the choice made. However, the script continues. A triplet of CEC's lead to a different agent-state, and immediately end the script, such that the agent can process the choice(s) of the player made during the script internally, to decide the continuation of the dialogue.

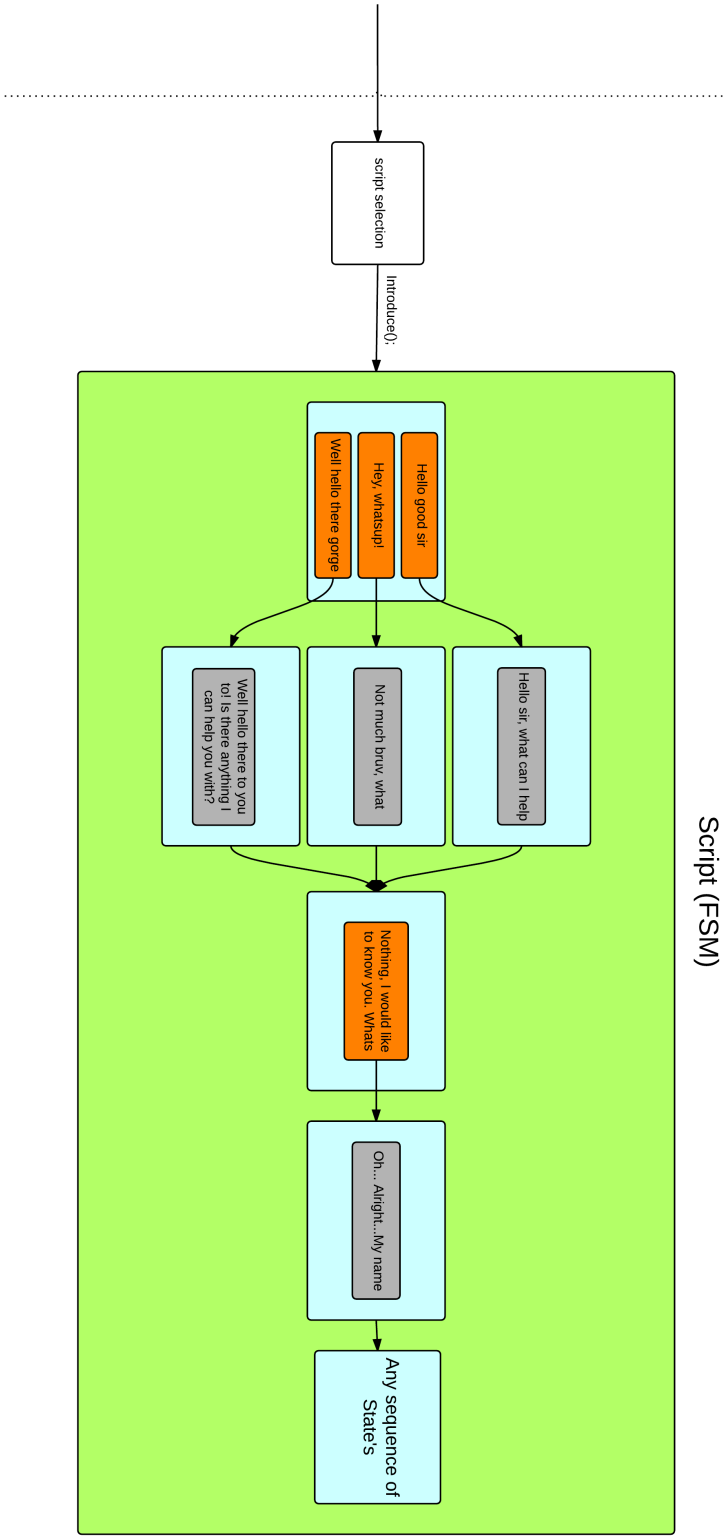

Figure 8: The agent has selected the action introduce() which opens the script introduce(). This is a script which starts with a state where the player can choose its "style" to introduce. While the reaction of the agent will depend on the chosen style of the player, internally the agent does not change his state. Furthermore, the script continues the same, no matter which choice the player has made.

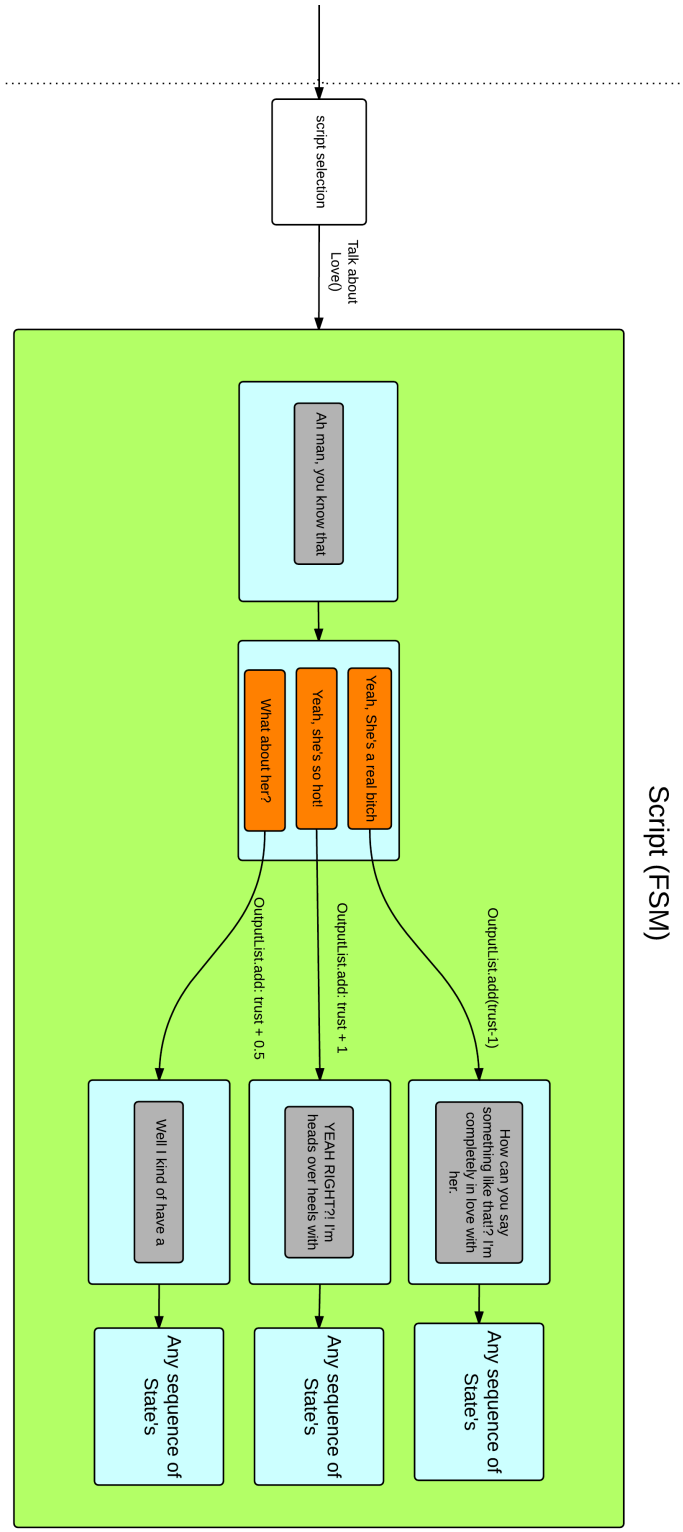

Figure 9: The agent has selected the action talkAboutLove. Apparently trust has exceeded some threshold and the agent starts to talk about personal stuff to the player. The second state in the script poses three options to the player. Each option results in a different update of the level of trust towards the player. However, for now the choice made at this moment is irrelevant for the continuation of the action. As such we can continue with the script and temporarily store the chosen value until an end-state has been finished.

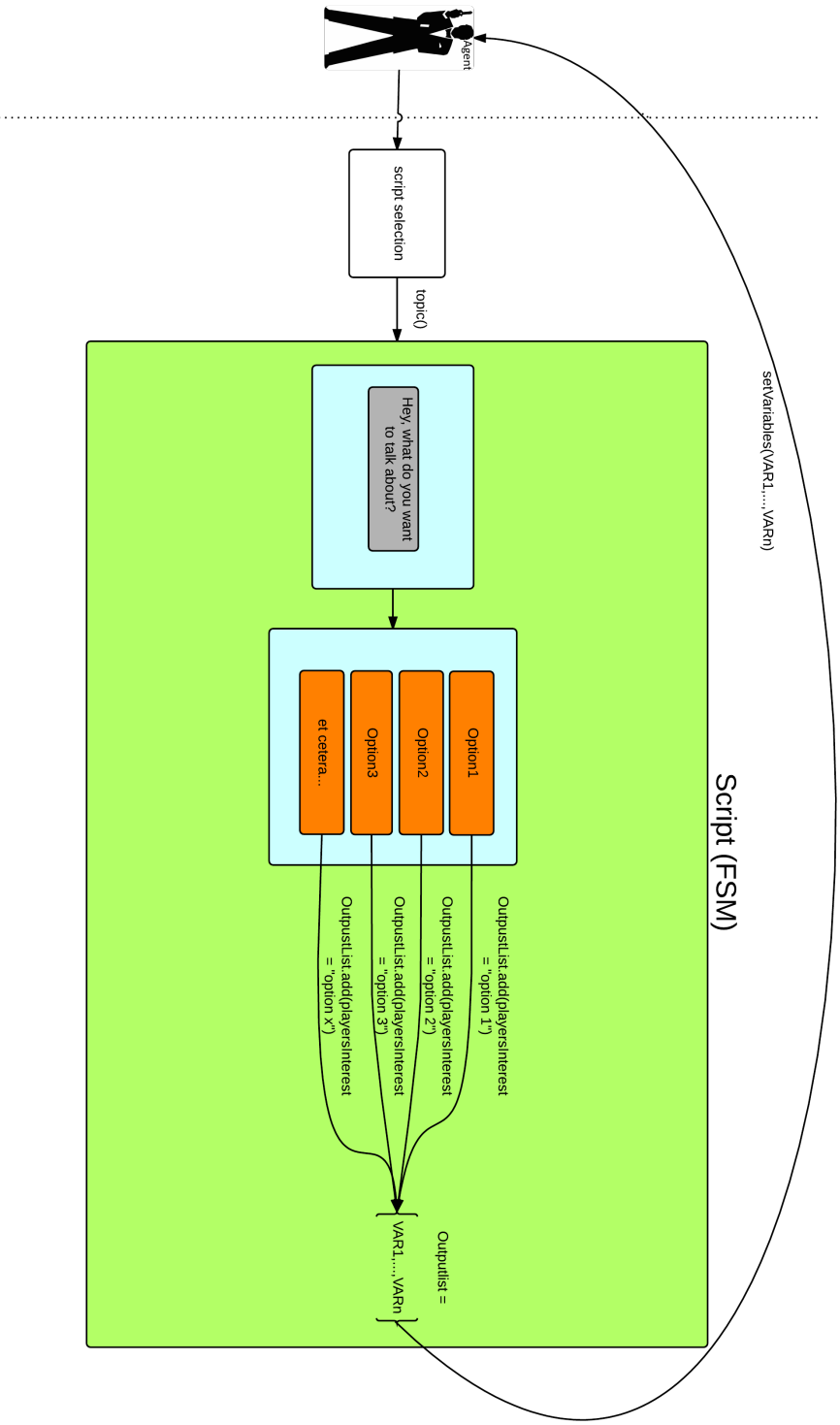

Figure 10: The agent has selected the action topic, which includes asking the player what he wants to talk about. Now, considering that there are quite a few options it is more efficient to send the players choice to the agent and then execute a new plan, which addresses the topic chosen. The list of variables - in this case just the topic option - is sent to a belief update function, after which the beliefbase of the agent is updated and a new plan will be selected.

4.3.4 Context

Some actions derive from the same intention or plan but differ in execution. As an example we take one from our exercise in Appendix B. The agent has an intention to give the player the option to talk about life in his village. However, possibly, other topics have been discussed which influence the execution of the dialogue. In the example we take from Appendix B, there could have been conversation before about the player's occupation. In this conversation the player tries to get the NPC to talk about the whereabouts of drugs. It can do this by either presenting himself as a junky, or by presenting himself as a detective. So, we already have three different contexts for the conversation about life in the village:

1. The agent has no beliefs about the player
2. The agent believes the player is a detective
3. The agent believes the player is a junky

Now, we could write a separate action with a separate script for every action. This is how it would be done in the naive model. Moreover, an action does not necessarily have just one belief defining the context. For example, in the same script about the player trying to find the drugs by telling his occupation, the player can offer the NPC protection if he went for the detective role. This protection might reduce the agent's fear, which causes him to talk more freely. Obviously, for every possible belief that belongs to the context of an action the amount of possible contexts grows exponentially. Given that there will be a lot of intentions the agent could have, we would easily get an explosion of actions. So, we can think of a more generalised design with the new features in red (figure 11). The most essential difference with the previous model is that in this model the action selection function of a plan uses beliefs as an input. The beliefs are passed on to a script as context, which in turn can trigger conditional content within a script.

This is both easier to understand as to implement. However, processing the arguments in a script is not trivial. We will show this together with an example. Suppose we have three beliefs which belong to the context talkAboutVillageLife:

1. Player is a detective
2. Player is a junky
3. Trust player because he offers protection

So we get the plan talkAboutVillageLife(). At action selection we check for all the contextually crucial beliefs. Every belief becomes an argument of the action. So, talkAboutVillageLife(belief1,belief2,....,beliefn) in our case becomes talkAboutVillageLife(Detective,Junky,Protection). Suppose in our example we only have belief(playerJunky). So what is sent to the script now is talkAboutVillageLife(false,true,false).

We propose two solutions to cope with variables in scripts.

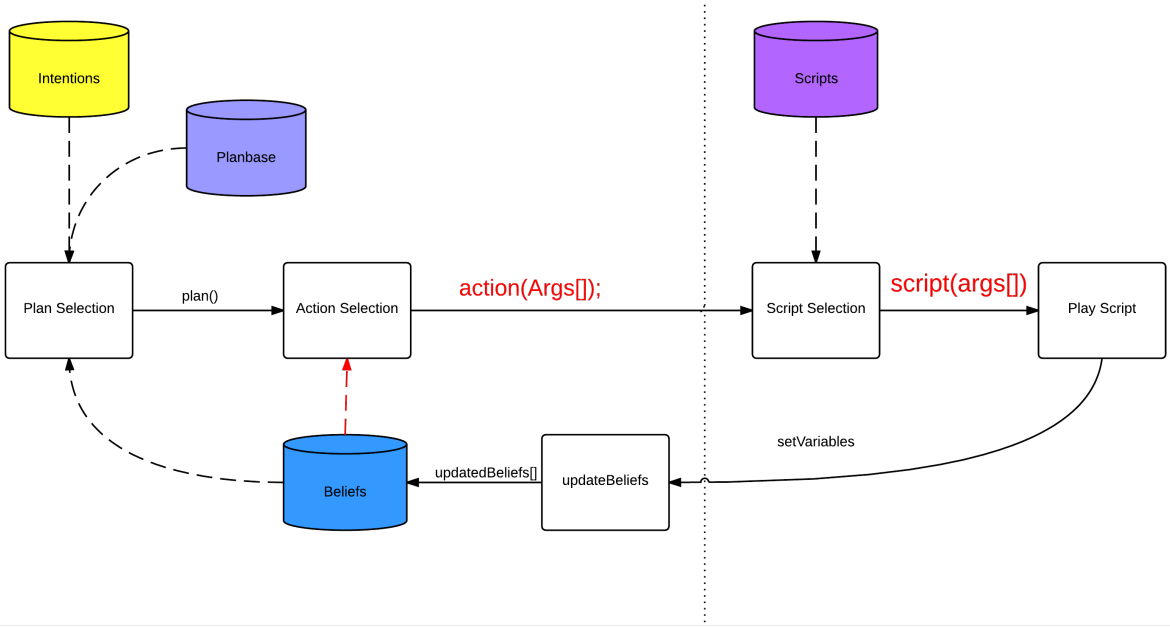

Figure 11: The most essential difference with the previous model is that in this model the action selection function of a plan uses beliefs as an input. Beliefs as belief(playerDetective) and belief(getProtection) are now incorporated in the action talkAboutVillageLife() as arguments. The action then becomes talkAboutVillageLife(playerDetective, playerJunky, getProtection, etc). So now we only have to create one plan with one, albeit more complicated, script.

String replacement

Once an action is selected and opens a script every argument given in args[] is a boolean. If the corresponding belief was true, the argument gives true (see 1). So, a script gets a list of booleans. In the script we have a list of variables (which we call #cond[i]# in figures 12 and 13), which can be replaced by a string if requirements are met. A requirement for each variable is a list of booleans which must be true and a list of booleans which must be false. In figure 13 we use one formula, which covers all variables in the script. This formula lacks power, for every argument given must be positive. Positive in a sense that for example a belief like belief(notAskedAboutCar) is true will lead to a non-empty sentence. While in this case you only want a non-empty sentence if you have already asked about his car. It would become something like " I know I asked about your car before, but I forgot what you said "; after which the general context-independent conversation about the car continues. Secondly, more complex conditions can not be expressed through this formula. For example if condition 1 and 2 are true then X but if 1 2 and 3 then not X but Y.

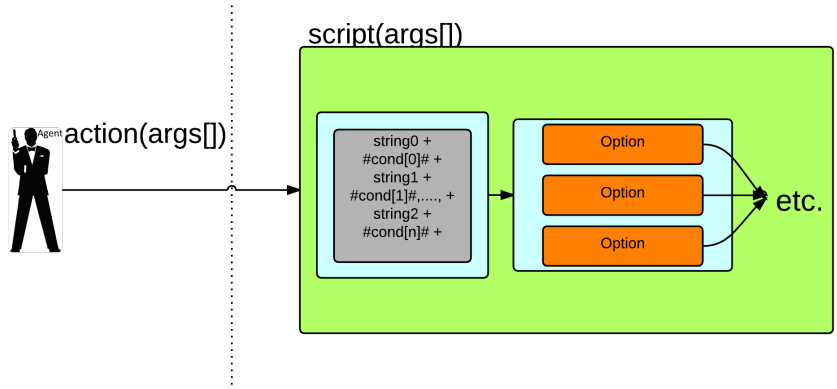

Figure 12: Once an action is selected and a script is opened, every argument given in args[] is a boolean. For every element in args[], if the corresponding belief of the agent was true, the element is true. If the corresponding belief was false, the element is false. Lets stay args[] is of size 4. Then every variable #cond[i]# in the script becomes replaced by its corresponding content (a string) if some formula is true. This formula can in this case consist of a conjunction of max 4 predicates, which state that an element of args[] should be true or false.

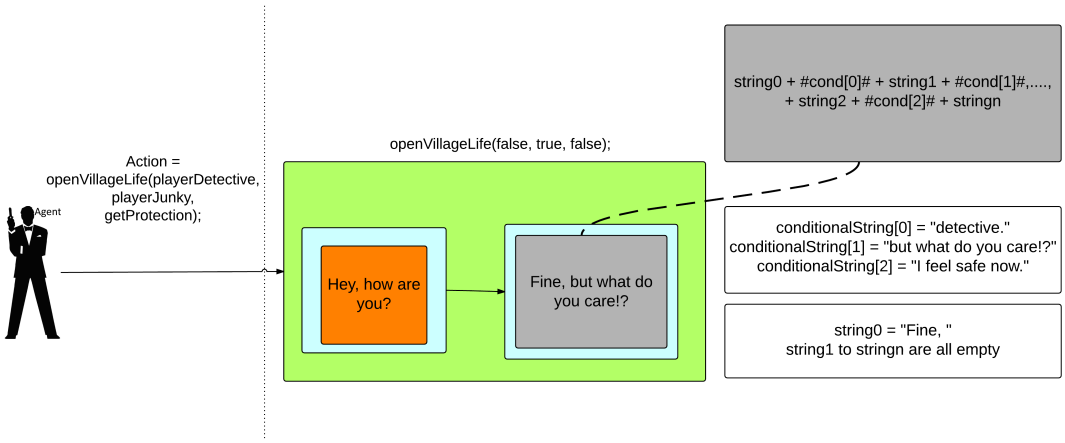

Figure 13: For the sake of simplicity, the formula we use for all conditional strings is:

$\forall args_i \in args[] :$ if $(args_i)$ then $(\#cond[i]\# = conditionalString[i])$ else $(\#cond[i]\# = "")$

The first argument is false, so cond[0] becomes an empty string. The second argument is true, so cond[1] becomes conditionalString[1], which is "but what do you care?!". The third argument is false, so cond[2] also becomes an empty string. Since string0 will always be "Fine," and string 1 to n will always be empty, we get "Fine, but what do you care?".

Conditional content

The second design uses conditional content. Implementation-wise it is different and we think it is a more natural design and easier to understand and use. Moreover, since we can now release conditional states, we can also set variables in conditional content. With string replacement this is not directly possible.

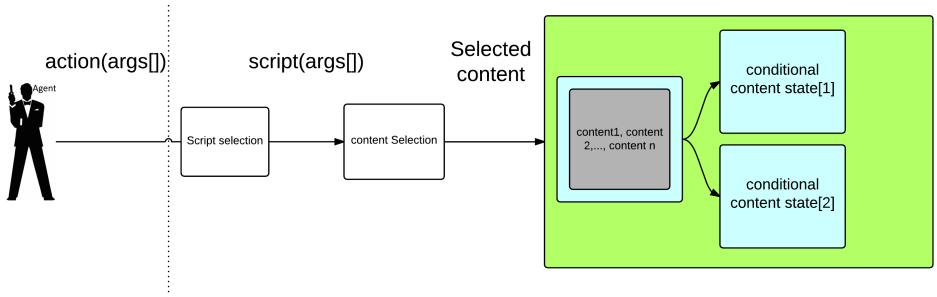

Figure 14: Instead of replacing variables with strings we write conditional content and release it based on the input of the action. Content can then be one element of a state or an entire state or even several states.

We use the same example as with the string replacement but immediatly also show how easy it is to add complete states (see figure 15), whereas with string replacement this could be hard.

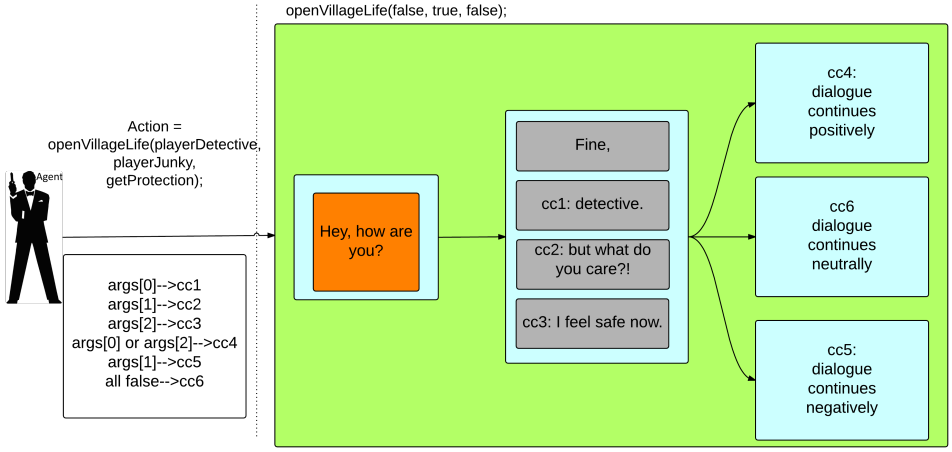

Figure 15: Only args[1] is true. So, cc2 and cc5 become released. cc5 constitutes an entire state. In this state several commands could be placed. For example, a statement by the NPC, followed by a variable that will be set, followed by an option the player has to take.

4.3.5 Transitions between scripts

The script writer has to think of certain requirements that should be met for the script. Who starts the script? What message should the audience receive? Who ends it, etc. A script that is ended by the player, followed by a script that is started by the agent should have some transition describing the relation between the two scripts. If the agent disagrees with what is said, this transition could be something like "bullshit, " followed by the script. This could imply that we, worst case, would have to write a transition for every possible succession of actions, or the agent could become less believable. This would completely nullify scalability initially provided by using BDI. This is why we want to design some mechanism, which automatically inserts a transition between two actions.

In discourse analysis we call this glue, which connects otherwise two seemingly disjoint utterances (in our case scripts), discourse markers. A transition in this sense is actually part of the context, but because it is a specific type of context we can try to think of a modular way to cope with the context.
We add the transition mechanism in our design models

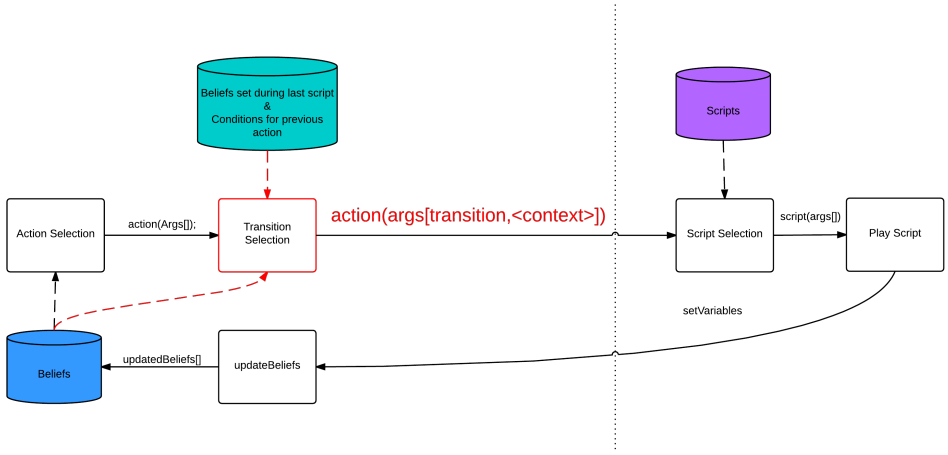

Figure 16: This image only shows the part of the process posterior to plan selection. The action now gets another argument. The first argument of an action refers to the transition. We combine the transition with the script that expresses the action chosen by the agent.

The transition selection mechanism

As explained in the background section we can categorize discourse markers. Louwerse and Mitchell [40] categorise them on continuity of the topic, acknowledgement of the statement made, and emphatics to the interlocutor. Similarly, we categorize a pair of actions by opposing the beliefs that are the conditions for the actions. For example, consider two actions, the first taken by the player, the second taken by the NPC:

1. talkAboutFairGovernment
2. talkAboutUnfairGovernment

The conditions of action 1 partially conflict with the conditions of action 2. Action 1 was triggered because the player wants to talk about the government and believes the government is fair. Action 2 was triggered because, even though the agent also wants to talk about the government, it believes the government is unfair. According to the dimensionality of Louwerse and Mitchell [40], we say it belongs to the category of positive continuation (same subject) and partial acknowledgement.[1] Now let's assume the agent is sympathetic with the player, to fill in the last dimension of emphatics. In our set of discourse markers we have the same category; "anyway" and "but then again" show positive continuity, partial acknowledgement and emphatics. So, the agent would probably put the words "but then again" in between the two subsequent scripts. For obvious reasons, we also take into account the variables set during the previous script. For example, perhaps the player started a conversation about the government, initially with no opinion about the fairness of the government. However, halfway the script it had to option to state it thinks the government is either fair or unfair. This has an influence on the acknowledgement dimension. Once we have chosen a transition, we need some way to include it in the script. In this case string replacement would probably be an easy and effective solution. Where every script always starts with a variable that will be replaced by the first argument of the action, which will be the discourse marker.

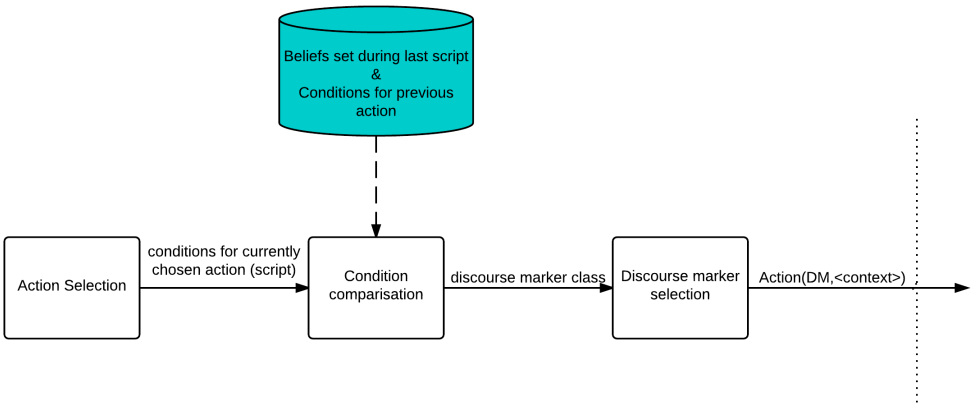

Figure 17: So we check the conditions for the currently selected and last selected action together with beliefs set during the last script. If and how beliefs between the different conditions conflict, determines what kind of glue is necessary between two scripts. This class is sent to a discourse marker selection mechanism. This last step is only meaningful if there are more then one possible discourse markers per class. You could then create some variance by randomly picking discourse markers that represent the same class.

[1]Partial, in this case, means not any. Louwerse and Mitchell (Louwerse, 2003) do not define this category in the dimension of acknowledgement

4.3.6 Final model

We conclude the section with our total model. A plan is selected based on beliefs, desires and intentions. In the plan some action is selected. The context-relevant beliefs for the action are then added as arguments. The action with its context-relevant beliefs is sent to the transition mechanism. Here, a discourse marker is selected, which will function as transition. The transition is added to the head of the list of arguments of the action. The agent opens a script with the action. In the script the transition and relevant conditional content is included. During the script possible variables are set; these represent the players agency within a script. When the script is finished, the set variables is sent to the agent to be converted to new beliefs. New plans are selected.

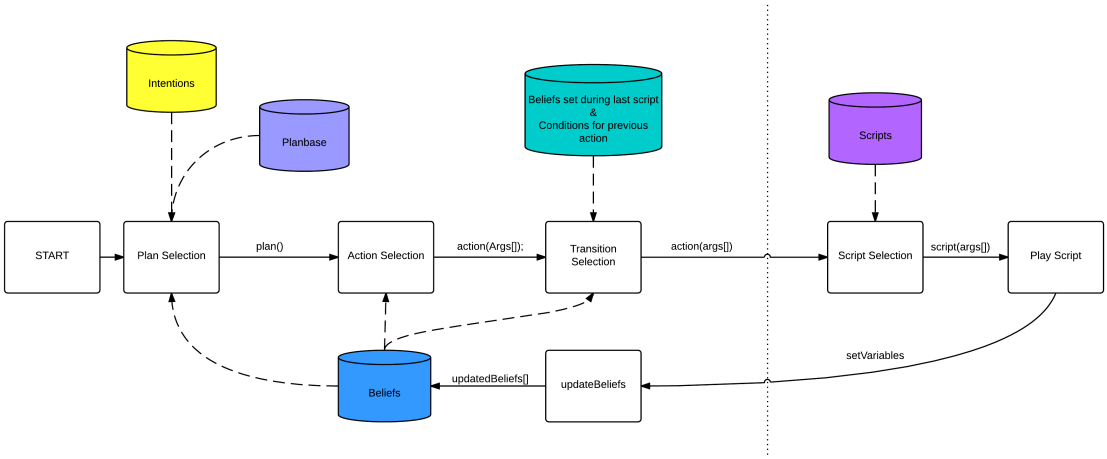

Figure 18

4.4 Second Design

As described in 3.2, we designed an exercise for two game designers of &ranj. A short presentation was given, separately, on the current research. We then presented them with a conceptual game, which was a simplified version of game(2) from section 3.1.2. A general storyline, the BDI of the agent and the goal of the game were given as extra information for the exercise. The task of the game designers then was to use filled in script building blocks to create scripts for the conceptual game. After the exercised was finished, we conducted an interview. The exact exercise can be found in Appendix B.

Points that were discussed are:

- Building the actions
- The script building block's shape and function

4.4.1 Building the actions

We argued that determining the size of an action has a direct influence on the trade-off between expressive control and interactivity. This is why it is of major importance to think carefully when doing so. Both game designers argued that this will probably be done at the first two stages (see 1) of the game production process. Depending on what the goals of the game are, it could be argued whether at some point in the game it should offer interactivity or not.

4.4.2 The script building block's function

We proposed that the building block's function would be for the programmer to communicate with the game designer. After designing the general concept and BDI of the game, the programmers would then implement the BDI and its reasoning engine. After which the game designers would have lost their view on how exactly the BDI has been implemented. This is why we needed the script building block. The programmers could fill in everything in the building block, hand it over to the game designers, and the game designers could write scripts with the OZ Dialog Editor and all would be fine. This was designed from the view of a programmer and, not very unsurprisingly, this lacked the game designers' view. The conducted interviews resulted in four insights (see figure 19 and a filled in example in Appendix C):

1. Since the general concept of the game and the general BDI are created by a team of both game designers and programmers, and the action's specifics shall be determined at this cooperative stage of the process, we fill in the script building blocks at these stages of the process. They then no longer function as a communicative medium from the programmers to the game designers, but as administration. The script building blocks are constructed in such a way that they support the programmers in designing the BDI and support the game designers in writing their scripts. If they both follow what has been decided and administrated in the cooperative stages of the production process, the BDI and scripts should coalign, even when the task-segregation is sustained.

2. Game designers noted that they could not really create a script based on the building block alone. They noted that any script is created with the game in its entirety in mind. Even though this statement emanates from their strong-story perspective, it is still salient in general. At &ranj, every game exists of certain beats [70] or levels and specific dialogues can only take place in specific beats or levels. In this case, the relevance of thinking of the story in its entirety whenever writing a script is high. Nevertheless, even in strong-autonomy games, some understanding of the whole is required to write the scripts. However, we think this was a deficiency of the exercise. Obviously, the conditions in which the actions can occur, plus the input arguments give an idea of the action's relation to the game in its entirety. However, even though we tried to clearly explain the concept of conditions, they did not use them as context for the script. Thus,

initially, we ought to remove this element of the script building block.[1] However, as they stated, they write scripts always in relation to the whole. Since actions are defined in the first two stages of the development process (see 1), their relation to the whole should be defined at these stages as well. This means conditions of actions are defined at these stages, and programmers should not deviate from them when implementing the BDI. In other words, we need this part of the script building block, not for the game designers, but for the programmers.

3. It was stated that the "message to the audience" field should not be a story (this is how we filled in this field in the exercise in Appendix B). Rather, they prefer a list of keywords of subjects to be discussed in the dialogue. Furthermore, every input argument (element of the context) should be noted in the "message to the audience" field with its corresponding keyword describing the conditional content's topic or subject to be discussed.

4. If there is some structure of beats and actions can only be applied during certain beats, we could argue that this information should be stored. Albeit in the script building block or in an extra visualization showing the script-structure of the game. We considered two types of ordering of beats, which could be added to the script building block. Transitive ordering with integers or intransitive ordering with strings.

    - **Transitive ordering**: If in a game it is the case that certain events can happen from a certain moment, we could argue that we categorize the beat with an integer. So, if the game has 15 beats and the beat number on the script building block of an action says 10, then the action can be applied in beats 10 to 16. Moreover, the context of the script is that beats 1 to 9 have occurred. Additionally, if there is no beat number given, we could say the action is applicable at any time of the story. This probably is most, or even only, suitable for a strong story game.
    - **Intransitive ordering**: If in a game certain events can only occur during a specific state of the agent or the environment, it would make more sense ordering beats based on the respective states. In this case a set of strings would most accurately describe the beats in which this action could be applied. However, this would probably be similar to the context-conditions of an action. If the game would become of considerable size, then game designers might use specific context-conditions which indicate best for them in what kind of context the script should be written.

[1] In Appendix B, the conditions that triggered the action are not actually put in the building block. However, during the exercise, it was made clear that they originally did belong to the building block and that they could use this information to write the scripts.

The new script building block is illustrated below:

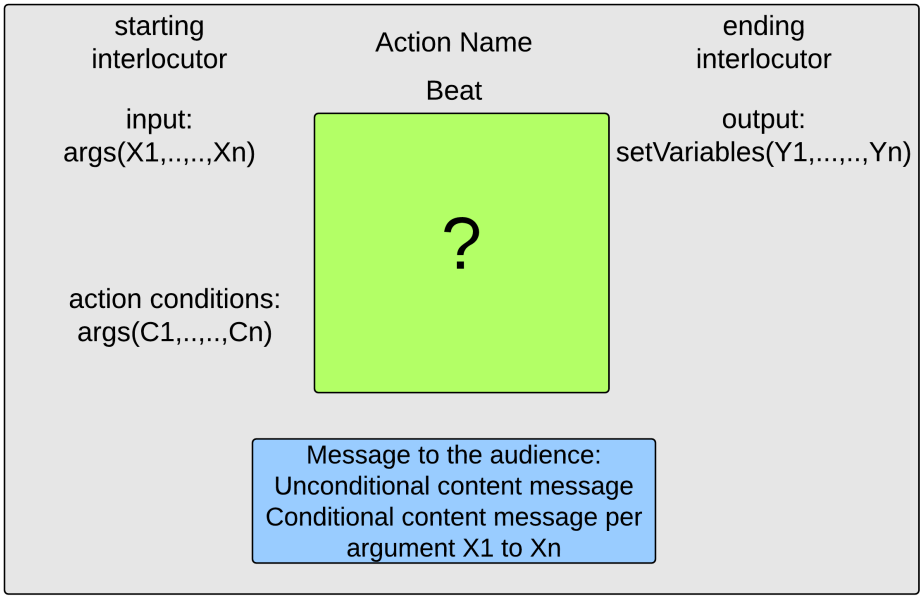

Figure 19

4.5 Reusability

Once a game has been created we have a database of actions translated with scripts, plans explained through actions and characters described by a BDI. A new character could probably be added quite easily by copying part of the BDI. Generally, it is believed certain modules of a BDI are reusable for other agents [71, 72]. However, we believe that scripts could probably be reused in other agents and games as well, to express the same actions. One issue with this presumption, is that we combine BDI and scripting to put actions to a higher-level. High-level actions are created because specific details can add a lot of nuance to an action, which only makes sense in the action in its entirety. Very specific scripts are most likely not reusable. However, we believe that context-dependent actions perhaps offer more reusability. Their general content could be reused and for every element of the context some conditional content is thought of and this specific conditional content for this action could be reused as well. If and how this works is something that would become apparent when an actual game would be made with our approach.

4.6 Conclusive remarks

In this chapter we elaborated on the motivations for certain design-choices that in their turn impose certain limitations. Based on these design-choices we illustrated our initial design. We illustrated an action-selection-action-expression-process, design patterns which describe the authorial affordances for the action-expression mechanism, and a script building block. Additionally, we proposed modifications to the action-selection mechanism, which we think are an addition to the practicability of the approach. We then tested our design by conducting two interviews with game designers and discussed their remarks. This led to a change in the function and shape of the building block and gave us some insight in the issue of finding the right size of an action.

5 Conclusion

In this thesis, we looked to design an architecture for creating interactive narratives in which the focus lies, not on a strong story whose dramatic beat order is omnipresent, but rather on a strong character who is highly responsive to the player's moves. We addressed the issue of both, the lack of expressive control over the character's expression supported by current approaches for creating interactive characters, and the lack of agency players experience in their interaction with characters whose dialogues are fully scripted; accompanied with the supposition that creating complex dialogue structures is impracticable when dialogues are fully scripted.[1]

We support the expressive AI approach [17]. Interpretive affordances give the audience an idea of what kind of actions should and could be taken and authorial affordances give the artists the tools to inscribe their intention in the machine. From this perspective we sought for a solution to the issues addressed. We found that, whatever framework might be used, socialness and believability should be prioritized. An architecture that enforces rationality and intelligence would be inappropriate for building believable agents. An architecture that enforces realism would generate externally realistic behaviour and realistic internal processing, but this is not our goal.

We stated that believable agents should have strong personalities. On strong personalities we are able to use a Theory of Mind on them to form a social model. We rely on this model to predict the consequences of our actions. This exemplifies the aforementioned interpretive affordances. We argue that a sociable agent with a strong personality possesses a combination of aspects and our architecture should support the authorial affordances to bring about these aspects in the agent. The agent should be able to:

- Act aware of past actions
- Act aware of its social role in context of social norms and values of the environment
- Act emotionally, emotions need to expressed broadly
- Act aware of its environment
- Engage in joint activities
- Have several behaviours run in parallel
- Express changes in behaviour in some manner; when using discourse markers this means it should be able to classify its actions
- Initiate its own turns

We argued that the BDI framework is stark and in this regard it does not enforce rationality, intelligence, or realism. Moreover, it does allow for extensions to the framework. These extensions could cover all of our desired properties. However, the authorial

[1] For current approaches, read: current language processing and language generation methods

control scripting offers could substitute the requirement for these properties. Then, instead of having an actual emotional agent, the agent expresses its actions with certain emotions that are encapsulated by the scripts. We illustrated a mechanism to include context in actions and a mechanism to express changes in behaviour. Moreover, we discussed research on the other desired aspects with respect to BDI [31,33,49,51–53,63,73]. [1]

BDI's restrictions become apparent when we try to express the agent's intentions in a believable manner. Previous games made with BDI [1, 68] did not require the agents to be eloquent and nuanced in their use of language. Moreover, the approach of using language templates, used in Glengarry Glen Ross [1], did not offer the authorial affordances to create such an agent. This is why we argued to bring actions to a higher level and script the expression of that action as a whole. This way emotions can be expressed broadly, and awareness of social context could be expressed quirky, nuanced and ambiguous like no language generating agent has done before.

As stated before, by scripting actions, we construct fixed parts. Even though it is possible to offer interaction within those fixed parts, all interaction is in this fixed part is predetermined. So, we arrive at a new balance between expressive control and autonomous interactive agents.

RQ 1: Can we gain an understanding of the process of creating a game with our approach of BDI and scripting?
To gain an understanding we designed an architecture constructed of a combination of BDI and scripted dialogues. The exact design of our architecture was shaped by the process and task-division that takes place at the game development at &ranj. Our main concern was to keep the tasks of action-selection and action-expression segregated in such a way that the programmer's task is to implement the action-selection part of the agent and the game designer's task is to implement the action-expression part of the agent. Based on this design-choice we construed a model describing action-selection-action-expression process, we illustrated some possible authorial affordances supported by scripting actions, and we designed a building block. This building block describes an action and all the building blocks together describe the game. We tested this architecture by analysing it with help of already existing games at &ranj and creating a conceptual game with paper-prototyped script building blocks. Next, we created an exercise in which two experienced game designers had to create this same game with filled in script building blocks we offered them. An interview was conducted to obtain their feedback about the architecture and our approach as a concept in general. In the method chapter, we posed questions to support the statement that it is not trivial to combine BDI and scripting. We found out that some questions are implicitly answered by others, so we do not discuss those.

[1] Mateas [33] as well as Sengers [31], did not specifically design their mechanisms for a BDI-agent. However, for both it is not hard to imagine how these could be added to the BDI framework.

Which parts of the game are covered by the BDI and which parts are covered with scripting?
The process model in section 4.3.6 illustrates this. Initially, the agent chooses plans based on its intentional state. These plans consists of actions the player can take or actions it take on its own initiative. Once an action is selected, the agent considers which beliefs are context-relevant. It searches for a cohesive device to give semantic meaning to the action currently selected and the previous action. The action is then expressed with the OZ Dialog Player. During the running of a script the player can influence the outcome of the script. This outcome is sent to the agent and the agent processes this into new beliefs. And the cycle starts again.

Can we sustain the production-efficient task-segregation currently present in serious gaming companies?
We proposed two restrictions required to sustain this task-segregation:

1. We are not able to use and update beliefs for reasoning within a script.
2. We can not stop scripts halfway

As long as these restrictions are maintained, we believe task-segregation can be sustained. However, we did foresee an issue where actions and scripts are not coaligned. A script building block was designed as administrative element of the architecture. At early production stages programmers, and game designers (and visual designers, if animations would be added to the game), together set up a collection of actions representing the game. The script building blocks have all information stored required to realize coaligned actions and scripts, even though they are constructed in segregation. The elements of a building block are:

- Action Name
- Beat
- Input Arguments: Transition + Conditional Content
- Output Arguments: Agency of player within a script
- Starting Interlocutor
- Ending Interlocutor
- Message to the audience: Unconditional + message per conditional content
- Context-conditions of action

How does the BDI "communicate" with scripts? The agent selects an action which opens a script. In the OZ Dialog Editor, artists can write the script; the expression of the action. Context-relevant beliefs are arguments of an action. These arguments are passed on to the script, and can trigger scripted conditional content. Interactivity within an action requires a combination of authoring both in the OZ Dialog Player as well as in the BDI, and is illustrated by the design patterns in section 4.3.3 . An action's output arguments define variables that can be set during the playing of a script. During a consequential choice, or consequential end-choice these variables are set. When a script is finished, the list of set variables is processed to new beliefs which are added to the agent's beliefbase.

RQ 2: Can we retain merits of BDI, like reusability, scalability and proactiveness, when combining it with the narrative control of scripted dialogues? In general, we think the merits of BDI as stated in 1.3.2 hold. There are some issues we came across, which we will discuss here.

Reusability:
We believe that certain modules of the BDI, like the transition mechanism, could be reused. Moreover, we argue that even scripts could possibly be reused. For example, we would need to create new actions for a new agent, but actions, which either fully or partially, express the same intention, could, respectively fully or partially, be used again. A prefatory greeting would probably have the same structure over different games and with different characters. We think that reoccurring context for a specific action could also be reusable in some occasions.

Scalability:
We mentioned that two subsequent utterances can be seemingly unrelated if there is no cohesive device describing their semantic relation. This could be both, when the utterances originate from the same agent and when the two utterances' origins are different agents. A lack of semantic relation implies a lack of interpretive affordances. This could imply that we, worst case, would have to write a transition for every possible succession of actions or the agent could become less believable. This would completely nullify scalability. To sustain scalability as well as believable flow, we designed a mechanism which incorporates discourse markers from the field of discourse analysis. Discourse markers function as cohesive devices giving two otherwise seemingly unrelated utterances their semantic meaning. Louwerse and Mitchell [40] categorised discourse markers in three dimensions: continuity, acknowledgement, and emphatics. We proposed that actions could be categorised the same way. The mechanism would then select the appropriate transition between two scripts based on the categorisation. Still, if the artists decide that some transition does not really hit the mark, they could change this specific case and write the transition by hand.

Proactiveness:
We have not incorporated turn-taking specifically in our architecture, but in our exercise the agent did initiate its own action and there did not seem to be any additional restrictions necessary to the architecture to incorporate turn-taking. However, within a script there is no room for turn-taking. So, in a game where turn-taking is a prominent part of the game, the size of actions would generally be smaller.

RQ 3: How should the size, or level-height, of an action be decided?
We already argued that this is mostly subjective. The game designers that were interviewed agreed. They stated that at the first two steps of the game development process (see 1), game designers and programmers would together set up a general concept of the game and a general setup of the BDI. Through an understanding of the general concept of the game, it becomes apparent where interaction is most relevant. At these points in the game it could be opted to end a script, either because you would want the agent to be able to initiate its own turn from this point out, or because it is simply impractical to write every possible branching narrative path at that point. The latter could be a case where you would ask the agent what he wants to talk about. If there would be a hundred subjects, it would be illogical to write the dialogues of all those possible subjects in one script. We would not gain any control over separating the question and its hundred possible continuations, except for the hundred possible transitions between the two. However, we argued that this could be generalized with the transition mechanism without much loss of believability. Moreover, it is not difficult to imagine how initially a high-level action could be constructed, which later on in the development appears to be too big and some extra interaction is desired halfway. The script can be cut in two parts and some new context conditions possibly have to be thought of for them to become appropriate.

We believe we provided solid ground for future researchers in this field of study by clearly pin-pointing down what it means to express actions of a BDI with pre-written scripts.

6 Discussion

We argue that theory and practice of AI models are often too far apart. Some issues of an approach are overlooked or neglected when a problem is attacked at a purely conceptual level [69]. Andrew Stern argues that possibly fundamental points only rise to the surface when you actually try to create something with your AI framework [69]. For example, standard BDI might not offer the authorial affordances to shape our agents. Implementing the agent in a real example might show us that we need some way to track emotions or it might prove our intuition on the need of some sort of theory-of-mind-like mechanism. It also shows you how tools which are used to shape the agent might need adaptation. Even though the latter might not be real science, these are still things that are important to know in practice. Therefore it would be ideal to create a game with and without turn-taking and with this architecture to create a demonstration game.

6.1 Future research

The scope of this research was limited to creating conceptual games on paper rather than building a full game. Every discussion point here essentially would require an actual game to be made.

6.1.1 Context

We stated that reusability of actions would be lost without the ability to create context-dependent actions, which are expressed by scripts with conditional content. A good method for designing context-dependent actions could greatly improve the practicability, reusability and scalability of this approach.

We stated that instead of a exponential growth in amount of actions, we end up with one action, possibly including a large set of context. Obviously, this is not the holy grail. We move complexity from the agent to the script. It might prove to be too difficult for game-designers to great coherent conditional content for all contexts such that believable flow is kept.

Moreover, we cope with context on action level. However, we think it coheres most with folk-psychology to make this distinction at plan level. The intention to change the world to a certain state depends on the current state of the world. Its execution depends on the context. So, the plan depends on the context. Actions are the most elementary behaviours of the agent. Thus, it is weird that they could still have a different execution dependent on the context. This conceptual weakness may not actually seem that important. But if our proposed mechanism fails, this might be something to reflect on.

6.1.2 Script building block

We state that the script building block should help produce coaligned scripts and actions. However, we have no indication at all that this actually still holds on a larger scale.

Classifying scripts

We believe that there is probably some modularity to be found in categorising scripts. For example, two successive scripts, the first one ended by the player and the second started by the NPC probably have some specific type of transitions. Moreover, a script ended by the player could have a different impact on the turn-taking mechanism than a script ended by the NPC.

6.1.3 Transitions

We proposed that actions or scripts could probably be categorised the same way as discourse markers can. In designing our overly simple conceptual game, we did not find any counter-evidence. However, games on a large scale could show us very different situations in which it would not work. Or it might show that categorisation of actions, with the current discourse marker categorisation frameworks, is impractical on a larger scale. It might also show that players do not feel any schizophrenia [29], even if one script bluntly follows up on another. Then the whole transition mechanism would be unnecessary.

We also indicated that comparing two actions based on the conditions of the intentions might not work. Actions could belong to intentions with several plans, which then again could have several actions. Some of the conditions, which triggered the intention to become active, might be out of context of the specific actions for which we are looking a transition for. A second solution is by letting the BDI explain its own reasoning [50]. The agent would say his belief about X has changed, which causes it to pursue goal G. However, considering believability, we argue that this might not be the most poetic solution. Nevertheless, one could think of alternate solutions. Like showing a balloon filled in with the thought process of the agent. Perhaps player's suspension of disbelief is sustained this way.

6.1.4 Turn-taking

We did not actively address turn-taking in our architecture. Some indication of when turns can be taken is inherent to this approach, namely, when some script is finished. We suspect that some new kind of condition should be added to the action. If conditions are met, a turn can be taken with this specific action. Moreover, we do suspect that this would generally result in smaller actions and we do not know whether this has a large impact on the effectiveness of our approach. Turn-taking might need the introduction of a new set of discourse markers. For example, saying "ehh" in a conversation may indicate that the speaker wishes to take the turn, even though it does not know exactly what to say yet [74].

References

[1] T. J. Muller, A. Heuvelink, K. van den Bosch, I. Swartjes *et al.*, "Glengarry glen ross: Using bdi for sales game dialogues." in *AIIDE*, 2012.

[2] B. G. Studios, "The elder scrolls iv: Oblivion, 2k games," *Rockville, MD*, 2006.

[3] M. Peeters *et al.*, "Personalized educational games-developing agent-supported scenario-based training," *SIKS Dissertation Series*, vol. 2014, 2014.

[4] I. M. T. Swartjes, *Whose story is it anyway?: How improv informs agency and authorship of emergent narrative.* University of Twente, 2010.

[5] R. Aylett, "Narrative in virtual environments-towards emergent narrative," in *Proceedings of the AAAI fall symposium on narrative intelligence*, 1999, pp. 83–86.

[6] K. Huotari and J. Hamari, "Defining gamification: a service marketing perspective," in *Proceeding of the 16th International Academic MindTrek Conference.* ACM, 2012, pp. 17–22.

[7] M. Csikszentmihalyi, *Finding flow: The psychology of engagement with everyday life.* Basic Books, 1997.

[8] D. J. Shernoff, M. Csikszentmihalyi, B. Shneider, and E. S. Shernoff, "Student engagement in high school classrooms from the perspective of flow theory." *School Psychology Quarterly*, vol. 18, no. 2, p. 158, 2003.

[9] R. Aylett, R. Figueiredo, S. Louchart, J. Dias, and A. Paiva, "Making it up as you go along–improvising stories for pedagogical purposes," in *Intelligent Virtual Agents.* Springer, 2006, pp. 304–315.

[10] A. Bartish and C. Thevathayan, "Bdi agents for game development," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 2.* ACM, 2002, pp. 668–669.

[11] M. Bratman, "Intention, plans, and practical reason," 1987.

[12] A. S. Rao, M. P. Georgeff *et al.*, "Bdi agents: From theory to practice." in *ICMAS*, vol. 95, 1995, pp. 312–319.

[13] N. R. Jennings and S. Bussmann, "Agent-based control systems," *IEEE control systems*, vol. 23, no. 3, pp. 61–74, 2003.

[14] L. Braubach, W. Lamersdorf, and A. Pokahr, "Jadex: Implementing a bdi-infrastructure for jade agents," 2003.

[15] A. Bogaard van den, "Bdi in games," 2012.

[16] T. Kamermans, "Realistic game dialogues with ai," 2013.

[17] M. Mateas, "Expressive ai: A hybrid art and science practice," *Leonardo*, vol. 34, no. 2, pp. 147–153, 2001.

[18] S. S. Iyengar and M. R. Lepper, "When choice is demotivating: Can one desire too much of a good thing?" *Journal of personality and social psychology*, vol. 79, no. 6, p. 995, 2000.

[19] J. S. Bruner, *Acts of meaning*. Harvard University Press, 1990.

[20] T. Porter, "Depicting perception, thought, and action in toy story," in *First International Conference on Autonomous Agents*, 1997.

[21] G. R. Jonsdottir, K. R. Thorisson, and E. Nivel, "Learning smooth, human-like turntaking in realtime dialogue," in *Intelligent Virtual Agents*. Springer, 2008, pp. 162–175.

[22] M. Harbers, K. Van den Bosch, and J.-J. Meyer, "Modeling agents with a theory of mind: Theory–theory versus simulation theory," *Web Intelligence and Agent Systems*, vol. 10, no. 3, pp. 331–343, 2012.

[23] M. Mori, "Bukimi no tani [the uncanny valley], energy,[online] 7 (4), 33-35," 1970.

[24] K. Cheng and P. A. Cairns, "Behaviour, realism and immersion in games," in *CHI'05 extended abstracts on Human factors in computing systems*. ACM, 2005, pp. 1272–1275.

[25] F. Thomas and O. Johnston, *Disney animation: The illusion of life*. Abbeville Press, 1981.

[26] J. Bates *et al.*, "The role of emotion in believable agents," *Communications of the ACM*, vol. 37, no. 7, pp. 122–125, 1994.

[27] S. Smith and J. Bates, "Towards a theory of narrative for interactive fiction," 1989.

[28] W. S. Reilly, "Believable social and emotional agents." DTIC Document, Tech. Rep., 1996.

[29] P. Sengers, "Schizophrenia and narrative in artificial agents," *Leonardo*, vol. 35, no. 4, pp. 427–431, 2002.

[30] M. Johansson, "Do non player characters dream of electric sheep?: A thesis about players, npcs, immersion and believability," 2013.

[31] P. Sengers, "Do the thing right: an architecture for action-expression," in *Proceedings of the second international conference on Autonomous agents*. ACM, 1998, pp. 24–31.

[32] C. Breazeal, "Toward sociable robots," *Robotics and autonomous systems*, vol. 42, no. 3, pp. 167–175, 2003.

[33] M. Mateas and A. Stern, "A behavior language: Joint action and behavioral idioms," in *Life-Like Characters*. Springer, 2004, pp. 135–161.

[34] H. Sacks, E. A. Schegloff, and G. Jefferson, "A simplest systematics for the organization of turn-taking for conversation," *language*, pp. 696–735, 1974.

[35] H. Prendinger and M. Ishizuka, "Social role awareness in animated agents," in *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001, pp. 270–277.

[36] D. Tannen, "10. when is an overlap not an interruption? one component of conversational style," *Selected Papers*, vol. 119, 1983.

[37] M. Halliday, "K, and hasan, r.(1976). cohesion in english," *L ondon: L ongman*.

[38] A. Knott, "A data-driven methodology for motivating a set of coherence relations," 1996.

[39] D. Schiffrin, *Discourse markers*. Cambridge University Press, 1988, no. 5.

[40] M. M. Louwerse and H. H. Mitchell, "Toward a taxonomy of a set of discourse markers in dialog: A theoretical and computational linguistic account," *Discourse Processes*, vol. 35, no. 3, pp. 199–239, 2003.

[41] L. Polanyi and R. J. Scha, "The syntax of discourse," *Text-Interdisciplinary Journal for the Study of Discourse*, vol. 3, no. 3, pp. 261–270, 1983.

[42] V. Petukhova and H. Bunt, "Towards a multidimensional semantics of discourse markers in spoken dialogue," in *Proceedings of the Eighth International Conference on Computational Semantics*. Association for Computational Linguistics, 2009, pp. 157–168.

[43] T. Bickmore and J. Cassell, "how about this weather?" social dialogue with embodied conversational agents," in *Proc. AAAI Fall Symposium on Socially Intelligent Agents*, 2000.

[44] K. Aijmer, *English discourse particles: Evidence from a corpus*. John Benjamins Publishing, 2002, vol. 10.

[45] M. Groen, J. Noyes, and F. Verstraten, "The effect of substituting discourse markers on their role in dialogue," *Discourse Processes*, vol. 47, no. 5, pp. 388–420, 2010.

[46] G. Erkens and J. Janssen, "Automatic coding of dialogue acts in collaboration protocols," *International Journal of Computer-Supported Collaborative Learning*, vol. 3, no. 4, pp. 447–470, 2008.

[47] A. B. Loyall, "Believable agents: building interactive personalities," Ph.D. dissertation, Mitsubishi Electric Research Laboratories, 1997.

[48] M. Dastani, "2apl: a practical agent programming language," *Autonomous agents and multi-agent systems*, vol. 16, no. 3, pp. 214–248, 2008.

[49] M. P. Sindlar, M. M. Dastani, and J.-J. C. Meyer, "Bdi-based development of virtual characters with a theory of mind," in *Intelligent Virtual Agents*. Springer, 2009, pp. 34–41.

[50] M. Harbers, "Explaining agent behavior in virtual training," *SIKS disseratation series*, vol. 2011, no. 35, 2011.

[51] E. Norling and L. Sonenberg, "Creating interactive characters with bdi agents," in *Proceedings of the Australian Workshop on Interactive Entertainment (IE'04)*, 2004, pp. 69–76.

[52] M. Dastani and J.-J. C. Meyer, "Programming agents with emotions," in *ECAI*, 2006, pp. 215–219.

[53] S. Marsella, J. Gratch, and P. Petta, "Computational models of emotion," *A Blueprint for Affective Computing-A sourcebook and manual*, pp. 21–46, 2010.

[54] A. Guerra-Hernández, A. El Fallah-Seghrouchni, and H. Soldano, "Learning in bdi multi-agent systems," in *Computational logic in multi-agent systems*. Springer, 2005, pp. 218–233.

[55] A. Guerra-Hernández and G. Ortiz-Hernández, "Toward bdi sapient agents: Learning intentionally," in *Toward Artificial Sapience*. Springer, 2008, pp. 77–91.

[56] T. Phung, M. Winikoff, and L. Padgham, "Learning within the bdi framework: An empirical analysis," in *Knowledge-Based Intelligent Information and Engineering Systems*. Springer, 2005, pp. 282–288.

[57] V. Dignum, J.-J. Meyer, H. Weigand, and F. Dignum, "An organization-oriented model for agent societies," 2002.

[58] D. Kinny and M. George, "Commitment and e ectiveness of situated agents," in *IJCAI-91*, 1991, pp. 82–88.

[59] P. R. Cohen and H. J. Levesque, "Teamwork," *Nous*, pp. 487–512, 1991.

[60] M. Fisher, "Metatem: The story so far," in *Programming multi-agent systems*. Springer, 2006, pp. 3–22.

[61] G. De Giacomo, Y. Lespérance, and H. J. Levesque, "Congolog, a concurrent programming language based on the situation calculus," *Artificial Intelligence*, vol. 121, no. 1, pp. 109–169, 2000.

[62] K. V. Hindriks, F. S. De Boer, W. Van der Hoek, and J.-J. C. Meyer, "Agent programming in 3apl," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357–401, 1999.

[63] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A bdi reasoning engine," in *Multi-agent programming*. Springer, 2005, pp. 149–174.

[64] A. M. Turing, "Computing machinery and intelligence," *Mind*, pp. 433–460, 1950.

[65] J. Bates, B. Loyall, and W. S. Reilly, "Broad agents," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 38–40, 1991.

[66] B. Loyall, A. Bryan, and L. J. Bates, "Hap a reactive, adaptive architecture for agents," 1991.

[67] M. Mateas and A. Stern, "Façade: An experiment in building a fully-realized interactive drama," in *Game Developers Conference*, vol. 2, 2003.

[68] J. Van Oijen, W. Van Doesburg, and F. Dignum, "Goal-based communication using bdi agents as virtual humans in training: An ontology driven dialogue system," in *Agents for games and simulations II*. Springer, 2011, pp. 38–52.

[69] A. Stern, "Virtual babyz: Believable agents with narrative intelligence," *Advances in consciousness research*, vol. 46, pp. 215–228, 2002.

[70] M. Mateas and A. Stern, "Towards integrating plot and character for interactive drama," in *Socially Intelligent Agents*. Springer, 2002, pp. 221–228.

[71] L. Braubach, A. Pokahr, and W. Lamersdorf, "Extending the capability concept for flexible bdi agent modularization," in *Programming multi-agent systems*. Springer, 2006, pp. 139–155.

[72] L. Chang and X. He, "Towards adaptable bdi agent: A formal aspect-oriented modeling approach." in *SEKE*. Citeseer, 2009, pp. 189–193.

[73] N. Vergunst, "Bdi-based generation of robust task-oriented dialogues," *SIKS dissertation series*, vol. 2011, 2011.

[74] V. Petukhova, L. Prévot, and H. Bunt, "Multi-level discourse relations between dialogue units," in *Proceedings 6th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (ISA-6), Oxford*, 2011, pp. 18–27.

Appendix A

JADEX implementation in ActionScript®3

This is an overview of the current state of the implementation of JADEX in ActionScript®3 by van den Bogaard [15]. This implementation is quite limited. However, we did not actually let our architecture be influenced by this. To create a demonstration game with the approach presented in this thesis, most likely this should be extended to some degree.

- A belief is an object of any value which has an action script signal which listens when a belief is set to some value. All beliefs together are in the agent's belief base.
- A goal is an object which can be in an active, option or suspend state with the following arguments in its constructor:
  - Parameters: Any property which belongs to the goal
  - Creation Conditions: Conditions which must be true before the goal is added to the goal base. These will most often be other goals
  - Context Conditions: Conditions which must be true for the goal to not be suspended. These will most often be beliefs
  - Drop Conditions: If all drop conditions are true the goal will be set to option and will no longer be adopted (and as such will not be deliberated on until told otherwise)
  - Recur Conditions: If all recur conditions are true the goal will be adopted again.

  A goal is suspended if it does not meet context conditions. A goal is optioned if context conditions are met but there is no possible plan. A goal is active if context conditions are met and there is a plan which could be activated.
- A plan is an object which can be in an active, inactive or suspended state and the following arguments are given in its constructor:
  - Triggers: Triggers for a plan will be goals, message events, changed facts, added facts, and internal events.
  - Preconditions: conditions for the plan to be activated from an inactive state.
  - Context conditions: conditions which decide whether the plan should be in active or suspended state.
  - Drop conditions: If drop conditions are met the plan will be set to inactive. Its body consists of any sequence of actions.

- The goal and plan deliberation algorithms are at its most basic form. Currently there is no actual reasoning on the effectiveness of plans. What is more, the initial implementation goes by the notion that every plan only has one possible goal as trigger.

Appendix B

Exercise for interview

In this section we show the exercises used in the testing. The exercises are quite crude and can be slightly vague. In the exercise we gave the game designers extra information about the precise intention of the exercises. If readers have questions about the exercises we are willing to answer them by mail.

We first showed them some drawn examples with explanation of how we thought actions could look like. (They can be requested by mail. They are really poorly drawn with scratches throughout et cetera. So, we will not show them here).

We distuinguish actor roles from character roles as explained in the joint action part of section 2.2.2 .

Domain:
Interrogating a villager in a village where a gang is secretly producing and smuggling drugs.

Story:
You enter in a village where the villagers are supported by a gang through provisions. However, the gang is the main drug producer and forces the villagers to keep their mouth shut in return. Additionally, the villagers have to smuggle the drugs to a nearby metropolis and some of the gang members can be quite violent. The villager does want to get rid of the gang. However, he doesn't talk easily, obviously. He is scared like hell.

BDI:

Beliefs:
not(toldSmuggle)
not(getProtection)
not(playerJunky)
not(playerDetective)
not(likePlayer)
not(discussedOccupation)
not(discussedVillageLife)

Goals:
Actor goals:
optionConversationPlayerOccupation
optionConversationVillageLife

Character goals:
goWithCopToStation
tellAll

Intentions:
Actor Intentions:
giveOptionPlayerOccupation
giveOptionVillageLife

Character Intentions:
ending

Plans:

giveOptionPlayerOccupation:
contextConditions = {beliefBase.discussedOccupation == false}
do{
beliefBase.discussedOccupation = true;
openPlayerOccupation(transition, playerDetective)
}

giveOptionVillageLife:
contextConditions ={beliefBase.discussedVillageLife == false}
do{
beliefBase.discussedVillageLife = true;
openVillageLife(transition, playerDetective, playerJunky, getProtection);
}

ending:
contextConditions = {beliefBase.discussedVillageLife == false AND beliefBase.discussedOccupation == false AND beliefBase.end == false}
do{
beliefBase.end = true
openEnding(transition, playerDetective,toldSmuggle, getProtection, likePlayer);
}

Building blocks:

Action Name	openPlayerOccupation
Initiating Interlocutor	Player
Ending Interlocutor	NPC
Output Arguments	playerDetective, getProtection, playerJunky
Input Arguments	transition, playerDetective
Message to the Audience	Somewhere in the village a villager is wandering. You converse with him. You can choose how to introduce yourself; either as a junky or as a detective. Either way, you suggest you want to know where the drugs are at. If you're a detective, he will suggest his fear some way or another and you can offer him protection. If you're a junky he will suggest disliking you but you can recover somehow.

Action Name	openVillageLife
Initiating Interlocutor	Player
Ending Interlocutor	NPC
Output Arguments	playerDetective, likePlayer, toldSmuggle
Input Arguments	transition, playerDetective, playerJunky, getProtection
Message to the Audience	You ask a very open question. "How is life in the village?" The agent will show a lack of faith in you if he doesn't think you are a detective. He will have a cover-up story. You can try to recover, but if he also thinks that you are a junky he doesn't trust you at all. If he does think you're a detective he will start suggesting that life there isn't all that great. The conversation can go anywhere, depending on how you anticipate to his suggestions. If he thinks you offer him protection he can tell you about the fact that the gang uses villagers to smuggle drugs to a nearby metropolis.

Action Name	ending
Initiating Interlocutor	Player
Ending Interlocutor	Anyone
Output Arguments	none
Input Arguments	toldSmuggle, likePlayer
Message to the Audience	If he told you about the smuggling or if he likes you he will be your witness and tell the truth about everything. If not then he will leave you.

Appendix C

Filled in script building block: second design

Action Name	findOutAboutDrugs
Beat Number	1
Action Conditions	discussedAboutDrugsBefore == false
Initiating Interlocutor	Player
Ending Interlocutor	NPC
Output Arguments (can either be true or false)	playerDetective, getProtection, playerJunky
Input Arguments	transition, playerDetective
Message to the Audience	Say hello Propose as junky or detective Ask about drugs NPC shows fear NPC doesn't like junky As detective, can offer protection Conditional Content (playerDetective): if(detective) then (Start about protection)