

Steps Towards Creating Socially Competent Game Characters

GOTHENBURG MONOGRAPHS IN LINGUISTICS 44

Steps Towards Creating Socially Competent Game Characters

Jenny Brusk



UNIVERSITY OF GOTHENBURG
PHILOSOPHY, LINGUISTICS & THEORY OF SCIENCE

Dissertation for the degree of Doctor of Philosophy in Linguistics, University of Gothenburg

© Jenny Brusk, 2014

Cover: Thomas Ekholm

Picture: © Jenny Brusk 2014

Printed by Reprocentralen, Humanistiska Fakulteten,
University of Gothenburg, 2014

ISBN 978-91-628-8890-9

Distribution:

Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg, Box 200, S-405 30 Gothenburg, Sweden

For Thomas, Thea and Mira

Abstract

Ph.D. dissertation in Linguistics at University of Gothenburg, Sweden, 2014

Title: Steps Towards Creating Socially Competent Game Characters

Author: Jenny Brusk

Language: English

Department: Department of Philosophy, Linguistics and Theory of Science,
University of Gothenburg, Box 200, S-405 30 Gothenburg

Series: Gothenburg Monographs in Linguistics 44

ISBN 978-91-628-8890-9

This thesis investigates and presents approaches towards creating more socially competent NPCs by means of natural language dialogues. The aim is to provide hands-on solutions for game developers who want to offer interactions with the NPCs in the game that are socially driven rather than functionally motivated and that encourage the player to build and maintain relationships between the character they control and the other game characters. By means of gameplay design patterns (GDPs), i.e. a semi-formal means of describing, sharing and expanding knowledge about game design, a selection of games have been analysed in order to identify existing and hypothetical GDPs for game dialogues. The analysis resulted in a number of GDPs that support, or could support, social interaction among game characters. A selection of the identified patterns has then been modelled using Harel statecharts and implemented in State Chart XML, a candidate to become a W3C standard.

Keywords: dialogue systems, non-playable characters, computer games, SCXML, statecharts, socially oriented dialogues

Acknowledgements

This work would not have been possible without the support of my supervisors Torbjörn Lager, Staffan Björk, and Robin Cooper. As my primary supervisor, Torbjörn has been my faithful companion from the very beginning. His encouragement, patience and never-ending support have kept me going and this thesis would not have been without him. I am also deeply indebted to my assistant supervisor Staffan Björk, who has been a great friend, teacher, inspiration and a valuable sounding board for my work on games. Robin Cooper has also been involved in the process of writing this thesis, in particular in finalizing the script and reaching an end to this project. Robin has provided me with new energy, self-esteem, hope, and great feedback when I had almost given up.

I would also give my warm thanks to Simon Dobnik, who was my opponent during my final seminar. Simon's critique and well thought through feedback improved this thesis considerably.

Thanks to Svenska Spel, Gotland University, department of information and communication (IKI) at University of Skövde, department of FLOV University of Gothenburg and Graduate School for Language Technology (GSLT) for financial support.

Over the years I have had the opportunity to collaborate with some amazing people. First, I'd like to thank Mirjam Palosaari Eladhari, my former colleague at Gotland University, for her friendship, collaborations, interesting discussions and support throughout the tough years. Thanks also to Anna Hjalmarsson and Preben Wik at the Department of Speech, Music and Hearing at KTH for friendship and cooperation in the DEAL project. Visiting KTH always gave me a lot of positive energy! In 2009 I was invited as a visiting scholar at the Institute for Creative Technology, University of Southern California thanks to David Traum. My stay at ICT was one the greatest experiences I have had over these years and I learned immensely from David and the rest of the Natural Language group. Thank you for making me feel so welcome and for taking such interest in my research. It really boosted me! Thanks also for letting me take part in interesting seminars, lectures, and for openly sharing your research. A particular thanks to Ron Artstein who helped me with the statistics and taught me how to use it – I am so grateful! Special thanks also to Sudeep and Angela for being the best office neighbours!

Thanks also to all PhD students, supervisors and associates of GSLT and FLOV at the University of Gothenburg for interesting courses, discussions,

collaborations, feedback, support and fun. A special thanks to my reviewers Rolf Carlson and David House for always believing in me. I would also like to thank Staffan Larsson for providing me with valuable feedback at the early stage of my work, Sally Boyd for reading and giving feedback on my work on gossip, and Åsa Abelin for helping me with various problems concerning my studies.

Thanks to my friends and colleagues at the computer game development programs in Skövde University for keeping my spirits up, in particular Ulf Wilhelmsson for reading parts of my script and providing insightful feedback.

Several anonymous reviewers are worth their acknowledgements for taking their time to provide constructive feedback on the papers I have submitted to various conferences.

Thanks to all my friends and relatives who have stood by my side all these years. A special thanks to Malina, Pontus, Isak, and Karl for generously sharing their home whenever I needed to visit Gothenburg.

I could not have pursued this work without the support of my beloved family – Thomas, Thea, and Mira – you make everything worthwhile.

Contents

| | |
|--|-----------|
| CHAPTER 1: INTRODUCTION | 1 |
| 1.1 NATURAL LANGUAGE INTERACTION IN GAMES | 2 |
| 1.2 CHOICE OF TECHNOLOGY | 3 |
| 1.3 A SIMPLE GAME SCENARIO | 4 |
| 1.3.1 The Waiter Character | 4 |
| 1.3.2 Extending the Model with a Dialogue Manager | 6 |
| 1.4 THE STRUCTURE OF THE THESIS..... | 12 |
| CHAPTER 2: CONVERSATIONAL AGENTS | 14 |
| 2.1 CLASSIFICATION OF CAS | 14 |
| 2.1.1 Dialogue Systems..... | 14 |
| 2.1.2 Platform and Setting..... | 16 |
| 2.1.3 Examples of ECAs | 18 |
| 2.2 NPCs..... | 18 |
| 2.2.1 Natural Language Interaction in Games | 19 |
| 2.2.2 NPC Roles..... | 21 |
| 2.2.3 Believability of NPCs | 22 |
| 2.2.4 Examples of Conversational NPCs | 23 |
| 2.3 SOCIAL ACTIVITIES | 25 |
| 2.3.1 Context..... | 25 |
| 2.3.2 Communicative Acts..... | 26 |
| 2.3.3 Cooperation..... | 26 |
| 2.4 DIALOGUE MANAGEMENT TASKS | 28 |
| 2.4.1 Initiative | 29 |
| 2.4.2 Turn-taking..... | 29 |
| 2.4.3 Incremental Text or Speech Processing | 31 |
| 2.4.4 Multi-party dialogue | 31 |
| 2.5 POTENTIAL DESIGN DIFFERENCES BETWEEN PRACTICAL DS AND GAME DS ... | 32 |
| 2.5.1 Correctness and cooperativeness | 33 |
| 2.5.2 Reliability and Efficiency | 34 |
| 2.5.3 Error Handling | 35 |
| 2.5.4 User Role and Setting | 35 |
| 2.6 RULE-BASED APPROACHES FOR DIALOGUE MANAGEMENT..... | 36 |
| 2.6.1 Finite State-based Approach | 36 |
| 2.6.2 Frame-based Approach | 38 |

| | | |
|--|--|-----------|
| 2.6.3 | Plan-based Approach..... | 40 |
| 2.6.4 | Information State Update Approach..... | 41 |
| 2.7 | DISCUSSION..... | 42 |
| CHAPTER 3: GAME DIALOGUES | | 43 |
| 3.1 | LAYERS OF COMPUTER-BASED GAMES | 43 |
| 3.2 | THE GAME SETTING | 45 |
| 3.2.1 | The Player..... | 46 |
| 3.2.2 | Interactivity and Agency | 47 |
| 3.2.3 | Game Dialogues | 49 |
| 3.3 | LANGUAGES FOR COMMUNICATING GAMEPLAY | 53 |
| 3.3.1 | Gameplay Design Patterns..... | 55 |
| 3.4 | GDPs FOR GAME DIALOGUES..... | 58 |
| 3.4.1 | ELIZA..... | 58 |
| 3.4.2 | Zork | 59 |
| 3.4.3 | Grim Fandango..... | 60 |
| 3.4.4 | The Elder Scrolls III: Morrowind..... | 61 |
| 3.4.5 | The Elder Scrolls IV: Oblivion..... | 62 |
| 3.4.6 | Façade..... | 63 |
| 3.4.7 | Mass Effect..... | 64 |
| 3.5 | A COMPARATIVE ANALYSIS..... | 65 |
| 3.5.1 | Hypothetical Gameplay Design Patterns..... | 66 |
| 3.6 | CONCLUDING REMARKS..... | 68 |
| 3.7 | COLLECTION OF PATTERNS | 70 |
| CHAPTER 4: TECHNOLOGICAL FRAMEWORK..... | | 73 |
| 4.1 | THE DFP FRAMEWORK | 73 |
| 4.2 | VOICEXML..... | 74 |
| 4.2.1 | SRGS and SISR..... | 76 |
| 4.3 | STATE CHART XML (SCXML)..... | 78 |
| 4.3.1 | Hierarchy and History | 79 |
| 4.3.2 | Concurrency..... | 81 |
| 4.3.3 | Broadcast Communication | 83 |
| 4.3.4 | Data Model | 84 |
| 4.3.5 | External Communication..... | 85 |
| 4.4 | SCXML IN THE BIGGER PICTURE | 86 |
| 4.5 | IMPLEMENTING DIALOGUE MANAGEMENT STRATEGIES IN SCXML..... | 88 |
| 4.5.1 | FSMs in SCXML..... | 90 |
| 4.5.2 | Frame-based Approach..... | 91 |
| 4.5.3 | Plan-based Approach..... | 93 |
| 4.5.4 | Information State Update Approach..... | 93 |
| 4.6 | ADVANTAGES OF USING HAREL STATECHARTS | 94 |

| | | |
|--|--|------------|
| 4.6.1 | Visual Representation | 94 |
| 4.6.2 | Statecharts and the Iterative Design Process..... | 96 |
| 4.6.3 | Summary | 96 |
| 4.7 | EXAMPLES OF INTERACTIVE SYSTEMS USING A SIMILAR APPROACH..... | 97 |
| CHAPTER 5: FACE MANAGEMENT FOR NPCS | | 99 |
| 5.1 | FACE MANAGEMENT | 99 |
| 5.1.1 | Threats to Face | 100 |
| 5.1.2 | Brown and Levinson: Politeness Theory | 100 |
| 5.1.3 | Walker, Cahn and Whittaker: Linguistic Style Improvisation..... | 102 |
| 5.1.4 | Prendinger and Ishizuka: Social Filter | 103 |
| 5.2 | INTRODUCING <i>P</i>..... | 103 |
| 5.2.1 | Calculating SD and P | 104 |
| 5.3 | APPLYING <i>P</i> IN A PRACTICAL DIALOGUE..... | 105 |
| 5.3.1 | Activity Analysis..... | 106 |
| 5.3.2 | Mental Model..... | 107 |
| 5.3.3 | The Effects of the Social Filter | 108 |
| 5.3.4 | Parallel Dialogues | 112 |
| 5.4 | SUMMARY AND CONCLUSION | 113 |
| CHAPTER 6: DEAL | | 115 |
| 6.1 | INTRODUCTION TO DEAL..... | 115 |
| 6.1.1 | Framework | 115 |
| 6.2 | TRADE..... | 116 |
| 6.2.1 | The Activity | 117 |
| 6.3 | IMPLEMENTATION IN SCXML | 118 |
| 6.3.1 | Opening..... | 120 |
| 6.3.2 | Trading..... | 121 |
| 6.3.3 | Code Example – Negotiation | 126 |
| 6.4 | DISCUSSION | 128 |
| CHAPTER 7: CASUAL CONVERSATIONS..... | | 129 |
| 7.1 | SMALL TALK | 129 |
| 7.1.1 | Integrating Small Talk with the Opening Phase | 130 |
| 7.1.2 | Giving Small Talk the Same Status as the Practical Conversation..... | 131 |
| 7.1.3 | Giving Practical Conversation Precedence | 133 |
| 7.1.4 | Implementation | 134 |
| 7.2 | GOSSIP | 135 |
| 7.2.1 | Background | 135 |
| 7.2.2 | The Activity | 136 |
| 7.3 | GOSSIP IN FICTIONAL STORIES | 137 |
| 7.4 | A FIRST ATTEMPT TO MODEL GOSSIP | 139 |

| | | |
|--|---|----------------|
| 7.5 | TWO EXPERIMENTS ON GOSSIP CONVERSATIONS | 141 |
| 7.5.1 | Hypotheses..... | 142 |
| 7.6 | EXPERIMENT I: IDENTIFYING GOSSIP TEXT | 142 |
| 7.6.1 | Material and Procedure..... | 142 |
| 7.6.2 | Results | 144 |
| 7.6.3 | Discussion..... | 145 |
| 7.7 | EXPERIMENT II: IDENTIFYING GOSSIP ELEMENTS IN A TEXT | 146 |
| 7.7.1 | Material..... | 146 |
| 7.7.2 | Procedure..... | 147 |
| 7.7.3 | Results | 147 |
| 7.7.4 | Discussion..... | 151 |
| 7.8 | EFFECT OF INTERPERSONAL RELATIONS | 152 |
| 7.8.1 | Conclusion..... | 154 |
| 7.9 | TOWARDS A COMPUTATIONAL MODEL OF GOSSIP..... | 155 |
| 7.9.1 | Determining the Appropriateness for Initiating Gossip | 155 |
| 7.9.2 | Initiating Gossip | 156 |
| 7.9.3 | A Statechart Model for Initiating Gossip | 157 |
| 7.9.4 | Conclusion..... | 159 |
| 7.10 | DISCUSSION..... | 160 |
| CHAPTER 8: CONCLUSIONS AND FUTURE WORK..... | | 163 |
| 8.1.1 | SCXML | 168 |
| REFERENCES..... | | 170 |
| APPENDIX I: WAITER DM..... | | 187 |
| APPENDIX II: WAITER DM RUN – LOG..... | | 200 |
| APPENDIX III: DEAL..... | | 207 |
| APPENDIX IV: MOVIE-GOSSIP | | 221 |

List of Tables

| | |
|--|-----|
| Table 1: A Classification of CAs | 17 |
| Table 2: Example slots in a frame-based dialogue system | 39 |
| Table 3: Gameplay Design Pattern: Information Passing (part I) | 56 |
| Table 4: Gameplay Design Pattern: Information Passing (part II) | 57 |
| Table 5: New Gameplay Design Patterns for Dialogue | 71 |
| Table 6: Gameplay Design Patterns for dialogues found in earlier studies | 72 |
| Table 7: Hypothetical Gameplay Design Patterns for Dialogue | 72 |
| Table 8: Effects of agreeableness personality dimension | 110 |
| Table 9: Effects of extroversion personality dimension | 111 |
| Table 10: A comparison between gossip and the opinion genre | 139 |
| Table 11: Schema for canned gossip responses | 140 |
| Table 12: A preliminary rating of all excerpts. | 143 |
| Table 13: Gossip ratings of all 16 questions sorted by their mean value | 144 |
| Table 14: Inter-coder reliability | 148 |
| Table 15: Relationship between the different elements and gossip. | 148 |
| Table 16: Gossip – third person focus | 148 |
| Table 17: Gossip – substantiating behaviour | 149 |
| Table 18: Gossip – pejorative evaluation | 149 |
| Table 19: Co-occurrences grouped by excerpts | 150 |
| Table 20: Correlation between gossip and each of the three features | 151 |

List of Figures

| | |
|--|-----|
| Figure 1: The waiter's action manager | 5 |
| Figure 2: A detailed view of Serve..... | 6 |
| Figure 3: Waiter's dialogue manager..... | 8 |
| Figure 4: Resolve exchange | 9 |
| Figure 5: Updated action manager | 10 |
| Figure 6: Waiter statechart | 11 |
| Figure 7: Updated dialogue manager | 12 |
| Figure 8: Components of a spoken dialogue system..... | 15 |
| Figure 9: A finite state graph representing a simple practical dialogue..... | 37 |
| Figure 10: Layers of a computer game (Adams, 2010) ©Ernest Adams, 2010. <i>Used by permission</i> | 44 |
| Figure 11: Interaction Model (Adams, 2010) ©Ernest Adams, 2010. <i>Used by permission</i> | 44 |
| Figure 12: Picture of dialogue menu in Grim Fandango (LucasArts, 1998)..... | 49 |
| Figure 13: Picture of dialogue menu in <i>Morrowind</i> (Bethesda Game Studios, 2002)..... | 50 |
| Figure 14: Picture of dialogue wheel in <i>Mass Effect</i> (BioWare, 2008) | 50 |
| Figure 15: Game Statechart..... | 78 |
| Figure 16: Pause-and-resume | 80 |
| Figure 17: Emotion and behaviour..... | 82 |
| Figure 18: Game environment with invoked characters | 87 |
| Figure 19: Conversation module | 89 |
| Figure 20: Emotions manager | 107 |
| Figure 21: Dialogue manager for introvert waiter | 112 |
| Figure 22: Parallel dialogue | 113 |
| Figure 23: DEAL interface..... | 118 |
| Figure 24: Dialogue manager for Shopkeeper in DEAL..... | 120 |
| Figure 25: Opening..... | 121 |

| | |
|---|-----|
| Figure 26: Trading..... | 121 |
| Figure 27: Define object of interest..... | 122 |
| Figure 28: A detailed view of Define object of interest..... | 123 |
| Figure 29: Negotiation..... | 124 |
| Figure 30: Resolve exchange. | 125 |
| Figure 31: A modified Trading state..... | 126 |
| Figure 32 Opening with small talk module..... | 131 |
| Figure 33: Dialogue manager for switching between small talk and practical dialogue..... | 132 |
| Figure 34: Giving the practical dialogue precedence..... | 134 |
| Figure 35: A model of gossip based on the opinion genre..... | 140 |
| Figure 36: Chart illustrating co-occurences grouped by excerpts..... | 149 |
| Figure 37 A statechart model for initiating gossip..... | 158 |

Abbreviations

| | |
|------|-----------------------------------|
| 2D | Two dimensional |
| 3D | Three dimensional |
| ABNF | Augmented Backus-Naur Form |
| AM | Action Manager |
| API | Application Programming Interface |
| ASR | Automatic Speech Recognition |
| BDI | Belief-Desire-Intention |
| CA | Conversational Agent |
| CPU | Central Processing Unit |
| DFP | Data-Flow-Presentation |
| DM | Dialogue Manager |
| ECA | Embodied Conversational Agent |
| DS | Dialogue System |
| FADT | Formal Abstract Design Tool |
| FIA | Form Interpretation Algorithm |
| FSM | Finite State Machine |
| FTA | Face Threatening Action |
| GDP | Game Design Patterns |
| GOAP | Goal Oriented Action Planning |
| GTA | Grand Theft Auto |
| GUI | Graphical User Interface |
| IPA | Intelligent Personal Assistant |
| ISU | Information State Update |
| IVR | Interactive Voice Response |
| LSI | Linguistic Style Improvisation |
| MDA | Mechanics-Dynamics-Aesthetics |

| | |
|-------|--|
| MDP | Markov Decision Processes |
| NL | Natural Language |
| NLI | Natural Language Interaction |
| NLG | Natural Language Generation |
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| NPC | Non-playable Character |
| P | Power |
| PC | Playable Character |
| POMDP | Partially Observable Markov Decision Processes |
| RPG | Role Playing Game |
| SCXML | State Chart XML |
| SD | Social Distance |
| SDS | Spoken Dialogue System |
| SISR | Semantic Interpretation for Speech Recognition |
| SRGS | Speech Recognition Grammar Specification |
| TH | Talking Head |
| TTS | Text-To-Speech |
| UI | User Interface |
| UML | Unified Modelling Language |
| VH | Virtual Human |
| VW | Virtual World |
| W3C | World Wide Web Consortium |
| XML | eXtensible Markup Language |

Chapter 1

Introduction

Once you move away from shooting games, when you are face to face with characters and you are not necessarily blowing their brains out the speech part becomes much more important.

David Braben, head of Frontier Developments¹

Most of the non-playable characters (NPCs) the player encounters in a computer game have only a brief appearance in the player's game life. The main reason for this is that the roles they possess are functionally motivated, for instance as shopkeepers to enable trade, enemies to offer challenges, or helpers of various kinds to support the player's progression through the game (see e.g. Isbister, 2006).² An NPC may also have a dramatic role as part of the narrative, but their repertoire of possible actions and reactions is usually limited in accordance with their functional role and so the function of the NPC often overrides its potential for playing an interesting role figure in the story. Engaging in a conversation with an NPC typically means that you are confronted with a number of pre-scripted dialogue choices appearing on the screen. When selecting one of these choices the NPC responds and a new set of options is generated. Some of the options may be reliant on the relationship your character has developed with the NPC in question, as in *Mass Effect* (BioWare, 2008) and *The Sims*TM (Electronic Arts, 1998-). The dialogue choices may also be used as means for defining and developing the playable character's personality traits as in the role-playing game (RPGs) *Dragon Age II* (BioWare, 2011). The personality traits may in turn affect which dialogue and other gameplay options become available for the player later in the game.

This thesis investigates how to create NPCs that exhibit human-like behaviour and that can engage in natural language conversations with a human player, i.e. basically an embodied conversational agent (ECA) (see for example Cassell, Sullivan, Prevost, & Churchill, 2000b) designed for a game. Traditional ECAs have mainly been designed to handle practical dialogues, i.e. dialogues "focussed on accomplishing a concrete task" (Allen, Byron, Dzikovska, Ferguson, Galescu, & Stent, 2001, p. 29), and they are typically placed in a realistic setting. An NPC,

¹ In an article written by Mark Ward (2002): Fast forward to the future of games, BBC News, Aug 30. Available at: <<http://news.bbc.co.uk/2/hi/technology/2223428.stm>>

² Chapter 2 provides a more detailed overview of these roles.

on the other hand, operates within a given dramatic role in a fictional game world and is thus also a carrier of the story. This means that the requirements for a game character may differ from a traditional ECA. Therefore, this thesis also explores the design space and identifies requirements for a game dialogue system.

1.1 Natural Language Interaction in Games

While the traditional ECA typically substitutes a human in a real-life situation, an NPC plays a dramatic role in a story, which implies that NPCs “[...] do not have to be realistic—rather, they have to behave appropriately given the context of their roles in the fictional world” (Zubek, 2005, p. 22), i.e. they have to be believable. According to Hayes-Roth and Doyle (1998, p. 202) a believable character should be a “good conversational partner” that tries to “get the message” rather than strive for achieving perfect understanding, and be able to “express themselves” rather than retrieve correct responses from a database. These requirements assume that the dialogue understanding component of the dialogue system is capable of identifying the speaker’s underlying intention and that the character has the ability to generate an answer that fits the current context – independently of whether it is the best or most correct answer. However, this is very difficult to achieve in a dialogue system. From another point of view, this also suggests that a character may have flaws that typically are unacceptable for traditional conversational agents that solve real-world tasks, for instance being wrong or ignoring misunderstandings. It also suggests that these characters should be more socially oriented, i.e. care more about the interpersonal relationship than the accuracy of their statements. In order for this to be possible, these characters need to be equipped with social skills.

Natural language interaction has been used in games before. The first examples of natural language interactions in game-like environments are found in early multiplayer virtual worlds, so called multi-user dungeons (MUDs) that were developed by university students with a particular interest in natural language processing (see for example Bartle, 2003). Other examples include text-based adventure games, such as *Zork* (Infocom, 1980) and *The Hitchhikers guide to the Galaxy* (Infocom, Inc., 1984) that accepted natural language input as the only means for interacting with the game. And consequently, interacting with the game components, navigating in the world, talking to the other game characters, and performing meta-commands, were all conducted in the same interface. The input language was restricted so a part of the challenge was to figure out what could be said and how to say it in order to be understood. Today, games have in many respects abandoned text-based interaction in favour of direct manipulation

of graphical user interfaces (GUI), but speech-based interaction has become more and more common as a complement or alternative to the GUI. Most games have support for voice-based interaction, mainly for issuing commands to the system, and there are a number of middleware products that provide Software Development Kits for speech recognition specifically designed for game engines.

Despite these new technological improvements, game dialogues are still mainly constructed as branching dialogue trees, presenting the player with a limited number of options that each unfolds a new branch in the tree. This is mainly due to the complexity of natural language and the difficulty in creating a system that can understand and produce unrestricted natural language conversations, or as Adams (2010, p. 185) put it: “Game designers would like to be able to include natural language in games without trying to solve a decades-old research problem”. Also, a game cannot afford to cause player frustration due to poor interpretation of the input.

1.2 Choice of Technology

An aim of this thesis is to present a work that is directly accessible for the gaming industry. It was therefore an early decision to use standard web technology for building voice applications since standards are free, accessible, stable, distributable (i.e., it works on different browsers and has backward and forward compatibility), can easily be validated, and provide consistency. To use standard *web* technology may furthermore be motivated by the fact that games played through a web-browser have increased dramatically, mainly due to Facebook, which declares that more than 250 million people play Facebook games every month.³ Also, XML is extensively used for storing data in game engines and therefore most game engines already have support for parsing XML documents.

The World Wide Web Consortium⁴ (W3C) is an organization that develops standards for the web among which VoiceXML is the most commonly used standards for building dialogue systems. In 2006, the W3C introduced the Data Flow Presentation (DFP) framework⁵ for keeping the components that control the data and flow of a multimodal application separate from the component(s) that communicate with the end user, for example a VoiceXML application. W3C also introduced State Chart XML (SCXML) as one of the languages for specifying the

³ At Game Developers Conference (GDC) 2013, Facebook announces that over 250 million people play Facebook games every month. See e.g. <<http://www.theverge.com/2013/3/26/4149838/facebook-says-over-250-million-playing-games-each-month>>

⁴ The World Wide Web Consortium (W3C) is an international organization that develops technical specifications and guidelines that “ensure the long-term growth of the Web”, see <<http://www.w3.org/Consortium/mission.html>>

⁵ <<http://www.w3.org/Voice/2006/DFP>>

flow of an application. SCXML can be described as an attempt to render Harel statecharts (Harel, 1987) in XML. In its simplest form, a statechart is just a finite state machine, where state transitions are triggered by events appearing in an event queue. This thesis investigates whether standard web technology in general and SCXML in particular is expressive enough to implement game dialogues.

A recent trend in dialogue management design is to use machine learning techniques and stochastic approaches for identifying and implementing dialogue strategies (see for example Schatzmann, Weilhammer, Stuttle, & Young, 2006; Pieraccini, Suendermann, Dayanidhi, & Liscombe, 2009; Paek & Pieraccini, 2008; Young et al., 2010). These approaches require that an extensive amount of data is collected and processed, which is a time consuming task even if this could be done to some extent by means of user simulations and unsupervised learning (see for example Schatzmann, Weilhammer, Stuttle, & Young, 2006). The standards used in this thesis do not currently support stochastic approaches and so the examples given here have been constructed using hand-written rules.

1.3 A Simple Game Scenario

The dialogue models presented in this thesis are visualized using Harel statecharts (Harel, 1987) and as an introduction to statechart notation, this section will present a simple game scenario. As has previously been mentioned, many game characters are functionally motivated and the dialogues they are capable of engaging in are typically practical. It is also well known that practical dialogues have been successfully implemented in commercial applications, due to their limited domain and clear task (see for example Allen et al., 2001). For these reasons it seems reasonable to start with a game scenario, in which an NPC can engage in a practical dialogue with the player. This dialogue will also serve as a base model for the extensions that will be presented later in this thesis.

The setting is a restaurant or tavern, in which the playable character is a customer that orders something from an NPC in the waiter role. Initially, the system-controlled waiter will automatically serve the customer a random dish from the counter, just like the waiter character in the game *Café World* (Zynga Inc., 2009), but a conversational module is then added so that the player can order something from the menu using natural language.

1.3.1 The Waiter Character

The waiter switches between two basic behaviours: idle and serve. Idle is the default behaviour and describes a waiter animated with a neutral standing pose while waiting for a new customer to arrive. As soon as a customer has been detected at a table, the waiter changes action from being idle to serve. Apart from

that, no actual interaction takes place between the waiter and the customer. Figure 1, below, illustrates the waiter’s action manager, i.e. the mechanism that controls the waiter’s behaviour. Just like ordinary finite-state machines, statecharts have a graphical notation – for “tapping the potential of high bandwidth spatial intelligence, as opposed to lexical intelligence used with textual information” (Samek, 2002). The rounded boxes denote states and the arrows between the states denote possible transitions. The black dot represents an initial pseudostate. Labels connected to transitions represent events (specified by “On” in the models) and/or conditions (marked “If”) that will trigger the transition. In Figure 1 “done.Serve” is an example of a system-generated event that is raised when the state `Serve` has reached its final state. “EnterCustomer”, on the other hand, is user-generated as it is raised when the PC enters the restaurant.

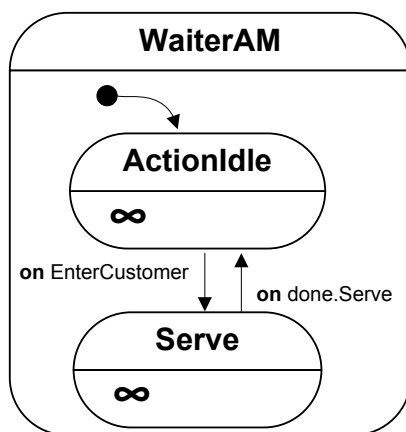


Figure 1: The waiter's action manager

One extension to finite state machines that Harel introduced was hierarchical states, i.e. states that contain another statechart that in turn may contain another statechart down to an arbitrary depth. An infinity sign inside a state box indicates that we are dealing with a hierarchical (or compound) state with hidden sub-states. In order to view the content of such a state in detail, Harel statecharts offer a zoom-in function. A state that lacks an infinity sign is atomic, which means that it does not contain any sub-states.

As can be noted in the illustration, the waiter’s action manager (AM), represented by the state `WaiterAM`, is an example of a hierarchical state as it contains two states, `ActionIdle` and `Serve`, each corresponding to the waiter’s basic behaviours mentioned earlier. The infinity sign inside each of these states also indicates that they in turn (may) contain sub-states. For example, to complete the `serve` action the waiter has to follow a specific plan: go to the counter, pick up a plate, return to the table and finally serve the plate. `Serve` therefore needs to be decomposed into states that correspond to each of these sub-actions and the sub-actions need to be sequentially ordered: `GoToCounter`, `PickUpPlate`,

GoToTable, and finally, ServePlate. Figure 2, below, illustrates Serve with its child states:

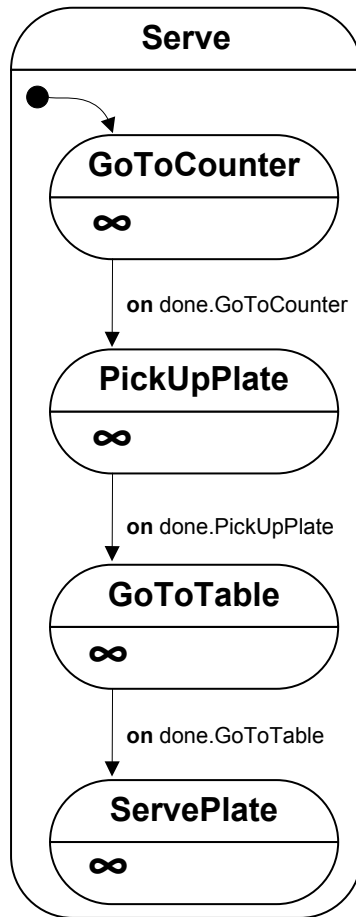


Figure 2: A detailed view of Serve

1.3.2 Extending the Model with a Dialogue Manager

So far, the waiter has been modelled as a character with two available behaviours to switch between, idle and serve, each represented as child states of the actions manager (AM). The next step is to give the waiter the ability to engage in a dialogue with the player (character) to take the order and receive payment.

In the simplest and most straightforward case, the waiter greets the customer and takes the order. The customer responds by ordering something that is available on the menu. Hence, a successful dialogue between the customer and the waiter could be as follows:

W1: Hello, may I take your order?
C1: I would like bacon and eggs please.
W2: Bacon and eggs. Coming right up!
W3: [Waiter walks to the counter and picks up a plate of bacon and eggs]
W4: Bacon and eggs, that'll be 4 euros please.
C2: 4 euros, ok [Customer hands over 4 euros]
W5: Thank you!

Dialogue 1: A simple dialogue between the waiter and a customer

The waiter establishes contact by greeting the customer in W1. The waiter then takes the order and confirms it (W1, C1, and W2) before leaving the table to pick up the ordered dish (W3). When the waiter returns to the table to serve the dish, payment is requested (W4). The customer confirms the price and hands over the money (C2). The waiter receives the money and ends the interaction in W5.

In the current example the task is to take an order from the customer, serve it and receive payment. The current game has a limited setting that in its entirety constitutes the domain of the dialogue as well. In other games, the dialogue domain may only constitute a particular gameplay mode, which means that it takes place in a different interface from the rest of the game. Since we may use analogies from the real world, the concepts associated with the domain and the expected behaviours are already familiar to the player.

Figure 3, below, illustrates the flow of the dialogue presented above. An event containing the prefix “ca”, such as “ca.order”, is generated when the player says something to the NPC (ca stands for “communicative act”). Examples of such communicative acts are “greet” and “order”. An event can also contain a data payload, which can be accessed through the parameter “_event.data”. Important to note is that even though the dialogue manager is modelled by means of Harel statecharts, it is not necessary to implement it in SCXML.

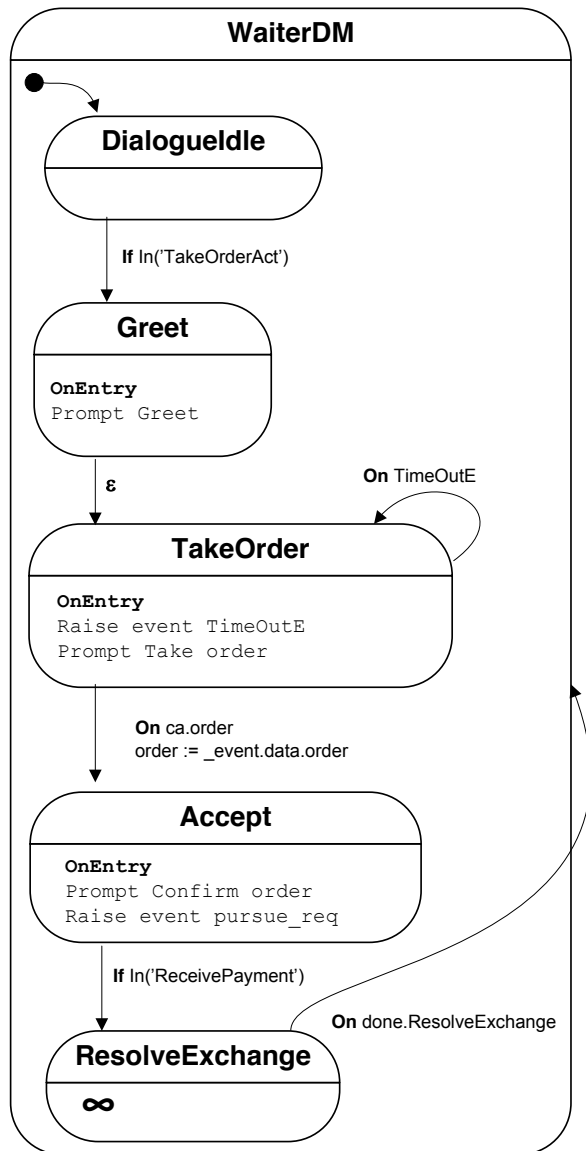


Figure 3: Waiter's dialogue manager

While waiting for a customer to arrive and take a seat, the waiter is in the state `DialogueIdle`. As illustrated in the dialogue model, a transition to `Greet` is taken when the condition `In('TakeOrderAct')` is fulfilled. The “`In()`”-predicate states that a specific state, in this case `TakeOrderAct`, must be active in order for the condition to return “true”. The state `TakeOrderAct` will be presented further below. Harel statecharts allow actions to be carried out along transitions or within a state, either upon entering the state or when the state is left (Harel, 1987), so when the `Greet` state is entered, the system outputs a prompt in which the waiter greets the customer. Usually, a transition is activated by a triggering event or when a certain condition holds. However, in some cases it is desirable to transition to the next state as soon as the state’s on-entry and/or on-exit scripts have been executed. In this case it is possible to use so called empty

(ϵ) transitions, which are transitions that do not specify any event or condition. In the model depicted in Figure 3, an empty transition connects `Greet` with `TakeOrder`, which is activated as soon as the waiter has greeted the customer (as specified in the on-entry script). Upon entering `TakeOrder`, a new prompt is generated, this time to take the order. Also, a time out event is raised that will fire if the customer does not respond to the waiter’s request. If the user instead answers within the set time, the time-out event will be aborted and a state change may occur. If not, the waiter can repeat the request. These first steps correspond to line W1 of Dialogue 1 above.

If the customer orders something that is available on the counter, a transition to `Accept` is conducted. The waiter confirms the order upon entering the state (line W2 of the dialogue) and the event “pursue_req” is raised, which triggers the waiter to go the counter and pick up the requested dish (line W3).

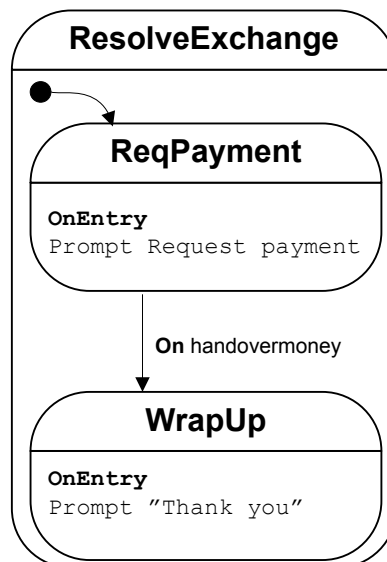


Figure 4: Resolve exchange

Returning with the dish and serving it, all that is left for the waiter to do is take the check and resolve the exchange, which is pursued in the compound state `ResolveExchange` (see Figure 4, above) `ResolveExchange` consists of two states: `ReqPayment` and `WrapUp`. `ReqPayment` corresponds to line W4 of Dialogue 1, above. When the customer hands over the money (line C2), a transition to the final state `WrapUp` is triggered (corresponding to line W5).

As has already been indicated, the waiter’s AM needs to be adjusted to handle the interaction with the customer. The dialogue can only be initiated after the waiter has detected a new customer, but before the serve action is initiated. Furthermore, the waiter needs to approach the customer before an order can be taken. This means that the waiter’s action manager needs two additional steps: “go to table”

and “take order”. Furthermore, when the waiter has taken the order and is about to carry out the next step, to serve, the dialogue will either have to be put on hold or be terminated. In this example, the waiter will return with the dish and request payment and so the waiter’s communicative acts must in some way be synchronized with the waiter’s other actions.

Figure 5 below, illustrates the waiter’s updated actions manager, which apart from the inclusion of the state `TakeOrderAct` now also contains the state `ReceivePayment` to enable the money transaction.

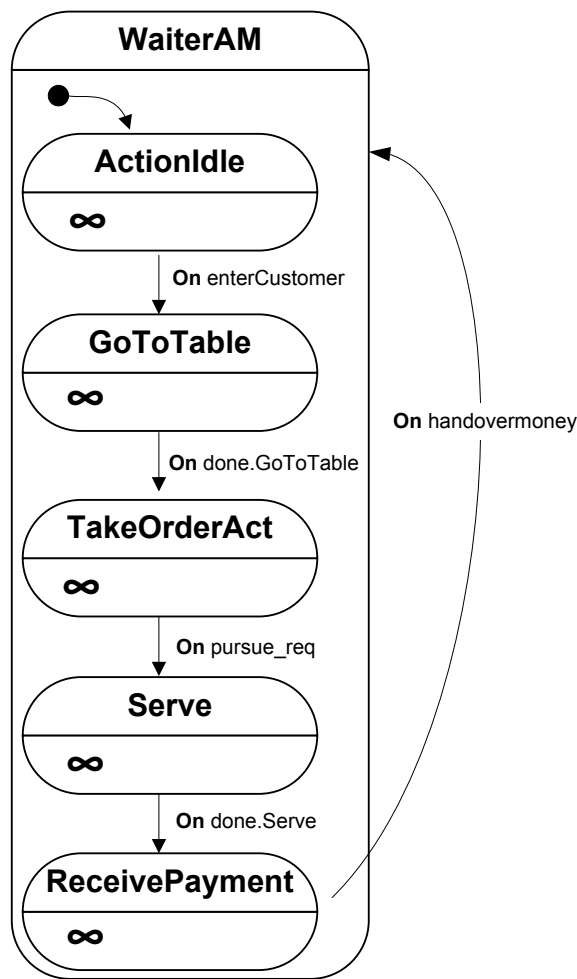


Figure 5: Updated action manager

The default start state is still `ActionIdle`, since the waiter is assumed to idle until an event is detected that triggers a transition to the newly added `GoToTable`, which is when a new customer has taken a seat. When the table has been reached, the waiter’s next move is to take the order and so yet another state has been added, `TakeOrderAct`, which is synchronized with the dialogue

manager, such that a transition is triggered from `DialogueIdle` to `Greet` when `WaiterAM` is in the state `TakeOrderAct`. This corresponds to the condition `In('TakeOrderAct')` in `WaiterDM` (see Figure 3).

As soon as an acceptable order has been placed, transition to `Serve` within `WaiterAM` is triggered. The dialogue manager is put on hold until `Serve` reaches its final state and a transition to `ReceivePayment` has been conducted, which is synchronized with the state `ResolveExchange` in the dialogue manager. Now, currently the AM and the DM are said to be synchronized, but in fact we have only created two separate statecharts that represent the AM and DM, respectively. However, statecharts also support the possibility to have states running in parallel, independently from each other, but loosely coupled. Hence, in order to synchronize the waiter's behaviour, the AM and DM must run concurrently, as illustrated in Figure 6, below:

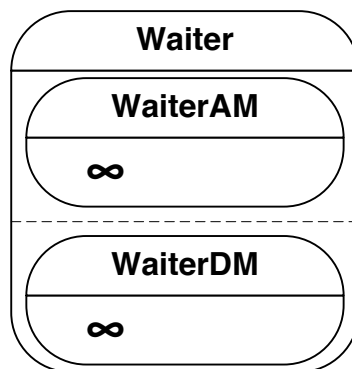


Figure 6: Waiter statechart

The current dialogue model does not account for potential failures, such as when the customer orders something that is not available on the counter. One way to deal with this circumstance is that only the dishes that are accessible on the counters at the time of the order are acceptable orders. The customer must then be notified that the requested dish cannot be served, which means that two more states must be added – one in which the waiter confirms and accepts the order and another one in which the order is rejected. When an ordered is accepted the waiter may go to the counter and pick up the dish, but when the order is rejected, the waiter has to take a new order. If the customer instead is silent the waiter will repeat the request when a certain time period has passed. Figure 7, below, illustrates the extended dialogue manager.

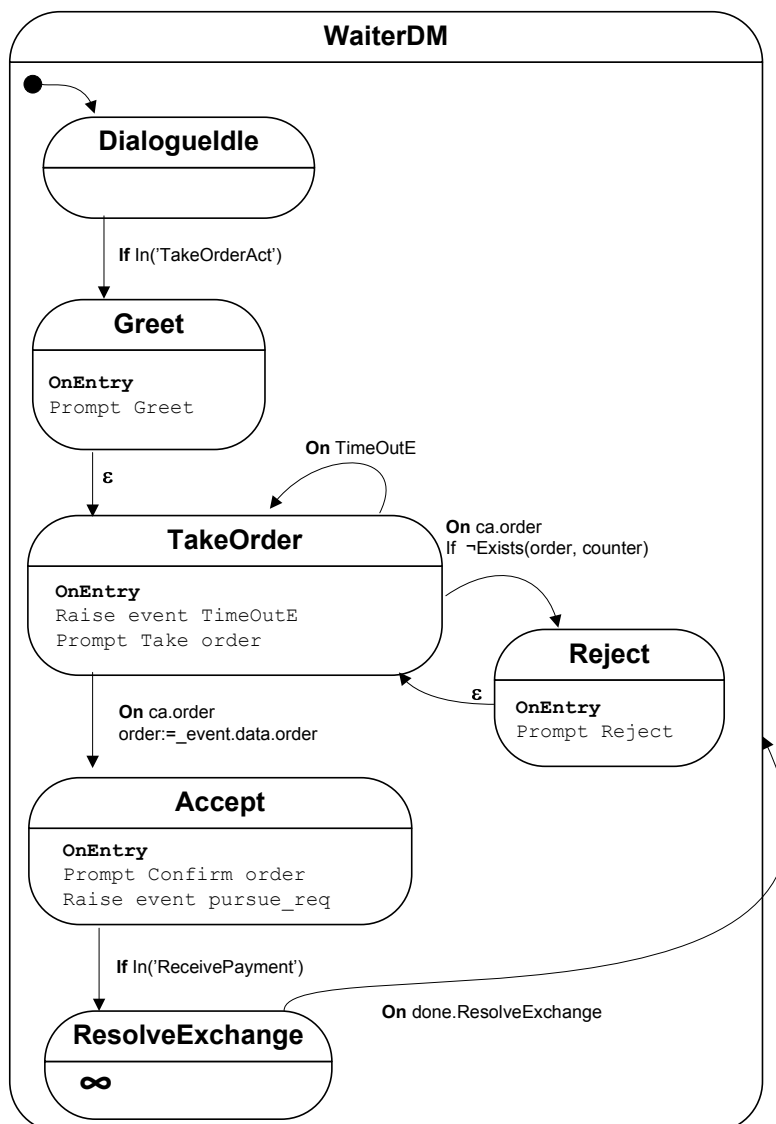


Figure 7: Updated dialogue manager

So far we have shown that statecharts are able to manage the dialogue as well as the waiter’s behaviour, but most importantly the synchronization between them.

1.4 The Structure of the Thesis

Having established that the main theme of this thesis is to investigate how we can create NPCs that are able to engage in interesting and socially meaningful dialogues using standard technology we now turn to the structure of the thesis.

Chapter 2 introduces conversational agents (CAs), in particular Embodied CAs (ECAs) and NPCs. Potential differences between practical dialogue systems and game dialogue systems that may have consequences for the design are also discussed. A number of dialogue management tasks for conversational NPCs are

suggested followed by a review of the most common strategies for managing dialogues in practical dialogue systems.

In chapter 3 we investigate how games can benefit from using natural language interaction and explore the design space between game worlds and dialogue systems. We will primarily revolve around actual and potential uses of a game dialogue system expressed by means of gameplay design patterns (Björk & Holopainen, 2005). A goal of the chapter is to identify how novel gameplay can be created by means of natural language interaction. The chapter is partly based on (Brusk & Björk, 2009).

In chapter 4, VoiceXML and in particular SCXML are presented as well as their position in the Data Flow Presentation (DFP) framework in which they play important roles. Harel statecharts (Harel, 1987) will be introduced as well since it constitutes the semantics of SCXML, but also because it is a useful tool for describing the flow of an event-based application, such as a game or a dialogue. We will use Harel statecharts throughout the thesis to illustrate and specify our game and dialogue models. Parts of this chapter have previously been published in (Brusk & Lager, 2008).

In chapter 5 we exemplify how our waiter character presented earlier may be extended with social awareness. We take Brown & Levinson's (1987) theory of politeness as a point of departure and present face management strategies and behaviours based on the interpersonal relationship and the character's mental state. The chapter is partly based on (Brusk, 2008) and (Brusk, 2010).

Chapter 6 introduces DEAL, a serious games project for language learning. We present a dialogue manager modelled using statecharts and implemented in SCXML, aimed at describing a shopkeeper at a flea market. By adding a module for negotiation we show by example how a shopkeeper in a game can become more interesting to interact with. The original version of the chapter has previously been published in (Brusk, Lager, Hjalmarsson, & Wik, 2007).

Chapter 7 is devoted to gossip conversation. We start by giving a background to gossip and the structure of gossip, followed by a presentation of two experiments conducted at Institute for Creative Technology, University of Southern California. The first experiment aimed at investigating how people intuitively perceive and define gossip while the second aimed at investigating whether people could agree upon a given definition of gossip, and investigate why if not. The results from the experiments in combination with earlier studies of gossip made by others formed a basis for a first model for initiating gossip, presented in the third part of the chapter. The chapter is based on the following articles (Brusk, 2009; Brusk, Artstein, & Traum, 2010; Brusk, 2010).

In Chapter 8 a final discussion, conclusions and further research is presented.

Chapter 2

Conversational Agents

A Conversational Agent (CA) is a program that can communicate with humans using natural language speech or text. Most CAs are designed for engaging in practical dialogues over the telephone – so-called interactive voice response systems (IVR-systems). Examples of tasks these agents are capable of performing are travel arrangements, bank transactions and providing weather reports. There are also embodied CAs (ECAs) of various complexities developed for multimodal communication.

This chapter starts with an overview of CAs, including a rough classification. Conversational NPCs will be given particular attention and the potential uses of natural language interaction in games will be discussed. In later sections, dialogue management tasks and various strategies for dialogue control will be presented as well as possible design differences between game dialogue systems and practical dialogue systems.

In the next chapter games and game dialogues will be presented in more detail.

2.1 Classification of CAs

There are a number of different types of CAs, all designed with a particular usage in mind. For the purpose of this thesis, CAs will be categorized based on the type of dialogue they can engage in and what type of dialogue system they therefore require, whether they are typically used in fictional or realistic settings, and the most common platform that they use.

2.1.1 Dialogue Systems

Dialogue systems can be either text- or voice-based and a spoken dialogue system (SDS) typically consists of the following components (e.g. Jurafsky & Martin, 2000; Clark, Fox, & Lappin, 2010) (see Figure 8): an automatic speech recognizer (ASR) (assuming speech input is available), which transforms an acoustic signal into text; a natural language understanding (NLU) component, involving a syntactic and semantic parsing of the input to a formal representation;

a dialogue manager (DM), controlling the flow of the dialogue; a language generation component, determining the surface structure of the utterance; and a text-to-speech component (TTS), transforming the surface utterance into a speech output.

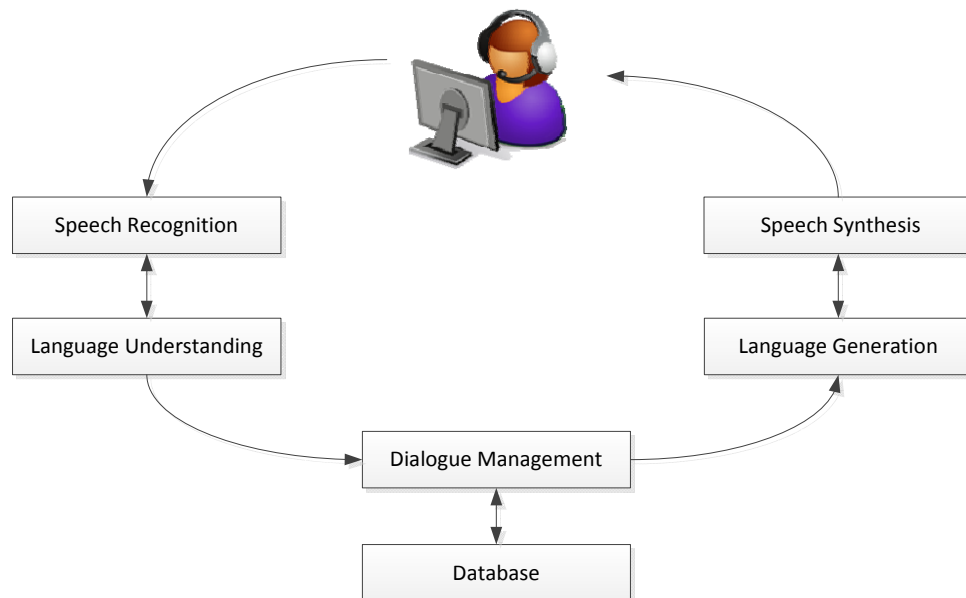


Figure 8: Components of a spoken dialogue system

A text-based dialogue system naturally lacks the ASR component as well as the TTS component. Some systems however provide a TTS component even though they do not accept speech input. A dialogue manager often needs to communicate with a back-end system, such as a database or knowledge base, in order to retrieve or update information.

There are two main types of DSs: command-based systems and conversational systems (Skantze, 2007; Pieraccini & Huerta, 2005; Larsson, 2005). A command-based system is typically used for practical dialogues in commercial applications and so the usability aspect is central (see for example Pieraccini & Huerta, 2005). Therefore, it requires good speech recognition (when speech is accepted as input) and language understanding, but less investment in dialogue management and response generation. Conversational systems, on the other hand, aim to simulate human language use and the challenge is therefore to figure out “how to model everything that people may say” (Skantze, 2007, p. 12).

In games, each of these two types may be of relevance. A command-based system may for example be used for controlling the game as a complement or alternative to the mouse and keyboard, i.e. for voice-based game interfaces. Conversational systems are however more relevant for engaging in dialogue with

the in-game characters. This issue will be further discussed in section 2.2.1, below.

2.1.2 Platform and Setting

In this rough classification, CAs are divided into four main types: Interactive Voice Responses (IVRs), Intelligent Personal Assistants (IPAs), chatbots and Embodied Conversational Agents (ECAs). It should be noted that the CAs have been classified according to the most common usage, and so there will be possibly several examples diverging from the classification. The main purpose is however to identify the CAs that are of main interest for this thesis and how they differ from other types of CAs.

IVRs are automated telephone systems that allow humans to interact using voice and/or touch-tone input. Apart from the practical uses mentioned earlier, IVRs have been extensively used for conducting surveys, in particular surveys of more sensitive natures (see for example Corkrey & Parkinson, 2002). IVRs differ from the other CAs in that they are mainly command-based.

A human user can engage in social chat with chatbots, such as *A.L.I.C.E.*⁶, by means of text input and text or speech output. Chatbots use pattern-matching techniques to parse the user's utterance and generate answers by selecting a response from a collection of pre-scripted phrases. This means that chatbots do not need to create an understanding of the user's input, but by having an extensive set of templates, the chatbot may appear to be able to maintain a long-lasting conversation. In that sense it could be argued that they make use of a conversational dialogue system. The ultimate goal for a chatbot is also to be taken as a human, i.e. to pass the Turing test, and every year the most human-like computer program is awarded the Loebner prize.⁷ *A.L.I.C.E.* has received the award no less than three times (2000, 2001, and 2004). The first and probably the most famous chatbot is *ELIZA* (Weizenbaum, 1966), which will be presented in more detail and analysed in section 3.4.1.

IPAs, such as Apple's *Siri* and Samsung's *S Voice*, have elements of both IVRs as well as chatbots. These agents assist the human user with various tasks and interact using natural language. They are specifically designed for smart phones and make use of the standard applications already available in phone, such as GPS, calendar, messages, music player, *etc.*, and are able to access information from a variety of online sources and combine this information to respond to the user's request – e.g. “are there any Japanese restaurants in this area open now?”. One could say that IPAs use a conversational dialogue system for managing practical dialogues. Different from traditional IVRs their domain is neither static

⁶< <http://alice.pandorabots.com/>>

⁷< <http://www.loebner.net/Prizef/loebner-prize.html> >

nor defined in beforehand; instead, the information they provide is reliant on the information that can be accessed through the channels mentioned above.

While more and more services are offered online or in shared public spaces, an increased interest in giving them a face and/or body has arisen that enables them to communicate “face-to-face” as well as to use non-verbal behaviours such as gestures and facial expressions in addition to speech. An ECA can range from being a two-dimensional agent with a limited set of facial expression to advanced 3-D characters equipped with a complete set of facial expressions, lip-sync and body movements. ECAs can be further divided into virtual humans (VH), talking heads (TH) and conversational NPCs, and they all share the goal of having a conversational system, but differ in either the setting or platform, or both. In the following section ECAs will be given more attention as they are the main targets of this thesis, but first a table listing the differences between the CAs is presented.

| CA | DS | Setting | Platform | Dialogue type |
|---------|----------------|---------|---------------------|------------------|
| IVR | Command | Reality | Telephone | Practical |
| Chatbot | Conversational | Reality | Web interface | Social chat |
| IPA | Conversational | Reality | Smart phone app | Practical/expert |
| TH | Conversational | Reality | Offline/web interf. | Practical |
| VH | Conversational | Reality | Virtual world (VW) | Practical/social |
| NPC | Conversational | Fiction | Offline/web/VW | Practical/social |

Table 1: A Classification of CAs

It should also be noted that the use of the word “platform” may be a bit misleading, but in the absence of a more appropriate word, platform will here stand for the environment within or through which the interaction is taking place. To further clarify, a virtual world (VW) is a persistent online community capable of handling thousands of simultaneous users, such as Second life (Linden Lab, 2003) and World of Warcraft (Blizzard Entertainment, 2005), but here we accept a broader definition that includes other types of 3D virtual environments as well.

Moreover, several of the CAs are claimed to use a conversational dialogue system, but the fact is that this is so hard to achieve that it should be regarded as the ultimate goal for these agents rather than that they actually fulfil this requirement currently.⁸ Also, most CAs engage in practical dialogues, but some

⁸ Most attempts to model human behaviour in general fail when it comes to the linguistic capabilities. In android science, for example, artificial humans are developed with appearances and behaviours that are highly anthropomorphized. However, the androids lack the ability to perform long-term conversations, and according to Ishiguro & Nishio (2007) this is considered to

also attempt to engage in more socially oriented dialogues even if these dialogues subordinate the task.

2.1.3 Examples of ECAs

Ordinarily, ECAs engage in practical dialogues and may be used as tourist guides, as for example *Waxholm* (Carlsson & Granström, 1996), or museum guides, such as *ADA* and *GRACE* (Swartout et al., 2010), installed at the Museum of Science in Boston, USA. They are also popular as instructors and educators (see e.g. Cassell et al., 2000b; Wik, 2011). The ECA *Rea* (e.g. Cassell, Bickmore, Billingham, Campbell, Chang, Vilhjálmsón & Yan, 1999) takes the role of a virtual real-estate agent that “interacts with users to determine their needs, show them around virtual properties, and attempts to sell them a house” (Cassell, Bickmore, Campbell, Vilhjálmsón & Yan, 2000a). *The August system* (Gustafson, Lindberg, & Lundeberg, 1999) uses a talking head modelled after the deceased Swedish author August Strindberg. Despite its visual appearance and name, it makes no attempt to simulate the author. Rather, the agent was used for studying how inexperienced users would communicate with a spoken dialogue system covering several domains, in this case information about Stockholm and The Royal Institute of Technology, as well as some basic knowledge about the author himself. Another author that has gone through a virtual resurrection is the Danish fairy-tale author Hans Christian Andersen. The ECA was developed within the NICE-project (e.g. Bernsen & Dybkjær, 2005) and could talk to the user about some of his works, his life, and objects in the environment, among other things. ECAs are also used as virtual patients, allowing medical students to practice medicine before actually meeting real patients (see e.g. Kenny et al., 2007).

The most complex ECAs, the virtual humans, are equipped with personality, emotions, and the ability to learn and adapt. The goal for these agents is not necessarily to convince someone that they are actually human, but to “serve as competent role-players to allow people to have a useful interactive experience” (Traum, Swartout, Gratch, & Marsella, 2008, p. 46).

2.2 NPCs

NPCs are often used to drive the story forward by informing the player (character) about the world, the characters, conflicts, places and so on, in order to

be the ultimate goal in robotics. On the other hand they have developed an android with “teleoperation” (remote control) functionality allowing the android to (appear to) keep up a long lasting conversation when it is in fact the human behind the scene that controls the conversational behaviour.

motivate the player to perform the intended actions for game and story progression. They may also constitute the obstacles the player must get past in order to succeed in reaching a specific goal. Just like other ECAs, NPCs are constrained by their role in the game and so is the player's possibility to interact with them. A *shopkeeper's* main function is for example to sell items to the playable character. In addition, such a character may also be able to provide game hints and selected information about other NPCs, events and the like that exist in the surrounding. A *mentor*, on the other hand, instructs the player and hands out quests but cannot sell items. The functional role assigned to an NPC thus determines whether it can engage in dialogues and, if so, what the player can talk to the NPC about.

2.2.1 Natural Language Interaction in Games

There are several ways in which natural language dialogue may come into play in games. Assuming the commonly made distinction between game (G), player (P), playable character (PC) and NPC, and stretching the notion of dialogue somewhat, we may distinguish between:

- (1) **P in dialogue with G:** Games may be 'voice controlled'. In games that require many actions to be performed simultaneously, voice commands can release the cognitive load. Voice interaction can be particularly suitable for games played on video game consoles since their controls only offer a few input combinations as opposed to the keyboard. *Tom Clancy's Endwar* (Ubisoft, 2008) is an example of a game that (according to the developers) can be completed using only voice control.
- (2) **P in dialogue with PC:** Player is directing their PC using dialogue. In text-based adventure games and Multi-User Dungeons (MUDs), typed commands are the sole way to interact with the game, meaning that no difference is made between controlling the game and engaging in P(C) - NPC dialogues.
- (3) **P in dialogue with P:** Player talking to player, using (voice-based) chat. In multiplayer online games, chat is often augmented with emotes, i.e. reserved actions (for example /yawn or /wave) that can be expressed through the avatar.
- (4) **NPC in dialogue with NPC:** The use of natural language for commenting on the states and the events of a game, such as the commentators talking to each other in FIFA (EA Sports, 1993-2012). This type of dialogue only involves the player as an audience (cf. movies). In some games, for example *Skryrim* (Bethesda Game Studios, 2011), the NPCs can talk to each other under certain

circumstances, which give the player the opportunity to perform some eavesdropping.

- (5) **P in dialogue with NPC:** For the purpose of letting NPCs provide the player with background story, quests and directions for progressing the game, but also in order to uphold ‘social relationships’ with NPCs. Dialogues will thus sometimes be task oriented, sometimes of a more socially motivated kind. This type of dialogue is what is typically meant when one discusses “game dialogues” and occurs first and foremost in story-based games, such as adventure games and role-playing games as a tool for conveying the story. A notable exception is *The Sims*TM (Electronic Arts, 1998), since it makes heavy use of dialogue even though the story is player-created. This means that no pre-authored content is needed; instead, dialogue is used to manipulate the social relationship with other characters. The player can choose from a number of communicative actions presented in a menu. The selection results in either a physical act visible to the player or through an icon symbolising the meaning displayed in a speech bubble over the speaker’s head, combined with speech. However, the character speaks in “simlish”, a fictional language that sounds real and when combined with the other visual cues, meaning is created that can be perceived and interpreted by the player.

The first dialogue type (1) is a typical example of a command-based system. It requires an ASR and a simple NLU component, but since it only has to be able to understand and respond to simple commands, a simple DM capable of mapping an input to a specific output is sufficient. The system cannot understand complex sentences nor can the player engage in natural language conversations with the game characters and so only a limited grammar specification covering predefined commands is necessary. The second pattern (2) is an extension of (1) and may therefore involve parsing more complex phrases that in some respect serve as commands just like in (1). Even so, its function is sufficiently supported by a command-based system. The game character/system is not expected to respond to the player’s command verbally, but may act upon the player’s commands in other ways. The player may for instance direct the character/game system by giving orders in terms of for example a verb phrase “open box” and “go north”, but as in the earlier mentioned *Zork* (Infocom, Inc., 1980), dialogues with other NPCs are conducted through the same interface. Player to player conversations (3), for example “chats”, are typical for multiplayer games. Unless the player expresses emotes, the system doesn’t have to parse the players’ contribution to the conversation. Conversations between NPCs in a game (4) are pre-scripted and reflect upon the events of the game, such as when the player waits too long to strike the ball in golf or when a team scores in football. (3) and (4) fall outside

the division between command-based and conversational dialogue systems, as they do not require any natural language processing – (3) because they are examples of real human conversations as opposed to simulated ones, and (4) because the human is merely the audience.

The last pattern for natural language interaction in games (5) corresponds to simulated conversations between the P(C) and an NPC and requires therefore a conversational dialogue system. As for ordinary face-to-face conversations, these conversations typically have a specific purpose or goal that can only be achieved through interaction, thus presupposing some form of cooperation. This is the only example that requires a conversational system. It is also the focus of this thesis.

2.2.2 NPC Roles

NPCs are thus used for various purposes in games, functional, social, as well as dramatic. Isbister's (2006, pp. 230-250) taxonomy of NPCs, for example, is based on those personality dimensions she considers determine their social behaviour: agreeableness (along the friendly-unfriendly scale) and dominance (from dominant to submissive). Below, we present the kind of NPCs from her taxonomy that the player (or PC) is likely to engage in dialogue with.

Friendly roles

Minions are friendly NPCs and the least dominant. Their role is to help and assist the PC in accomplishing his or her goals without ever challenging his or her authority. The most common player interaction with a minion is to give orders. A slightly less submissive is the *Rescuee*, an NPC in the need of the player's help. Different from a *Minion*, a *Rescuee* may develop into a new role after being rescued, for instance become the player's *Ally* or a *Minion*. A *Sidekick* accompanies the PC but is not as submissive as the *Minion* or *Rescuee*. Rather, they are more of a helper character that gives hints to the player and comments on the player's actions. The PC's relationship to the *Sidekick* may become deeper if the design of it allows so. An *Ally*, on the other hand, is on a level with the PC and assists the PC more actively, such as supporting the PC in fights. When a *Sidekick* follows the PC throughout most parts of the game, an *Ally* may appear briefly under a specific period in the game. Therefore, it is unlikely that the player forms any stronger attachment to an *Ally* as it potentially could with a *Sidekick*. As players sometimes are in need of guidance, specific NPCs may be designed to serve that need. *Guides* are typically accessible throughout the game to provide advice from the game's help system when needed. A *Mentor* also serves as a helper, but has more of a leadership role than either the *Sidekick* or the *Guide*. The *Mentor* may instruct and train the PC as well as hand out quests, as for example the character "Maze" in *Fable* (Lionhead Studios, 2004). As such,

the *Mentor* dominates the PC and the PC is in many ways reliant on the *Mentor*'s assistance in pursuing the game objectives.

Unfriendly roles

Among the unfriendly roles that Isbister (2006, pp. 240-246) presents; *Obstacles*, *Enemies*, *Competitors*, *Boss Monster*, and *Archenemy*, only *Obstacles* are noted as being susceptible to having some form of social interaction with the player (unless one counts "fighting" as a social interaction). Obstacles are used to obstruct the player from pursuing its goals and could in that role also socialize with the PC. Just like most of the friendly characters listed above, the unfriendly characters are often one-dimensional (also known as flat) and have therefore little to offer the player apart from resistance. However, taking into consideration that these characters could be multi-dimensional, or deep, they could offer an interesting challenge for the player. Imagine a situation where an NPC and the PC have conflicting goals, but still see that they need each other in order to succeed in reaching their respective goal. Under those circumstances, socially interesting interactions may emerge, where lies and other manipulative behaviour are present.

Neutral roles

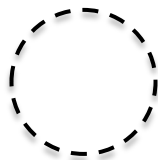
There are also a number of roles that the PC meets that are neither friendly nor unfriendly, such as *Informants* and *Traders* (Isbister, 2006, pp. 247-249). These characters have an agenda of their own but offer services that the PC is in need of. They can also be used for passing on messages and background information. Hence, they are also candidates for engaging in natural language dialogues, perhaps they are even the first ones to be considered due to the fact that these dialogues first and foremost are practical and thus share many similarities with practical dialogue systems.

2.2.3 Believability of NPCs

Human-like NPCs are adaptations of real people, but what characteristics do they need in order to be believable? According to the "media equation" (Reeves & Nass, 1996), people have a tendency to treat computers and other types of media as humans, even though they apparently lack several important properties for life-likeness. This may be explained by our ability to use pattern-matching in order to make sense out of the world, an observation that constitute the fundamental premise of gestalt theory:

There are wholes, the behaviour of which is not determined by that of their individual elements, but where the part-processes are themselves determined by the intrinsic nature of the whole. (Wertheimer, 1938, s. 2)

The strategies for recognizing patterns have in gestalt theory been identified as a number of gestalt laws: the law of similarity, the law of proximity, the law of common faith, the law of good continuation, and the law of closure. The last one, the law of *closure*, refers to way we create a gestalt by filling in the missing pieces, which allows us to interpret the following shape as a circle by subconsciously filling in the void:



If the lines instead would represent different human-like behaviours, it seems fairly reasonable to assume that a human would fill the missing pieces with other human-like behaviours and start treating the object as a human even though it is not. However, experiences from robotics show that when human likeness increases, so does the sense of familiarity up to a point when the robot instead becomes repulsive. This dip in familiarity is also known as *the uncanny valley* (Mori, 1970). Achieving even higher human likeness may turn the curve upwards, up to the point where 100% familiarity is reached, which corresponds to a “healthy human”. Increasing human likeness is thus worth an effort, but only to a point where familiarity is sustained and the uncanny valley is avoided.

2.2.4 Examples of Conversational NPCs

There are no current examples of commercial games in which the NPC can engage in natural language dialogues with the player. The closest example is *Façade* (Mateas & Stern, 2003)⁹, an interactive drama developed within a research project. The story in *Façade* revolves around a married couple that has some marital problems and to which the player character has been invited to for a social visit. The gameplay in *Façade* is based entirely on the social play between the characters and the player may “talk” to the NPCs using natural language text input and direct manipulation.

There are undeniably difficulties in on the one hand allowing free-text communication and on the other hand telling a story that is already authored. The reason is that free-text communication can result in unpredictable input and in order to sustain believability, the system must be able to handle a range of unrecognized and/or irrelevant contributions as part of the story progression. *Façade* has managed to balance this act in an impressive way by offering solutions that take both the local as well as the global context into account. The

⁹ More publications and a downloadable version of the project are available at <<http://www.interactivestory.net>>

global context is for instance used when the player says something irrelevant or not understandable, while the local context is applied when the input is relevant for the current beat¹⁰. *Façade* works because the world in which the story takes place is well defined and limited. The playtime is very short compared to commercial games and even though their work is impressive it seems difficult to scale up. A more detailed presentation of *Façade* along with an analysis of it in terms of gameplay design patterns will be given in chapter 3.

Another game-like research prototype that uses natural language interaction is the research project *Interactive Story Project* (e.g. Cavazza & Charles, 2005; Mead, Cavazza, & Charles, 2003) in which the user participates in a sitcom drama as an “active spectator”. The user may directly manipulate the objects in the environment as well as use voice to give the main character suggestions on what to do. The user may thus change the plan of the character and influence the development of the story, but is not acting out a particular role. The context determines the current interpretation of the input utterance, which means that an utterance can generate different plans depending on the situation. Since the user is merely a spectator, the characters have the freedom to choose those acts proposed by the user that are understood and relevant under the current circumstances. Hence, if the user’s contribution is not understood, the character can choose to ignore it.

Johan Boye and Joakim Gustafson and colleagues (Boye & Gustafson, 2005; Gustafson, Boye, Fredriksson, Johanneson, & Königsmann, 2005) have developed a game within the European NICE project (2002-2005) on Natural Interactive Communication for Edutainment. Their game world is based on the fairy-tale *Cloddy Hans* written by the Danish author H. C. Andersen and consists of two fairy-tale characters, *Cloddy Hans* and *Karen*, which have conversational capabilities. The player acts through *Cloddy Hans*, who serves as the player’s helper character. *Cloddy Hans* can understand commands as well as give advice to the player. The player can also talk to *Karen*, who in one of the scenes controls a drawbridge that the player and *Cloddy Hans* need to cross. The player must negotiate with her in order to make her lower the drawbridge. This character has thus a different function from *Cloddy Hans*, which motivates distinct dialogue behaviours (Boye & Gustafson, 2005). This was solved by dividing the dialogue manager into a generic (kernel) manager and a script-based specific dialogue manager, where the latter one made it possible to create unique behaviours for each character in the game.

¹⁰ Within dramatic writing a beat is the smallest element of structure (McKee, 1997). It is an exchange of behaviour in action/reaction that shapes the turning of a scene. In Mateas and Stern’s use, a beat may be a larger structure, comprised of anywhere from 10 to 100 joint dialogue behaviours.

2.3 Social Activities

Interacting with a CA is a form of social activity, which is when two or more agents “engage in action in a coordinated way, [...] which collectively has some purpose or function” (Allwood, 1993, p. 9). Typically, an *institutionalized* activity is associated with certain conventions and norms, and the participants are assigned specific roles that determine their communicative behaviour. The participants exchange appropriate communicative acts given the context, their roles, the purpose for interacting and the surrounding environment (e.g. Halliday, 1978; Allwood, 1995). The context furthermore determines the meaning potential (Halliday, 1978), i.e. what we can mean (see also Wittgenstein, 1953; Allwood, 1995).

A social activity can then be classified according to the following parameters (e.g. Allwood, 1995; Allwood, Traum, & Jokinen, 2000): The purpose for engaging in the activity, the artefacts, which are prominent in the activity, and the social and physical environment in which the activity takes place. The examples presented later in this thesis will be analysed using these parameters.

2.3.1 Context

Ordinarily, communicating with a practical dialogue system is an example of an institutionalized social activity. The domain is limited and well defined, within which the agent has expert knowledge. A travel agent that books flight tickets, for example, must have access to and be able to provide information about timetables, available seats, and prices for a particular route. The user does not necessarily expect it to be able to check whether they have enough money in their bank account to pay for the flight or to provide weather reports of the target destination. Interacting with a travel agent can also be said to constitute a specific situation type (Halliday, 1978), i.e. a situation classified according to the social activity that takes place (in this case trade); the relationship between the participants and their roles (travel agent and customer); and the role language plays in the situation (booking flight tickets). The meaning potential of the situation type in question increases the robustness of the system both because the number of possible interpretations is reduced, but also because the user’s response in some respect is predictable due to the predefined structure of the dialogue. Most of the NPC roles mentioned earlier typically also appear in specific situations, but since the NPC acts in a dynamic world the current game state must also be considered.

Most ECAs are displayed to give the illusion that they share the same physical space as the human user, thus acting in the user’s physical world with real world consequences and with knowledge directly applicable in the user’s primary

domain. Games, on the other hand, are generally based (more or less) on fiction and the domain is a world of make-believe, in which communication takes place at a higher layer by use of metaphors (Clark H. H., 1996). However, a game may include real world elements in the fictional world, which allows the player to use references to two different but overlapping universes. In *Animal Crossing* (Nintendo, 2001), for example, the game world time is mapped with the player's real time in the physical world. This means that the game characters can comment upon the player's presence in the game and complain if it has been too long since the last visit and it allows the characters to greet the player on his or her birthday and to celebrate Christmas in real time, but within the game domain. The fictional component of games is one of the major differences between interacting with an ECA as opposed to an NPC and in section 2.5 the implications of this difference related to their dialogue behaviour will be discussed.

2.3.2 Communicative Acts

The concept of communicative acts originates from speech act theory, where speech acts refer to utterances that change the state of the world, typically the mental state of the participants or the state of the dialogue (Austin, 1962; Searle, 1969; Traum, 1999). According to Allwood (e.g. Allwood, 1976, 1995), a communicative act is a speech act that takes the evocative intentionality into account – a speech act such as an assertion, for example, will in speech act theory only consider the *intention to express a belief*, whereas it as a communicative act also would include the *intention of evoking the belief* in the hearer. In dialogue system design the term “dialogue acts” is often used with basically the same meaning (see e.g. Traum, 1999; Jurafsky & Martin, 2000).

Assuming that the participants follow the conventions and social norms associated with the activity they currently take part in, they have a range of communicative acts to choose from. Among these acts, some will be assigned the sender role, while the others will belong to the receiver role (see e.g. Allwood, 1995; Allwood, Traum, & Jokinen, 2000). In dialogue with a shopkeeper, for example, the customer role is associated with sender obligations such as requesting a specific item, while the response obligation for the shopkeeper would be to respond to that request, for example by offering the requested item.

In this thesis, all user input to the dialogue manager will be treated as communicative acts.

2.3.3 Cooperation

Successful communication assumes that the participants take one another into mutual consideration and cooperate in trying to achieve a common purpose (see for example Allwood, 1976, p. 40). This includes attempting to understand the

other participants' intentions and making an effort to respond appropriately. In order to do so, Grice (1975) says that the participants of a conversation are expected to observe the *cooperative principle*:

Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. (Grice, 1975, p. 45)

Assuming such a principle is acceptable, he continues, one may distinguish four categories, or maxims, which will yield results in accordance with the cooperative principle (pp. 45):

A. Maxim of Quantity: Make your contribution as informative as is required (for the current purpose of the exchange) and do not make your contribution more informative than is required.

B. Maxim of Quality: Try to make your contribution one that is true, and

1. Do not say what you believe to be false.
2. Do not say that for which you lack adequate evidence.

C. Maxim of Relevance: Be relevant

D. Maxim of Manner: Be perspicuous, and

1. Avoid obscurity of expression.
2. Avoid ambiguity
3. Be brief
4. Be orderly

These principles describe an ideal cooperative interaction and participants may voluntarily or involuntarily fail on one or more accounts. A speaker can for example “violate” a maxim or “opt out” from both the maxim and the cooperative principle or be unable to fulfil a maxim – i.e. face a “clash”, or “exploit” the maxim by “blatantly fail to fulfil it” (Grice, 1975, p. 49) – as in the case of indirect speech acts (“can you pass me the salt”) and implicatures (Grice’s terminology for utterances with implicated meaning):

A: Do you know what time it is?

B: The mail was just delivered

Dialogue 2: Example of conversational implicature

At first glance speaker A seems to violate the Maxim of Relevance. However, the cooperative principle presumes a mutual compliance, meaning that even if the answer at first seems to be irrelevant, speaker A will interpret B’s answer as a relevant, however partial, contribution. Speaker A may therefore infer that speaker B intended to convey that the time is around the time of when the mailman usually comes. Conversational implicatures thus explain how we can

form a coherent discourse from utterances in a conversation that appears to be unconnected (Levinson, 1983).

Grice's cooperative principle has been used for dialogue system design. Bernsen and colleagues, for example, used Grice's cooperative principle to form guidelines for designing and evaluating dialogue systems for the purpose of "optimising the dialogue co-operativity of the system" (Bernsen, Dybkjær, & Dybkjær, 1996).

Allwood criticizes Grice's maxims for overlapping and for not being exhaustive and responds by introducing the notion of *Ideal Cooperation*, which he suggests is achieved when the dialogue participants in their actions (1) take each other into cognitive consideration, (2) have a joint purpose, (3) take each other into ethical consideration, and (4) trust each other to act in accordance with (1)-(3) (Allwood, 1976, 1995; Allwood, Traum, & Jokinen, 2000).

The minimal requirement for cooperation is that the agents take each other into cognitive consideration (1), that is, acknowledge each other and attempt to understand and perceive the other's actions.

Joint purpose (2) is related to the following dimensions: Degree of mutual contribution to shared purpose, degree of mutual awareness of shared purpose, degree of agreement made about purpose, degree of dependence between purposes, and degree of antagonism involved in the purpose. The agents must not have independent or antagonistic purposes in order to be counted as joint purpose. (Allwood, Traum, & Jokinen, 2000)

The ethical considerations (3) include strategies for social politeness and face management, which will be further discussed in chapter 5.

It is assumed that the dialogue participants are "motivated rational agents", which is characterized by the following principles (in short) (see e.g. Allwood, Traum, & Jokinen, 2000): An "agent" behaves intentionally and with a purpose; a "motivated" agent participates voluntarily, seeks pleasure and tries to avoid pain. A "rational" agent shows competent and adequate behaviour. Within the principle of adequacy, Grice's maxims of quantity, relevance and manner are included, while the maxim of quality partly belongs in the principle of motivated action and partly to the principle of competence (Allwood, 1976).

2.4 Dialogue Management Tasks

The dialogue management component of a dialogue system is responsible for controlling the flow of the dialogue by updating the dialogue context and deciding what the agent should say next, and how to say it. It may also include

having access to domain and task knowledge. (See for example Traum & Larsson, 2003; Ginzburg & Fernández, 2010; Jurafsky & Martin, 2000).

Speakers typically distribute their contributions in a structured way to avoid overlaps and misunderstandings, and they may give feedback and correct errors during the interaction. Bunt (1994) refers to these aspects of conversation as *dialogue control functions* and include features such as turn-taking and initiative. Other aspects, such as the number of participants, may have implications for how these control functions come into effect. Turn-taking, for example, becomes more complex when more than two persons are engaged in the conversation. In the following sections a number of tasks an NPC may need to be able to handle will be presented.

2.4.1 Initiative

A dialogue system can be single initiative, i.e., controlled by either the user (user-directed) or the system (system-directed). In the former, the system waits for an input from the user and then attempts to provide an answer. In the latter, it is the system that asks the questions and consequently, the sense of being in control is restricted. On the other hand, when the system is in charge, language understanding becomes more robust since the system's limitations are more obvious to the user.

In ordinary face-to-face dialogues, any of the participants may initiate and/or control the dialogue and initiative may vary during the course of the interaction. This is typically referred to as having *mixed initiative* (see for example Allen, Guinn, & Horvitz, 1999).

MIMIC – a voice-enabled telephone-based dialogue system (Chu-Carroll, 2000) is an example of a system that uses *adaptive* mixed initiative. The system automatically adapts the distribution of initiative based on the information extracted from user utterances and dialogue history.

2.4.2 Turn-taking

In dialogues there are at least two participants engaged in a conversation and they distribute turn in an orderly manner in order to avoid overlaps. If more than two participants are involved, turn taking becomes more complex since the participants now need a way to determine who should speak next. From studying transcripts of real human conversations Sacks, Schegloff and Jefferson (1974) found that people distribute turns in a regulated way and that turn-taking typically applies at specific points in the dialogue, for instance at sentence boundaries – something they refer to as a transition-relevance place (TRP). At a TRP the current speaker may select the next speaker, but if the selected speaker refuses to

do so, another speaker may self-select. If no one takes the turn, the current speaker may continue.

Turn taking in single initiative systems is easily regulated “because one agent initiates all interactions and waits for the appropriate response before moving on to the next interaction. Who has the turn is always well-defined, and the agent in control always initiates turns” (Allen, Guinn, & Horvitz, 1999).

Spoken dialogue systems offer different ways for the user to interact with the system that will affect turn taking. In so-called “push-to-talk” systems, the user presses a button in order to input speech, which means that the system always knows when the user is about to speak. Other technologies that are available are “barge-in”, “wake-up commands”, and “always listening” (Alewine, Ruback, & Deligne, 2004). In barge-in systems, the user may input speech while an audio prompt is playing, which causes the prompt to stop playing; wake-up systems react to certain keywords and starts processing speech as soon as a keyword has been detected; and the always-listening systems, finally, are always prepared for a user input.

Kronlid (2008) investigates turn-taking among a collective of (human or artificial) agents, and presents a dialogue manager for multiparty dialogues that for example can “identify and handle occurrences of shared TRPs, where the current speaker does not select an individual to become the next speaker, but instead selects a set of participants of which the next speaker should be a member” (p. 126). In multiparty settings, individual differences among the participants may also have to be considered; some participants may for example be more eager to take and hold the turn than others. The status of the participants, the distribution of roles (for instance doctor-patient, friends, mother-child *etc.*) in the situation and other contextual factors may also affect how, and if, the ordinary rules for turn-taking apply. Even silence may have to be considered as a turn, for instance when it occurs as the second turn of an adjacency pair (Jurafsky & Martin, 2000). Jan & Traum (2005) have studied these variations and present an algorithm for *simulated* turn-taking among background characters in a virtual world. In this case the actual information that is being exchanged is unimportant. Instead the “appearance of conversation and the patterns of interaction” have been in focus. Their simulation takes into account parameters such as probabilities for talkativeness (wanting to talk), transparency (producing explicit positive and negative feedback and turn-claiming signals), confidence (interrupting and continuing to speak during simultaneous talk), interactivity (the mean length of turn segments between TRPs) and verbosity (continuing the turn after a TRP at which no one is self selected). A first study showed that the agents simulating turn-taking were perceived to be more believable than those that had a random behaviour. This suggests that turn-taking behaviour among virtual agents needs to be unique for each individual agent, possibly in accordance with their

personality type, and that the behaviour is regulated by rules rather than randomly selected.

2.4.3 Incremental Text or Speech Processing

Kempen & Hoenkamp (1982; 1987) argue that the stages in spontaneous speech production: conceptualization, formulation, and articulation, run in parallel and that sentence production therefore is performed incrementally. That is, when a segment has completed the conceptualization and formulation stage it can be articulated and meanwhile other segments are being processed in the first two stages simultaneously. Observations that support this claim include hesitations, fill-ins, barge-ins and false-starts, i.e. errors or disfluencies that could have been avoided if the process was serial.

Most dialogue systems, however, generally process input in chunks and produce canned text responses, thus offering limited options for different types of interruptions. As a step towards creating more accurately modelled speech in spoken dialogue, Skantze and colleagues (Skantze & Schlangen, 2009; Skantze & Hjalmarsson, 2010) have presented a first attempt at generating speech incrementally and Visser and colleagues present a computational model for incremental grounding in spoken dialogue systems (Visser, Traum, DeVault, & op den Akker, 2012). It has previously not been possible to achieve incremental natural language processing using existing standard technology, but recently Google in cooperation with W3C presented the Web speech API¹¹, which is a JavaScript-based API for creating speech-enabled web pages that support continuous speech input.

For game dialogues based on natural language interaction, incremental processing would make it possible for the player as well as the NPCs to give continuous feedback, provide quick responses, interrupt and hesitate, and would thus be a valuable asset in creating believable dialogue behaviour.

2.4.4 Multi-party dialogue

A multi-party dialogue occurs when more than two participants engage in a conversation. In *Façade*, for example the player can engage in a conversation involving both characters, and in *The Sims*TM (Electronic Arts, 1998), socializing with the NPC constitute a large part of the gameplay and multi-party dialogue is implied even if the player only can address one character at a time.¹²

¹¹ <<https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>>.

¹² Several games in the Sims-series have been released, all sharing the core mechanics of the first game.

Multiparty dialogues taking place in virtual environments increase complexity in several ways (see e.g. Traum & Rickel, 2002): First, it may be difficult to identify the current speaker since the agent may be located outside the visual range displayed on the screen. Second, the user may have difficulties in separating the voices, especially when the source of the sound cannot be determined due to limitations of the sound system. Third, turn-taking may be confused when more than two agents are competing for the turn. Ways to handle multiparty dialogues among interactive agents have been proposed by for example Traum & Rickel (2002) and Kronlid (2008).

2.5 Potential design differences between practical DS and game DS

Game dialogues are in many ways practical, but some of the requirements that are important for practical dialogue systems may need to be adjusted to suit the game environment. Jokinen (2009, p. 99) summarizes the desiderata for a dialogue agent in the following four points: (1) physical feasibility of the interface, (2) efficiency of reasoning components, (3) natural language robustness, and (4) conversational adequacy.

The first point (1) basically means that the agent should be easy to communicate with, including ergonomic as well as usability aspects. The second point (2) mainly concerns the response time. If the dialogue system takes too long to respond the interaction becomes frustrating for the user. The agent may also be perceived as impolite. However, based on previous studies, Jokinen means that users tend to be more forgiving towards an agent if it is interesting and novel (Jokinen & Hurtig, 2006) in (Jokinen, 2009). Natural language robustness (point 3) means that the system is linguistically capable of handling the task. This is typically not an issue for simpler command-based systems that handle well-structured tasks, but for more elaborate tasks Jokinen stresses the importance of high quality of the agent's communicative abilities to increase the usability. The last point, conversational adequacy (4), refers in short to the agent's capability to handle miscommunication cooperatively, appropriately and intelligently (see e.g. 2.3.3). However, some of these requirements may perhaps be handled differently when designing game dialogue systems, which will be discussed below. Also, the requirements she suggests do not cover believability aspects and since they are important for NPCs the discussion below will also take such issues into account.

2.5.1 Correctness and cooperativeness

The user expects a practical dialogue system to be trustworthy and assumes that the information provided by it is correct and relevant. Cooperativeness is one way of making the interaction effective, functional, and user-friendly for the human user. Its benefits are obvious in these interactions since the user's main interest is to solve tasks. If the CA asks irrelevant questions or starts to talk about something else the user might get frustrated and confused and become uncertain whether the task has been solved or can be solved altogether. Cooperativeness in Grice's sense (see section 2.3.3) furthermore assumes a mutual interest in reaching a particular goal. If the agents have conflicting goals, however, the maxims might be violated (or in other ways fail to be fulfilled) on purpose with the intent to deceive or start a conflict for example.

In a fictional setting, such as a game, the possibility of violating the maxims when appropriate (or necessary) may actually support a more believable behaviour, or as Hayes-Roth and Doyle (1998) put it: "Rather than correctly, animate characters must behave appropriately—given their roles, their circumstances, and their imperfect human-like natures" (p. 208).

Since Grice's maxims have been used to form the guidelines for designing dialogue systems it could be interesting to see how different violations to the maxims could be used to increase the believability of a game dialogue system:

Maxim of Quantity: Many NPCs either give out too much information or too little information depending on the current game state and the interpersonal relationship. This is a tool for the designers to control how the events in the story are distributed. Information can also be treated as a reward, delivered when the player has achieved something in the game. Sometimes, a character reveals his whole life story to the PC – even though they just met. To overwhelm the player with information like that is in general not recommended and Sheldon (2004), for example, says that characters deserve more respect than being reduced to information booths.

Maxim of Quality: If the participants have conflicting goals, one or both are likely to withhold information or lie. We can think of situations where an NPC provides the player with false information or withhold information until a relationship has been established or a certain game state has been reached. In the game *L.A. Noire*, for example, the player investigates a crime and must take into consideration that the people (NPCs) she encounters have reasons to lie or withhold information, which also are the explicit options presented to the player in the form of "truth", "doubt", and "lie".

Another situation in which the maxim of quality may be violated is when the participants engage in a gossip conversation as they then might say something for which they "lack evidence" (see chapter 7.2).

Maxim of Relevance: For the player, a violation of this maxim would be to talk about events and objects not belonging to the game domain for example. An NPC could also violate the maxim of relevance by not taking into consideration the player's communicative acts or the history of the game session, which is an accumulation of the player's previous actions. There is also a possibility that the NPC will provide irrelevant answers due to misunderstandings. In *Façade* (Mateas & Stern, 2005), for example, the NPCs select an answer from a set of general answers in case of a misunderstanding.¹³ This answer may be perceived of as an irrelevant, but since these phrases are relevant in the global context, it always makes sense in some regard. The player may therefore find the answer believable since it creates the illusion that the character either does not care too much about what the PC says, which could be excused by the fact the NPCs may be distracted due to their marital problems, or prefers to pretend not to have heard or understood. In a story-driven setting such as this one it is thus easier for the user to create a reasonable explanation for an irrelevant answer.

Maxim of Manner: According to the maxim of manner, a speaker should try to be brief, clear and orderly and avoid ambiguities. How a speaker manages to follow these rules may be determined by its personality traits, social status, emotional state, and the current situation. For example, an extrovert character is possibly more talkative than an introvert character, a frightened or nervous character may have trouble being "orderly", and the function of a character may involve using "obscurity of expressions" and "ambiguities, for example a wizard speaking in riddles. In general an NPC does not need to follow any rules of etiquette, it only needs to play its part in a fictional story – be it functional or social – in a believable way.

2.5.2 Reliability and Efficiency

The purpose of interacting with a traditional CA is to be provided with a certain service. It is therefore important that the CA behaves consistently and that the interaction has a predictable outcome. In games, on the other hand, the believability of the character may clash with the predictability of the dialogue system. This requirement is also related to cooperativeness as discussed above. An unfriendly NPC, for example, could have good reasons for lying to the PC, especially if its goal is in conflict with the PC's goal.

The main motivation for interacting with a practical CA is to successfully perform a certain task. One requirement for achieving high user satisfaction in these interactions is efficiency, which includes limiting the number of turns and seeing that the response time is acceptable (see for example Walker, Kamm, & Litman, 2000). In a game, there can be several reasons for the player to engage in

¹³ Answers are primarily selected to suit the local context.

a dialogue: to socialize, get game hints or quests, or just for entertainment. A friendly NPC should perhaps be designed to handle longer exchanges and a variety of topics, while the requirement of a neutral NPC might be to efficiently and reliably help the player (or PC) to achieve some goal.

2.5.3 Error Handling

Error handling mainly concerns two different types of errors: no input and no match. The former refers to situations when the user is silent, i.e. when the system cannot detect any input, while the latter occurs when the system fails to understand the user's utterance. Problems in understanding can be of two types (Skantze, 2003, 2007): misunderstandings and non-understandings. Basically, misunderstandings occur when recognition works but the dialogue system misinterprets the user's intention, while a non-understanding occurs when it fails to obtain an interpretation altogether. The quality of the dialogue system may have strong impact on how the user perceives the system, which may be evaluated in terms of how many times the user must repeat the utterance before the dialogue system understands (Walker, Kamm, & Litman, 2000). Any dialogue system must be able to handle upcoming errors, but the strategies may differ. When the aim of a practical dialogue system is to (cooperatively, see e.g. point 4 above) reach a level of understanding in the dialogue that will lead to a successful completion of the task, a game dialogue system may allow an NPC to behave irrationally or emotionally instead. It can pretend to have understood, change subject or perhaps just end the dialogue abruptly (see for example Reilly & Bates, 1995).

2.5.4 User Role and Setting

We would also like to point to the fact that the user's role and the setting may affect the design choices. A typical CA may assume that the user will play a role that is part of his or her real identity (see for example Gee, 2003), such as pupil, doctor, or soldier. Whereas a human interacting with an NPC will be assumed to play a fictive role, designed at least in part by the game system (see also section 3.2.1).¹⁴ This change in perspective may have implications for how the human participant behaves, what expectations the human has of the NPC's abilities, and the purpose for interacting, all of which in turn may be influenced by the player type.¹⁵

¹⁴ For example, in most role-playing games, the player can define the characteristics of her avatar, but the properties to choose from are defined by the game system.

¹⁵ Bartle (1996) presents a rough classification of player types, consisting of four dimensions: achiever – basically referring to players whose main motivation is to seek rewards, killers – players who seek pleasure in sabotaging other players' game, socializers – who wants to interact with other players, and explorers – players whose main motivation is to investigate and explore

2.6 Rule-based Approaches for Dialogue Management

Several different approaches exist to dialogue management (DM) for spoken dialogue systems and they can be divided into three broad areas (see e.g. Lee, Jung, Kim, Lee, & Lee, 2010): rule-based DM, data-driven/statistical DM, and hybrid DM. Rule-based DMs are regulated by handcrafted rules. The data-driven approaches use machine-learning techniques to accommodate the inherent uncertainty of the ASR and are usually based on Markov Decision Processes (MPDs), in particular Partially Observable MPDs (POMDPs) (e.g. Pieraccini, Suendermann, Dayanidhi, & Liscombe, 2009; Young et al., 2010). The last area, hybrid dialogue management, uses a combination of the rule-based approaches and the statistically based approaches, where rules are defined to constrain the possible actions and thereby rule out those actions that are inappropriate in the current situation.

As have already been mentioned, the current versions of SCXML and VoiceXML require extensions to the standard in order to support stochastic-based dialogue management. Griol, Callejas, and López-Cózar (2010) present a proposal of how to integrate statistical approaches to VoiceXML and we expect to see more examples such as this one in the future, but currently these approaches are investigated in research projects and are not used for commercial purposes.

The rule-based approaches may be classified according to their complexity, ranging from least to most complex: Finite state-based, frame-based, plan-based and agent-based (e.g. Allen et al., 2001; McTear, 2002). The information state update approach (Larsson & Traum, 2000) has elements of all the aforementioned. In the following sections each of these strategies will be presented, followed by a section discussing the different approaches towards dialogue management in relation to standard web technology. In 4.5 we will discuss and present examples of how these strategies may be implemented using SCXML.

2.6.1 Finite State-based Approach

Simple practical dialogues can be successfully modelled using a finite state-based approach. Consider for example the following dialogue for ordering a ticket between different planets in the solar system:

S: What planet would you like to go from?
U: From Pluto to Mars

every inch of the game system and the game world. Depending on the motivation behind the play, different behaviours are revealed (see for example Yee, 2005)

S: Sorry dude, I didn't get that
 U: Pluto
 S: To which planet?
 U: Mars

Dialogue 3: An example of a dialogue between a space travel agent and a customer

In order to provide the user with the requested service – to order a ticket between two planets – the system needs to know the planet to depart from as well as the destination. In a finite state-based system, this information must be provided one at a time in a predefined order. In this example, the system starts by asking for the departure planet and when an acceptable input has been given, the system continues to ask for the destination. In order to be understood the user may not, as in the example above, provide both the departure and destination in one and the same utterance.

A finite state-based system can be visually represented by a graph consisting of nodes denoting states and directed, labelled arcs connecting the nodes denoting transitions between the states. The graph in Figure 9, below, illustrates Dialogue 3, above.

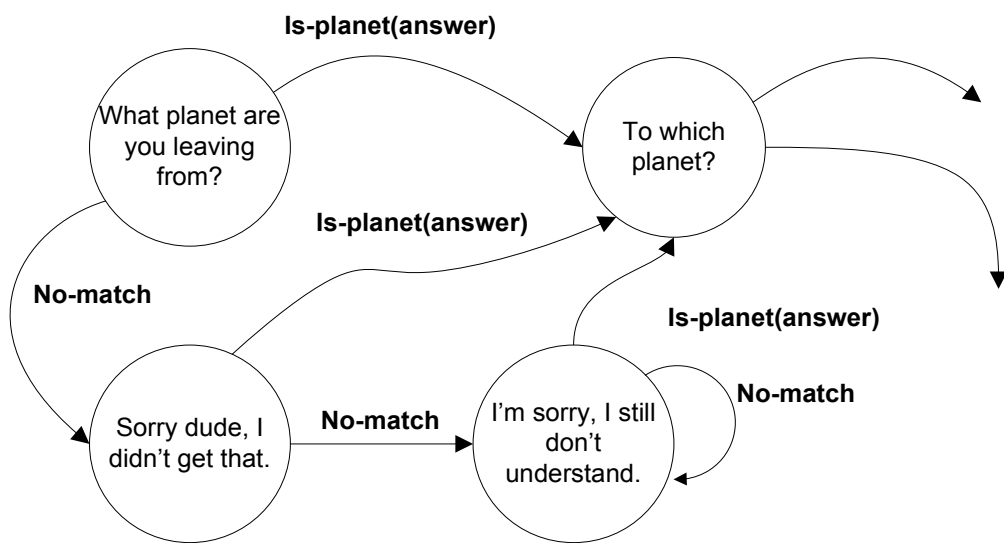


Figure 9: A finite state graph representing a simple practical dialogue

At each state the dialogue system will generate a prompt. Typically this is a canned text response (see e.g. Jurafsky & Martin, 2000) in the form of a pre-recorded utterance requesting specific information from the user. The system will change state if it can match the user's input with any of the outgoing transitions from the current state. If not, it will either re-prompt or start over (for instance after a number of re-prompts). This means that every exchange is within the

scope of the domain and the current state determines whether or not the input is accepted. The conversation is typically system-controlled with the advantage that the range of possible (or rather expected) user input is limited enough to perform a more robust recognition of the user's speech. Furthermore, since the system operates in a limited domain, ambiguity can be avoided to a greater extent since the system can assume the domain-specific interpretation (if such is available) (see 2.3.1).

2.6.2 Frame-based Approach

A frame-based¹⁶ system is in Allen et al.'s (2001) classification regarded as more complex than a finite state-based system as it can elicit information in any order from the user and fill several slots of information from one user turn. This means that a frame-based system requires the following components: (1) A frame (or template) that keeps track of the items of information that the system has to elicit from the user, (2) a more extensive recognition grammar, and (3) a dialogue control algorithm that can determine the system's next actions based on the contents of the frame (McTear, 2004).

The following dialogue example (Dialogue 4, below) is a translation of an authentic conversation between the Swedish national rail company (S) and a user (U):

- S: Hello and welcome to SJ. Choose one of the following services:
Timetable or cancellation.
- U: Timetable
- S: From where do you want to go, and where to?
- U: Stockholm and to Alingsås
- S: From Stockholm Södra to Alingsås. Is that correct?
- U: From Stockholm central station
- S: Sorry, I did not get that. Please answer yes or no. From Stockholm Södra to Alingsås?
- U: No
- S: From where do you want to go and where to?
- U: From Stockholm central station to Alingsås
- S: From Stockholm central to Alingsås. Ok.
- S: When do you want to leave?
- U: Today
- S: Today, the 17 of March, ok.
- S: What time do you want to leave?
- U: at two o'clock
- S: at 1400. Do you want to exclude departures with buses or X2000?
- U: No (Quiet)
- S: Sorry, I did not get that. Do you want [to exclude]
- U: [No]

¹⁶ Also known as form-based systems

S: There is a train leaving from Stockholm at 16:09, arriving in Alingsås at 19.15.

Dialogue 4: Talking to SJ conversational agent – an example of a practical dialogue

In order for the agent to be able to solve the task, it needs to acquire certain information from the user, such as point of departure, destination, date and time. In this example the agent starts by requesting the cities of departure and destination. Even if the agent does not explicitly ask for any other information, it can understand and handle any relevant information provided by the user, see e.g. Table 2, below (square brackets indicate implicit slots):

| Speaker | Utterance | Slots in frame |
|---------|---|--|
| S: | From where do you want to go, and where to? | <from=?> <to=?> [<date=?> <time=?>] |
| U: | I would like to go from Stockholm to Alingsås today at three o'clock | to=Alingsås, from=Stockholm date=today time=three o'clock |
| S: | You want to go from Stockholm central to Alingsås today, the 17 of March, at 1500. Is that correct? | <from=Stockholm> <to=Alingsås> <date=17 of March> <time=1500> |

Table 2: Example slots in a frame-based dialogue system

This is because all the information that the agent needs in order to be able to accomplish the task is collected within the same semantic frame (see e.g. Seneff, et al., 1991). SJ's conversational agent is thus a typical example of a *frame (or form)-based system*, i.e. a system that have empty slots that the user must fill in with data necessary for completing the task, just like traditional user interfaces (UIs). The slots can be filled in any order (at least initially) and several slots can be filled after one user turn, as in the example above. Since the user can "override" the system's question by providing more information than was asked for, frame-based systems are often regarded as having mixed initiative (see 2.4.1). Other examples of frame-based systems are *ATIS* (Seneff et al., 1991), *The Philips automatic train timetable information system* (Aust, Oerder, Seide, & Steinbiss, 1995), and *MIMIC* (Chu-Carroll, 2000). VoiceXML is another example of a frame-based system. The Form Interpretation Algorithm (FIA) that is used to fill the form in VoiceXML iterates through three phases: select, which identifies the next form item to be visited; collect, in which input is gathered; and process, during which the collected data is processed. VoiceXML handles mixed initiative through the <initial> element, which allows the user to provide data for any

or several of the fields in the form at the initial prompt. If there are remaining slots to be filled after initial has done its task, FIA revisit each empty slot in document order in a finite state-based manner until the form has been completed.

2.6.3 Plan-based Approach

For more complicated tasks, for instance tasks that require some form of collaboration or reasoning, more complex systems may be required. Such a system may need to be able to identify a goal and plan a sequence of actions to reach the goal under the current circumstances. These systems are referred to as plan-based systems and are suitable for creating expert systems, such as the *Circuit-Fix-It Shop* system (Smith & Hipp, 1995), in which the user and agent collaborate to solve a task in an otherwise static environment.

Plan-based systems are often based on the Belief Desire Intention (BDI)-model (Bratman, 1987) for specifying the agent's plans. The *belief* represents the agent's perception of the world; the *desire* corresponds to the agent's needs and wants, while *intention* refers to the agent's plan to pursue a goal to satisfy the need (desire). The plan is constructed according to the agent's belief about the world.

In dynamic environments, such as virtual worlds, the preconditions for executing the plan may change. That is, the agent's belief no longer corresponds to the actual state of the world. In this case the agent may need the ability to re-plan in order to achieve the goal, or change goal if desires change. These systems are sometimes referred to as *agent-based* systems (Allen et al., 2001; McTear, 2002).

To use a plan-based approach for modelling character behaviour in games is uncommon since it is too complex, requires too much CPU to process, and is difficult to debug. One exception is the game F.E.A.R. in which the developers made an attempt to create richer characters by use of so called Goal-Oriented Action Planning (GOAP) (Orkin, 2006). The NPC state machines had only three states but used the A* algorithm to plan sequences of actions that would lead to a successful completion of a desired goal in addition to the more common use of path-finding. This way the characters' behaviours could be varied since different actions could be taken to reach a particular goal (Orkin, 2006; Yue & de-Byl, 2006).

Even though a system is not capable of planning, i.e. generating a plan in real-time, it may still be able to specify plans in order to achieve a particular goal. The waiter character presented in chapter 1, for example, used a plan in order to serve the customer the requested dish (see Figure 2). Plans may thus be constructed using statecharts, and other finite state-based approaches, but they cannot be generated on the fly using these approaches.

2.6.4 Information State Update Approach

The information state update approach (see e.g. Cooper & Larsson, 1998; Larsson & Traum, 2000) takes the advantages of the plan-based approach and combines it with the advantages of the simpler finite state- and frame-based approaches. Similar to the plan-based approach, an information state can contain information about the “mentalistic notions such as beliefs, intentions, plans, etc.” as well as the dialogue state (Traum & Larsson, 2003).

“The term INFORMATION STATE of a dialogue represents the information necessary to distinguish it from other dialogues, representing the cumulative additions from previous actions in the dialogue, and motivating future action” (Larsson & Traum, 2000). An information state is conceived of as consisting of several interaction components rather than as a monolithic node. Depending on the dialogue that is being modelled, the kind of components may vary, indicating that the ISU approach should be viewed as rather “empty” framework that needs to be filled with theoretical content (see for example Larsson & Traum, 2000, p. 326). Larsson (2002), for example, implements a theory of issue-based dialogue management for handling questions under discussion based on Ginzburg’s notion of a dialogue game board (Ginzburg, 1996). A simplified version is also used to exemplify the ISU approach in (Larsson & Traum, 2000).

In an information state for a dialogue participant, private information is separated from the information common for all participants. Given the example used in (Larsson & Traum, 2000), the private information consists of two fields: the participant’s beliefs (*Bel*) – a static set of propositions, and the participant’s agenda (*Agenda*), which is a stack of (local) actions – in general those actions that should be performed in the next move (Cooper & Larsson, 1998). The set of common beliefs are those propositions that represent the common ground (see for example Clark & Brennan, 1991), in this case the information that has been established during the conversation. The second common field is *QUD* – a stack of questions under discussion. The latest dialogue move performed (*lm*) is also part of the shared information. In the information state approach, a *dialogue move* (for example an utterance) will trigger an update of the information state according to a set of *update rules*. The *update strategy* decides which rule(s) apply at a given point, from the set of applicable ones (Larsson & Traum, 2000).

The TRINDI Dialogue Move Engine Toolkit (abbreviated to TrindiKit) (Larsson & Traum, 2000) has been acknowledged as the first implementation of the information based approach to dialogue management (Bos et al., 2003). A later example is DIPPER that “borrows many of the core ideas of the TrindiKit, but is stripped down to the essentials” (Bos et al., 2003). Chapter 4.5.4 presents how Kronlid and Lager (2007) have implemented the ISU-approach using SCXML.

2.7 Discussion

The first part of this chapter introduced conversational agents and classified conversational NPCs as a type of ECAs. The roles the NPCs possess were then further divided into friendly, unfriendly and neutral, based on Isbister's (2006) classification in order to determine what type of dialogues they are likely to engage in, if they are likely to engage in dialogue at all.

Interacting with neutral NPCs, such as traders and informers is typically a goal-directed activity in which the characters engage in a practical dialogue and will in many respects resemble interactions with traditional ECAs. It is in general also important that the NPC is reliable and fairly cooperative to avoid user frustration. In these situations, the primary goal is not to socialize but to solve a particular task and the participants are expected to follow certain conventions associated with the task. It should furthermore be noted that practical dialogues have been successfully implemented in commercial systems using a finite state- or form-based approach and an example of a practical dialogue situated in a game-like environment was presented in chapter 1.3.

Practical dialogues in games is one possible use of natural language interaction in games, but for some game characters other types of dialogues are more relevant to engage in. With friendly NPCs, for example, the PC may develop a social relationship, but in order to do so they also need to be able to engage in casual conversations, i.e. conversations that are motivated by "interpersonal needs" (Eggins & Slade, 1997). In Chapter 7 approaches towards modelling casual conversations will be discussed..

It was also suggested that it could be interesting to have unfriendly NPCs that constitute non-violent obstacles to the player. They could for instance obstruct the player's progression in the game by spreading lies or in other ways manipulate the player (character). This is also a kind of a socially oriented dialogue and requires that the NPC is socially skilled. In chapter 5, a waiter character with the ability to adjust its behaviour according to the situation is introduced.

Chapter 3

Game Dialogues

The dialogue (language) spoken by the characters and the opportunities for the player to engage in dialogue are another material resource for action. Dialogue is a powerful means for characters to express their thoughts; it is instrumental for helping the player to infer a model of the characters' thoughts. Conversely, dialogue is a powerful means to influence character behaviour. If the experience makes dialogue available to the player (and most contemporary interactive experiences do not), this becomes a powerful resource for expressing player intention. (Mateas, 2001, p. 144).

This chapter will explore and analyse how dialogues with NPCs are treated in games today. Various games will be investigated in the search for patterns associated with dialogues and dialogue management tasks that recur in games. Actual and potential uses of a game dialogue system will be expressed by means of gameplay design patterns (Björk & Holopainen, 2005) and it will be investigated how novel gameplay can be created by use of natural language interaction. The next chapter will follow up on this and present an approach for managing natural language dialogues in games.

3.1 Layers of Computer-based Games

Computer games as artefacts are multi-faceted means of expression combining various artistic disciplines, such as visual art, sound, music, and storytelling, with low-level data representations and algorithms in order to create an interactive experience for one or more players.

Playing a game means interacting with the game objects under restrictions defined by rules through an interface. Figure 10 shows the relationship between the layers involved when playing a game as illustrated by game designer Ernest Adams (2010):

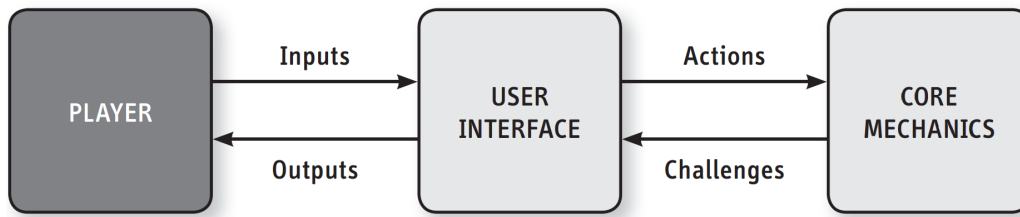


Figure 10: Layers of a computer game (Adams, 2010) ©Ernest Adams, 2010. *Used by permission*

The User Interface (UI) presents the run-time behaviour of the mechanics as an effect of the player’s input – also known as the *dynamics* of the game (see for example Hunicke, LeBlanc, & Zubek, 2004). The player’s interaction with the game dynamics and the responses to those actions, i.e., “the structures of player interaction with the game system and with other players in the game” (Björk & Holopainen, 2005) is often referred to as *gameplay*. In Adams’ (2010) model gameplay is more specifically defined as (1) the challenges that a player must face to arrive at the object of the game, and (2) the actions that the player is permitted to take to address those challenges (p. 38). In simpler terms, Sid Meyer suggests gameplay to be “a series of interesting choices” (quoted in Rollings & Morris, 2000). A *gameplay mode* consists of the particular subset of a game’s total gameplay that is available at any one time in the game, plus the user interface that presents that subset of the gameplay to the player (Adams, 2010, p. 40).

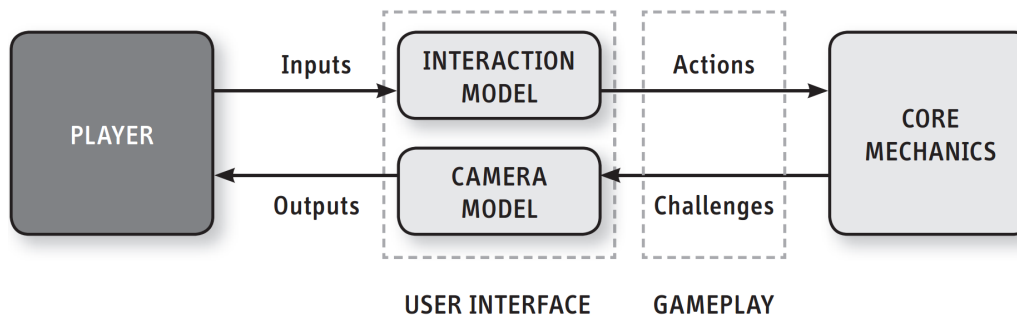


Figure 11: Interaction Model (Adams, 2010) ©Ernest Adams, 2010. *Used by permission*

The UI consists of two components: the interaction model and the camera model (see Figure 11, above). The interaction model describes the relation between the player’s input via the controls and the response to that action from the game system. It communicates the player’s possible and impossible actions and is responsible for sending the player’s input to the core mechanics. The camera model displays the game’s simulated physical space using graphics from a set point of view.

The core mechanics constitutes the data representations and algorithms, i.e. the code layer, in which the game rules are specified, and describe the “essential play activity players perform again and again in the game” (Salen & Zimmerman, 2004). Game mechanics is thus a means or method that facilitates player interaction according to the rule-set to attain the goals stated in rules (e.g. Hunicke, LeBlanc, & Zubek, 2004; Järvinen, 2008; Sicart, 2008). Game mechanics can also be described as “any part of the rule system of a game that covers one, and only one, possible kind of interaction that takes place during the game, be it general or specific” (Lundgren & Björk, 2003).

Games are designed with a player experience in mind and so “Player” is regarded as one of the layers involved in game design. However, the desired experience or “fun” factor varies depending on game genre. A horror game is successful or regarded as “fun” if the player experiences fear, for example, while an adventure game might rely on a strong story. Hunicke, LeBlanc and Zubek (2004) have therefore made an attempt to move beyond the concept of “fun” or “gameplay” and instead classify games according to their *aesthetic* experience, which they describe as “the desirable emotional responses evoked in the player, when she interacts with the game system”. Aesthetics constitute one of the layers in their MDA (Mechanics, Dynamics, Aesthetics) framework, aimed for reasoning about game design using different “lenses”, or views – the player’s as well as the designer’s.

3.2 The Game Setting

Similar to reality-based virtual worlds, (story-driven) game worlds are dynamic and populated with characters that regard the virtual environment as their primary domain (using Clark’s, 1996, sense). The game world may however differ significantly from the real world (as we know it) since it typically has a fictional framing. Juul (2005) suggests that video games are “half-real” since they apply *real* rules to *fictional* worlds. The game may use analogies from the real world, but there is typically no claim for it to be a perfect real-world simulator. Instead, simulation games are “stylized” and only implement “what are considered interesting core parts of the real-world game” (Juul, 2005):

[...] games are often *stylized simulations*; developed not just for fidelity to their source domain, but for aesthetic purposes. These are adaptations of elements of the real world. (p.172)

This applies to NPCs as well, not only because that makes them more interesting, but also because of the complexity of simulating human behaviour. A notable example are the characters in *The Sims*TM (Electronic Arts, 1998), as they simulate human life in most respects; they eat, sleep, go to work, study, use the bathroom and socialize with each other, but they do not spend time paying bills

for example even though bills have to be paid in the game. Instead, this action has been simplified to a single click with the mouse. The game (like many other games) also uses time anomalies to allow time to run at a faster pace than ordinary time.

3.2.1 The Player

In character-based games the player is typically represented by an avatar – an extension of the player into the game world (Wilhelmsson, 2001). The player's interaction with the other game characters is thus conducted through the filter of the avatar and the player is assumed to play a specific role that is an integral part of the game's story. Bartle (2003) distinguishes three levels of immersion the player may experience in this relation: at the lowest level he describes the *avatar* as a “puppet” to which the player has no emotional attachment. In that respect the avatar is merely a tool for interacting in the world. A *player character* is an extension of the self in the world and anything that happens to the player character also happens to one self. The highest level of immersion is achieved when the player *is* the character – which is when the character becomes the player's *persona*:

You're not role-playing a being, you *are* that being; you're not assuming an identity, you *are* that identity. If you lose a fight, you don't feel that your character has died, you feel that *you* have died. There's no level of indirection: *you are there*. (Bartle, 2003, p. 155)

The PC is typically the protagonist of the story and the players' goals are a reflection of the goals they wish to pursue for their character. When Bartle describes the relationship to the avatar in terms of different levels of immersion, Gee (2003) discusses the different player identities involved in play: the *virtual identity*, the *real identity* and the *projective identity*. The real identity corresponds to the physical person playing the game, while the virtual identity corresponds to the player's representation in the game world. The character's successes and failures are a result of the physical person's actions, combined with the virtual character's restrictions and abilities in the game world. The projective identity, lastly, is the interface between the virtual and the real identity and means both “to project one's values and desires onto the virtual character” as well as “seeing the virtual character as one's own project in the making, a creature whom I imbue with a certain trajectory through time defined by my aspirations for what I want that character to be and become [...]” (Gee, 2003, p. 55). Wilhelmsson (2001) prefers to use *Game Ego* because it “emphasizes its connection to the game player as being a part of him or her” (p. 166). Thus, a player has different identities when playing a game and different types of avatars offer different levels of immersion. This is different from interacting with a traditional ECA as these interactions usually are conducted within the user's primary domain. Only

when the user enters a virtual world may some similarity appear, but since these most often mirror real situations the user is still expected to use his or her real identity in these situations, even though it may mean using a specific role in a specific semiotic domain. It is reasonable to assume that the player's relation to its avatar and the fictional world will have impact on their behaviour during play and that the expectations of the interactions with other characters in this world are different from those that have a real-life framing.

3.2.2 Interactivity and Agency

While reading a book or watching a movie in the traditional way is a linear process, even if the story uses back-flashes or time-jumps (Chatman, 1978), playing a game is most often a non-linear process that can take different turns depending on the player's actions. Through his or her actions, the player can interrupt as well as re-direct the on-going exposition. This means that the player's actions affect the outcome of the game in different ways and that one play session almost never is exactly the same as a previous one. Interacting with the characters that populate the world may for example affect their present behaviour as well as their future actions. Ryan (2001) calls the ability to respond to changing conditions determined by the user's input "resource interactivity" and continues by distinguishing between four strategic forms of interactivity based on combinations of two binary modes of interactivity; internal-external modes and ontological-exploratory modes. In the internal mode the player uses his or her projective identity as described earlier in chapter 3.2.1, i.e. by experiencing the world from a first-person perspective using an avatar. In the external mode, the player is situated outside the context of the game, typically experiencing the game from a god-like perspective with an unclear role in the fictional world, as in *The Sims* (Electronic Arts, 1998), *Black & White* (Lionhead Studios, 2001) and most strategy games. In the other dichotomy, the exploratory mode stands in opposition to the ontological mode. In the former the world and plot is already created and cannot be altered, while it in the latter is unfolded on basis of the user's choices: "These decisions are ontological in the sense that they determine which possible world, and consequently which story will develop from the situation in which the choice presents itself" (Ryan, 2001). Combining these modes generate the following forms of interactivity:

- **External-exploratory.** The main type of interactive narrative that belongs to this category is hypertexts, in which a user may navigate through a network of textual fragments in a random fashion without participating in the fictional world. Even though a game could be constructed as a hypertext with included dialogues it would not offer the kind of player participation that is interesting for our purposes.

- **Internal-Exploratory.** The user participates in the fictional world, but his or her role in the world is limited to actions that have no bearing on the narrative events. Typical examples are action-adventure games such as *Myst* (Cyan, Inc., 1993), *Max Payne* (Remedy Entertainment, 2001) and *L.A. Noire* (Rockstar Games, 2011). Ordinarily, dialogues are conducted through menus (will be discussed further down in this section).
- **External-Ontological.** The user interacts using a god-view and may determine the fate of the characters by choosing a path at decision points. Hence, the player is situated outside the game diegesis without a single avatar to control. The Interactive Storytelling system presented earlier (see 2.2.4) is inspired by the sitcom television show “Friends” and makes use of this type of interactivity (Mead, Cavazza, & Charles, 2003). The audience may change the outcome by giving suggestions to the characters, but have no role of its own in the story.
- **Internal-Ontological.** In this mode, the user is given a particular role and determines their fate by acting within the time and space of a fictional world. For each run, a new life and life-story is produced and the destiny of the character is created dramatically by being enacted rather than diegetically by being narrated. Emergent games, such as the *Grand Theft Auto* series (Rockstar Games, 1997-2010) (*GTA* hereafter) belong mainly to this category, even though they contain linear assignments to pursue in order to progress the over-arching story. Dialogues may occur in these games, but since the player is free to move around, dialogues as well as other types of narrative blocks, may have trouble in taking the history of the play session into consideration and logical mistakes can therefore be made.

Different gameplay modes may use different forms of interactivity and a game may thus have elements of more than one of the aforementioned combinations. In *Heavy Rain* (Sony Computer Entertainment, 2010), for example, the player acts through four different characters and the actions performed through each of the roles affect how the story progresses and finally ends. But, even though the game features several different paths and endings, it still is predefined and the destiny is therefore enacted through the limitations of the narrative. It could thus be described as an internal-exploratory game with some ontological elements. The games in the *GTA-series* (Rockstar Games, 1997-2010) present a similar combination, but here the player can choose to play either mode. There is a narrative to unfold, but this can be ignored altogether if one prefers to play around in the open game world. In that respect *GTA* is a typical example of an Internal-Ontological interactivity form, a sandbox game.

As can be noted, the different forms of interactivity determine whether the player experiences agency or not, that is “the satisfying power to take meaningful action

and see the results of our decisions and choices” (Murray, 1997, s. 126). Mateas (2001) means that a player will experience agency when there is a balance between the material (that is, the materialisation of the narrative that can be experienced by the audience, e.g. objects, language) and formal (e.g. the plot) constraints. Imbalance may then be experienced when there are many things to do but no way to tell whether one action is more preferable than another or as Ryan (2001) suggests, “opportunities for action must be frequent, [...] but to maintain the narrative on the proper track, the range of actions must be severely restricted”.

3.2.3 Game Dialogues

When entering a dialogue, the player typically is faced with a text-based menu containing the various options available at the current game and/or dialogue state. In some games the options display the utterance as told verbatim by the PC, as in *Grim Fandango* (LucasArts, 1998a) (see Figure 12, below).

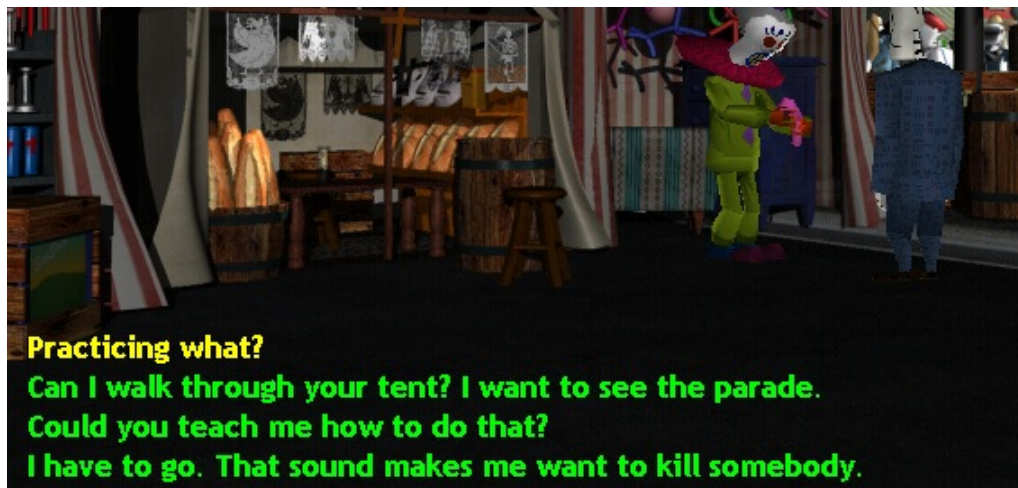


Figure 12: Picture of dialogue menu in *Grim Fandango* (LucasArts, 1998)

Morrowind (Bethesda Game Studios, 2002) presents the player’s choices as hypertext linked to a database, where each choice is an abstract description of the dialogue content (see Figure 13, below). Games such as *Mass Effect* (BioWare, 2008) and its successors use a similar representation, but present the player’s options in a dialogue wheel instead (see Figure 14, below).

Most game dialogues are constructed as branching trees, i.e. giving options to the player that each expands a new branch in the tree. This makes them susceptible to combinatorial explosion (Adams & Rollings, 2007; Bateman, 2007) and so to limit the possibilities they often have dialogue menus with options that eventually

fold back to one single dialogue menu, or link back to a previous menu (see e.g. (Adams, 2010, s. 187)).¹⁷



Figure 13: Picture of dialogue menu in *Morrowind* (Bethesda Game Studios, 2002)

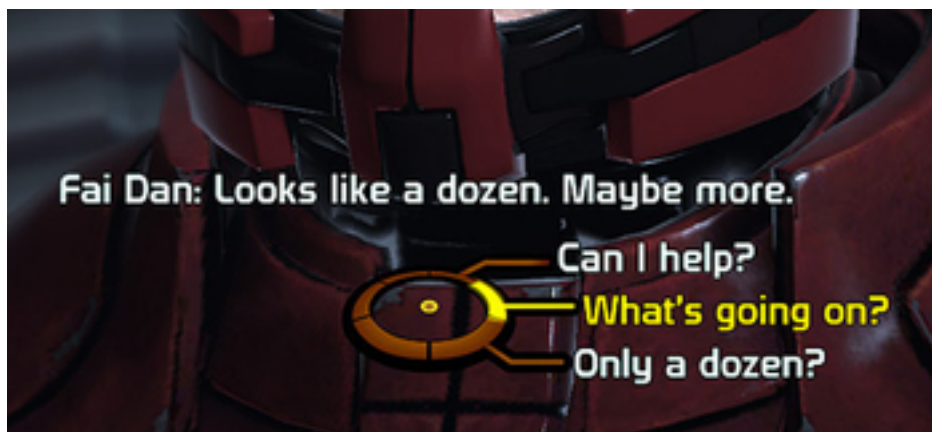


Figure 14: Picture of dialogue wheel in *Mass Effect* (BioWare, 2008)

Engaging in a dialogue typically means entering a new gameplay mode that is separate from the other game activities (see e.g. Adams, 2010, pp.186-192). In most games the switch is visibly noticeable, as in early adventure games, e.g. *Maniac Mansion* (Lucasfilm Games, 1987) and *Grim Fandango* (LucasArts, 1998a), and role-playing games, e.g. *Planescape Torment* (Black Isle Studios,

¹⁷ Cf. branching stories, where each branch in the story tree represents a unique plot line. However, in order to keep the number of different paths limited the branches eventually fold back to a single path or join another parallel branch.

1999). Even though dialogues in more recent game examples still are conducted in a separate gameplay mode the switch is less obvious to the player since the game seems to go on while the PC is engaged in the dialogue, as in for example *The Elder Scrolls V: Skyrim* (Bethesda Game Studios, 2011) and *Mass Effect* (BioWare, 2008).

Game dialogues are a means for moving the story, and thus the game, forward. In order to do this “[t]he words of the characters have affordances in that they reveal salient features of the game environment” (Wilhelmsson, 2001, p. 218) i.e., within an utterance the NPC can give hints to the player about important places, events, and characters by introducing them as new topics that the player later can ask about. Typically, the information is distributed in steps to resemble ordinary human dialogues. In the dialogue excerpt from the game *Space Bar* (Boffo Games, Inc., 1997), presented below, the player can either encourage Soldier 714-Z-367 to tell his story (option 1), or opt out for the information by selecting the second alternative:

1. ENCOURAGE HIM

Alias Node: “I've got lots of time and lots of curiosity. Let's hear your story.”

Soldier 714-Z-367: “I once lived the routine life of a soldier, in the service of the greatest of Zzazzl queens, the revered Zzoonz. Ours was the finest swarm on P'zzazz. I was in charge of guarding her majesty's tapestries, a position of great importance and responsibility.”

2. CHANGE THE SUBJECT

Alias Node: “I'll take a rain check on the story. But tell me, have you seen anything strange around here today, or seen anyone acting kind of odd?”

Soldier 714-Z-367: “Other than you asking many questions, no.”

The first alternative (1) provides the player with new facts (underscored) to ask about in the second round. The second alternative (2) however results in a dead end. Different forms of challenges can be added to the dialogue, such as requiring of the player to use the right phrasing or identifying specific goals that must be accomplished in order to progress the game. Such a goal could for instance be to manage to get specific information or objects from the character (as in the example above). In this particular example, the choices presented to the player are in the form of the communicative acts (“encourage him” and “change the subject” respectively) rather than the exact phrases that the PC (Alias Node) actually says.

Game dialogues are thus highly functional and serve the player with the proper background information to understand the game and get motivated to take certain

actions in order to progress the game. Other functions include trading and delivering quests, i.e. game objectives that the player may try to accomplish. Moreover, as in other fictional stories, the dialogue is essential for revealing information about the (main) character (see e.g. Field, 2005). In the light of this, dialogue trees with scripted options seem to offer some advantages: (1) The scriptwriter has full control of the system and the different dialogue paths are determined in advance, (2) the scriptwriter does not have to consider other possibilities than those presented to the player, (3) the system understands all input from the player and has answers to them, thus making it easier to control the flow of the dialogue and to create dialogues that are consistent with the game as a whole, (4) the player does not get stuck by not knowing how to formulate a relevant question. Instead the challenge can be to choose the option that most likely will help the player forward, and finally, (5) the NPC's, as well as the PC's phrases can be recorded by actors casted for a particular role.

Menu-based dialogues, such as the one presented above, however suffer from certain limitations: (1) the player cannot choose another dialogue move apart from those that are displayed in the interface, and any update of the options is not to be expected unless the player reaches another game (or dialogue) state. This means that there might be a lack of options that mirror the player's intentions. Hence the player may not experience agency in the dialogue. (2) In time the player may have figured out how the system works and manage to choose the right path immediately, thus making the other options superfluous. If, or when, the structure has been identified, the dialogues become predictable. This can also affect agency as the player may be faced with a number of options but cannot determine which alternative is the most preferable. Since some of the alternatives are unimportant, choosing them is meaningless since the outcome is not changed: "The *meaning* of an action resides in the relationship between action and outcome" (Salen & Zimmerman, 2004, p. 157). (3) As was discussed earlier, game dialogues are typically conducted in a separate interface, thus treating the dialogue actions as different from other actions. In this case, the mode of interactivity appears to turn from internal to external (as in the dialogue example above), where the player browses through a number of options leading to a certain outcome rather than experiencing the conversation as part of the diegesis. (4) The player may be able to get information from the character, but the player cannot provide new information in return (i.e. add new information to the game system). (5) Finally, the NPC's behaviour is in general limited to support certain functions rather than offering opportunities to socialize with the NPCs.

For a dialogue to be meaningful, the NPC must be able to take the local as well as the global context into account (as is the case in *Façade*). That is, if the NPC has no memory of previous encounters with the PC, the NPC will appear as either ignorant or incompetent (or both), which may affect believability negatively (see

for example Sheldon, 2004). A character should be able to have a recollection of previous encounters and remember at least core elements from these conversations as this information is assumed (by the player at least) to be part of their common ground – i.e., beliefs that are shared among the dialogue participants (e.g. Clark & Brennan, 1991). Furthermore, to be believable an NPC needs to behave variably but yet consistently (see for example Hayes-Roth & Doyle, 1998), i.e. according to its personality and the current situation. It is here assumed that a character that performs believably in the interaction with the player (character) is more likely to have the ability to evoke emotional reactions in the player as well as to make it possible for him or her to empathize with it, that is, to create the desired aesthetic experience.

There are thus several areas in which game dialogues may be improved to create more believable NPCs and meaningful play. In later sections of this chapter a number of games will be analysed in the search for patterns that define and constitute P(C)-NPC dialogues. The aim is to both to identify existing patterns as well as suggest new patterns that could create new forms of gameplay through dialogue. First, however, an overview of languages for communicating game design will be given since such a language is needed in order to talk and reason about gameplay.

3.3 Languages for Communicating Gameplay

One problem often faced in game development is communication – both among the team members, as they represent various disciplines, as well as with external partners, such as publishers and executive producers. It may for example be difficult to communicate at the right level, such as when a game producer wants to see the technical specification, but rarely has the adequate expertise to understand and review it (Bethke, 2003). In 1999, Church called for a common design vocabulary to allow designers to be able to “communicate precisely and effectively with one another”. He then presented a first attempt to formulate such a vocabulary and named it the “Formal Abstract Design Tools” (FADT) (Church, 1999).

Kreimeier (2002) argues that a language is needed in order to advance the game design profession. As he and others (e.g. Björk, Lundgren, & Holopainen, 2003b) have observed, game design has borrowed tools and techniques from other related areas, such as film studies and storytelling, perhaps sufficient to cover some, but not all, aspects of game design. This may have natural explanations in that game research as an academic field is immature and has previously been represented by scholars coming from related fields because of a lack of an

independent academic game design subject. The initiative for assembling a common language has thus mainly come from the game designers themselves. One attempt is the 400 project¹⁸, initiated by game designers Noah Fahlstein and Hal Barwood, which can be regarded as a “checklist” for game design, where each item in the list is specified by an ID, an “imperative statement”, such as “Provide Clear Short-Term Goals”, an explanation and a domain specifying the situation in which the rule may apply. Kreimeier (2002) promotes the use of game design patterns, a derivative of Alexander et al.’s (Alexander, Ishikawa, & Silverstein, 1977) “pattern language” and Gamma et al.’s “design patterns”, used in software development to present solutions to recurring problems in object-oriented software design (Gamma, Helm, Johnson, & Vlissides, 1995). A pattern language consists of a collection of reusable design solutions applied to an architecture where “[e]ach pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice” (Alexander, Ishikawa, & Silverstein, 1977, p. x). The patterns can thus be combined in a multitude of ways and no pattern is an isolated entity. The patterns are hierarchically organized such that a parent pattern consists of the patterns defined as its children. Kreimeier emphasizes that *game* design patterns first and foremost are concerned with *content* and then continues to give a number of examples in his article.

Björk and colleagues (2003a; 2003b) present a complete framework for describing games and interaction in games. Game design patterns are here treated as “higher level, hierarchical conceptual models of the possible designs of games which manifest themselves in the structural framework by defining the logical, physical and temporal relations of the elements in the framework” (Björk & Holopainen, 2003a). In 2005, Björk and Holopainen published *Patterns in Game Design* (Björk & Holopainen, 2005) consisting of a large collection of gameplay design patterns (GDPs) with the explicit purpose of establishing a semi-formal means of describing, sharing and expanding knowledge about game design.¹⁹ GDPs constitute the basis for the analyses made later in this chapter for two main reasons: (1) the co-author of the original paper on which this chapter is based is one of the founders of the concept of GDPs; (2) GDPs are well documented and constantly under development.²⁰ The number of patterns is increasing, with close to 300 general patterns today as well as several smaller collections (Lankoski & Björk, 2007a; 2007b). Other collections include (Cermak-Sassenrath, 2012), (Hullett & Whitehead, 2010) and (Milam & Seif El Nasr, 2010).

¹⁸ <<http://www.theinspiracy.com/>>.

¹⁹ Björk and others now prefer to use gameplay design patterns instead of game design patterns, which indicates a slight change in focus towards gameplay rather than game design in general.

²⁰ New patterns are for example continuously being added to the project’s wiki: <http://gdp2.tii.se/index.php/Main_Page>.

Also worth mentioning is the Game Ontology Project²¹, in which the important structural elements of games and the relationships between them are identified and hierarchically organized. This project resembles the gameplay design pattern projects in many respects, especially as they also claim that they “do not intend to describe rules for creating good games, but rather to identify the abstract commonalities and differences in design elements across a wide range of concrete examples” (Zagal et al., 2005).

3.3.1 Gameplay Design Patterns

Gameplay Design Patterns (GDPs) can be used and re-used for creating and analysing games. These patterns may be more or less specific and they may be related to other patterns such that one pattern can instantiate or modify another pattern. GDPs do not, however, reveal how to implement the features in the game, and do not attempt to provide that type of information either.

A *Gameplay Design Pattern Template* consists of the name of the pattern, a core definition and a general description. It also contains a description of how to use the pattern, e.g. the choices the game designer is facing by using the pattern, the consequences an inclusion of the patterns has to gameplay and the relations the pattern has to other patterns: *instantiates*, *modulates*, *instantiated by*, *modulated by*, or *potentially conflicting with*. A pattern A *instantiates* pattern B if the presence of B automatically follows from the inclusion of A, and consequently B is *instantiated by* A. When the inclusion of pattern A affects the gameplay of pattern B, A is said to *modulate* B. When the inclusion of A makes the presence of B impossible, A is *potentially conflicting with* B (and vice versa).

To give an example of how a complete description of a gameplay pattern is formulated, the gameplay design pattern for INFORMATION PASSING is presented in Table 3 (part I) and Table 4 (part II) in the following two pages.²²

²¹ <http://www.gameontology.org/index.php/Main_Page>.

²² The full pattern description was retrieved from <http://gdp2.tii.se/index.php/Information_Passing>, but the pattern was first introduced in (Lankoski & Björk, 2007b). Gameplay Design Patterns are marked in the text using small caps, following Lankoski's (2010) notation. A definition of the other patterns mentioned in the table can be found at the GDP wiki: <<http://gdp2.tii.se/index.php>>.

| | |
|---------------------------|---|
| Definition | <i>The passing, from a character to another, of information that has, or can have, influence on the gameplay.</i> |
| Description | When characters interact with each other in games it is quite common that they exchange information, and this regardless of if they are under player control or not. Besides the role this information can have in the development of a story, it can also trigger new goals for players, make them aware of action possibilities previously overlooked, or improve the chances of making good choices. |
| Examples | <p>Players of the <i>Thief</i> series of games receive new goals, or have current goals cancelled, as their characters overheard discussions between non-playable characters.</p> <p><i>Façade</i> allows players to talk to two non-playable characters through free text input. INFORMATION PASSING takes place in both directions since not only do the characters' utterances provide clues for potential conversation topics but the interpretation of the player's input by the game's parser results in the characters changing their opinion about the player's character and each other.</p> |
| Using the patterns | <p>INFORMATION PASSING is easily achieved as part of DIALOGUES in a game although the level of influence on gameplay can vary. While INFORMATION PASSING resulting in the completion or failure of goals directly steers the gameplay, merely providing the player with information also lets the pattern provide CLUES or RED HERRINGS (the latter also showing the possibility of introducing UNCERTAINTY OF INFORMATION). Similarly, the introduction of PREDEFINED GOALS or OPTIONAL GOALS as an effect of INFORMATION PASSING shows two distinctly different ways of influencing the gameplay.</p> <p>One design choice regarding INFORMATION PASSING is if the events are predetermined CANNED TEXT RESPONSES, which are part of PREDETERMINED STORY STRUCTURES (as for example in <i>Thief</i> series) or if the INFORMATION PASSING is part of the game system (as for example in <i>the Sims</i> series). This is especially important regarding the information characters under players' control can pass on to others, since providing options beyond a single utterance or a LIMITED SET OF ACTIONS easily requires advanced parsing capabilities since such a system in practice functions as a CHAT FORUM.</p> <p>One specific way of creating INFORMATION PASSING is to present it as GOSSIP between characters. EAVESDROPPING can be combined with GOSSIP or work independently as another way to instantiate the pattern, but also open up to having GAIN INFORMATION goals requiring STEALTH (i.e. the goal to move through a certain area and perform an action without being detected).</p> <p>The patterns of SYMMETRIC and ASYMMETRIC INFORMATION need not only be applied to players but also to a character level in relation to INFORMATION PASSING, as can COMMUNICATION CHANNELS and those related to PERFECT, IMPERFECT, DIRECT, and INDIRECT INFORMATION.</p> |

Table 3: Gameplay Design Pattern: Information Passing (part I)

| | | |
|-------------------------------------|-------------------------------|---|
| Using the patterns | <i>Diegetic Aspects</i> | One design possibility regarding how the INFORMATION PASSING is represented in the game environment is if it should be presented verbatim or symbolic to the players. While giving players the access to the exact information passed between the characters can more easily provide CLUES or RED HERRINGS it typically requires more resources to develop this type of content for the game. This extra work however also creates the possibility of providing players with INDIRECT INFORMATION and provide a form of PUZZLE SOLVING. In contrast, representing the INFORMATION PASSING as symbols (see for example the Sims series) makes it easier to integrate with the game system and provides easily creates AMBIGUOUS RESPONSES. |
| | <i>Interface Aspects</i> | For simple CANNED TEXT RESPONSES the interface for supporting INFORMATION PASSING can be just a button of a menu to choose from a LIMITED SET OF ACTIONS. |
| | <i>Narrative Aspects</i> | INFORMATION PASSING is typically a cornerstone in games' PREDETERMINED STORY STRUCTURES since not only can it be used to parcel out information to players but it can also be used to establish CONFLICTS without the immediate use of COMBAT, e.g. to create INTERNAL RIVALRY within FACTIONS. However, this typically makes it important that players' have IMPERFECT INFORMATION so that is a typically requirement for INFORMATION PASSING. |
| Consequences | | Typically, a game event containing INFORMATION PASSING results in information being gained by players also, although it may not be the same information or only the knowledge that information has been passed. When information is revealed, this may cause SURPRISES and the unfolding of PREDETERMINED STORY STRUCTURES but regardless of this, the event may be sufficient to complete GAIN INFORMATION goals. |
| | | The pattern becomes incompatible with PERFECT INFORMATION whenever INFORMATION PASSING events give information not only to characters but also to players. |
| Relations | <i>Can Instantiate</i> | CLUES, CONFLICTS, GAIN INFORMATION, INDIRECT INFORMATION, INTERNAL RIVALRY, PREDETERMINED STORY STRUCTURES, OPTIONAL GOALS, PREDEFINED GOALS, RED HERRINGS, STEALTH, SURPRISES, UNCERTAINTY OF INFORMATION |
| | <i>Can Modulate</i> | DIALOGUES, FACTIONS, PREDETERMINED STORY STRUCTURES |
| | <i>Can Be Instantiated By</i> | CANNED TEXT RESPONSES, CHAT FORUMS, DIALOGUES, GOSSIP |
| | <i>Can Be Modulated By</i> | AMBIGUOUS RESPONSES, EAVESDROPPING, LIMITED SET OF ACTIONS |
| | | POSSIBLE CLOSURE EFFECTS |
| <i>Potentially Conflicting With</i> | PERFECT INFORMATION | |

Table 4: Gameplay Design Pattern: Information Passing (part II)

3.4 GDPs for Game Dialogues

In the following sections a number of games and game-like applications will be analysed. The selection consists of commercial games and research projects that taken together cover a broad range of game dialogue systems in use. The research projects *ELIZA* (Weizenbaum, 1966) and *Façade* (Mateas & Stern, 2003; Mateas & Stern, 2005) have been included because they are both publicly available as well as interesting from a game dialogue perspective. Furthermore, even though it is not presented as a game, *Façade* contains a complete gameplay structure. Several of the examples are quite old, and there are two main reasons why they still have been chosen: First, they are among the first, or early, examples of games that use dialogue. Second, their relatively limited context and variety make them easier to analyse and describe than the more complex games that have come later. They can in some regard be described as prototypical for a certain kind of game dialogue.

The following questions are investigated:

- What gameplay patterns for dialogues exist in games and other game-like activities?
- What potential gameplay patterns could be suggested on the basis of what we know from experiences in practical dialogue system design?
- How can novel gameplay be created by use of gameplay design patterns associated with natural language interaction?

These patterns have not been described to the level of detail in the original collection but by situating them in the context of dialogue systems and by grounding them in concrete game examples, we want to demonstrate their meaning and potential. In section 3.7 all retrieved patterns are listed together with a definition.

3.4.1 ELIZA

ELIZA (Weizenbaum, 1966) (see chapter 2.1) is a computer program that allows a human agent to have a dialogue with a fictional Rogerian psychiatrist through a text-based interface (an example of P-NPC dialogue, or possibly P-G since the NPC is the only perceivable part of the game). Although it can be seen as a candidate to pass the Turing test (c.f. Saygin, Cicekli, & Akman, 2000, for modern versions of this test), the experience is typically a playful one for users aware of the true nature of the psychiatrist consisting of trying to make the program say illogical or inappropriate utterances. The interaction with the system enforces a turn taking structure similar to TURN-BASED GAMES (Björk and Holopainen, 2005).

After the first welcoming utterance from *ELIZA*, the program hands over the initiative to the user and waits for input, potentially for an infinite amount of time, hence making it a SINGLE INITIATIVE DIALOGUE SYSTEM (see section 2.4.1). Players have the possibility of FREE TEXT COMMUNICATION since they can type whatever they want to the system and the input is then handled through CHUNK-BASED DIALOGUE PROCESSING, i.e., the complete utterance is treated as one input segment. Since conversing is the only activity one can do in the system it is furthermore a trivial example of GAMEPLAY INTEGRATED CONVERSATION, i.e., a game in which conversation is the main gameplay element and where the dialogue is an integral part of the game mechanics, if one considers the activity gameplay.

ELIZA never admits to not understanding input from the user, and can maintain the illusion of always understanding the user through the possibility of responding with context-free questions. This “error-handling” is represented to the user as part of the conversation, and not as part of the interface, which can be described as the design pattern DIEGETICALLY CONSISTENT DIALOGUE, i.e. all utterances are consistent with the represented environment the dialogue takes place within. The program does not try to parse and then figure out the semantics of a user’s input; it relies on transformation rules to create its output based on the user’s input. This means also that *ELIZA* lacks a proper dialogue manager, but produces responses through TEMPLATE-FILLING. Given that the whole experience of interacting with *ELIZA* is in the form of a dialogue, *ELIZA* can be said to make use of the pattern PLAYER CONSTRUCTED WORLD (Björk & Holopainen, 2005), since all specific details of the conversation comes from the players. *ELIZA* can also be said to be able to manipulate a language-game (Wittgenstein, 1953), i.e., a small aspect of a language focusing on a specific activity and the actions related to the activity (see 2.3). Ordinarily this would require a model of the context and the current state of the activity but given that Rogerian psychology consists of mirroring patients’ statements as questions, *ELIZA* can avoid the need for getting a detailed understanding of the overall development of the dialogue.

3.4.2 Zork

Zork (Infocom, Inc., 1980) is a text-based adventure game where the player takes on the role of an adventurer exploring a fantasy environment in the search for treasures. The player controls his avatar through text input and all responses are likewise in text, giving an example of P-PC dialogue in a game. Since nothing happens until input is given, the game is an example of a TURN-BASED GAME (cf. 2.4.2) that uses CHUNK-BASED DIALOGUE PROCESSING. Unlike *ELIZA*, the text given to the system does not represent a conversation with a fictional character but rather instructions for the system on what actions, mental or physical, the player wants the PC to perform. Although these instructions use the representation of the game, i.e. the game’s diegesis, error and ambiguity handling require some of the

dialogue to be on a meta-level. Thus, it does not have DIEGETICALLY CONSISTENT DIALOGUE.

Given that *Zork* presents players with a diegetic game world, *Zork* cannot rely only on transformation rules as *ELIZA* to provide feedback. Instead it needs to relate the player's input to the current state of the game world. Specifically, this takes place on two levels: to determine if it makes sense diegetically (e.g. not trying to pick up a sword that is out of sight), and to generate an appropriate response. Some responses are simple uses of the gameplay design pattern CANNED TEXT RESPONSES, i.e. pre-scripted recorded or text-based utterances (c.f. Jurafsky & Martin, 2000) while other require more or less complicated algorithms. In addition to these types of responses, *Zork* needs to inform players when it does not understand the input as well as be able to handle loading and saving a game, which would correspond to the P-G dialogue in our model.

The way *Zork* uses dialogue for manipulating game components makes it possible to consider utterances during gameplay as illocutionary acts (Austin, 1962). This can be described as the gameplay design pattern ILLOCUTIONARY INTERFACE, since the interface to the game is through communicative acts (see section 2.3.2) regardless of whether the generated actions within the game are represented as dialogue or not. However, text can also be used to direct and communicate with other characters in the game, such as in the sequel to *Zork* where the player could tell the demon to kill the wizard by writing "DEMON, KILL WIZARD". In this case, the player could actually interact both with the game system (or the PC) as well as with other characters in the story through the same interface.

Allowing players free text input to the game parser means that players will probably have to experiment with what input is acceptable and not in the game. Since this interaction is indeed part of gameplay, *Zork* can be said to have the GAME INTERFACE AS PUZZLE.

3.4.3 Grim Fandango

Taking place in the Land of the Dead, the game *Grim Fandango* (LucasArts, 1998a) makes use of the Aztec belief that dead souls wait for four years until they reach the final home: the ninth underworld. The game is a single-player adventure game where the player controls Manny Calavera, who has to pay off the debts he gained from living a "less-than-perfect" life (LucasArts, 1998b) to be able to continue the journey himself. The gameplay takes place through making Manny move around, pick up and interact with objects in the environment, and start talking to characters he meets.

Unlike the previous examples, P-NPC dialogues in *Grim Fandango* are separated from other activities in the game. Talking to an NPC in the game takes the player

into a different GAME MODE (Björk & Holopainen, 2005) where the different phrases that Manny can say are listed in the form of CANNED TEXT RESPONSES that are consumed if selected. The selection of utterances depends on where in the narrative the player is, thus providing the basis for the pattern CONTEXT-DEPENDENT DIALOGUE. Each phrase is furthermore coupled with a pre-recorded sound file that can be interrupted, which could be interpreted as having the pattern BARGE-IN, but as there are no effects of interrupting the disposition it can also be described as an interruptible CUT SCENE. All dialogues are initiated and controlled by the player, thus making use of the SINGLE INITIATIVE DIALOGUE (see 2.4.1) pattern.

Each dialogue is purposeful in some way, either as INFORMATION PASSING (Björk & Holopainen, 2005), to provide DIEGETIC GAME HINTS, or as a key to change the game state. The dialogues in *Grim Fandango* are also important vehicles for conveying the story and creating an atmosphere that is consistent with the stage set. The game characters are presented both implicitly, through for instance RUMOURS, and explicitly by talking to them in person. The dialogues can also be regarded as a way to socialize with the other characters, as some of the choices have less or no impact on game progression but more have the function of supplying the player with a FREEDOM OF CHOICE (Björk & Holopainen, 2005).

As a gameplay challenge, the player must sometimes choose the correct phrasing among a list of paraphrases, as in the following excerpt:

Calavera: Come on, Glottis. I need you to be my driver.

Glottis: I told you... No, I can't. I'm... I'm... I'm too big.

Dialogue 5: Dialogue excerpt from *Grim Fandango*

Player options as a response to Glottis' utterance:

1. You're not too big! You're just right!
2. You're not too big. You just have a self-image problem.
3. You're not too big. The cars are just too small.
4. Screw the rules! Come with me!
5. Alright, back in the shack, mac.

The first three phrases are similar, but it is critical that the player chooses the right attitude in order to succeed. (The right way to approach Glottis is to address his vanity so the correct answer would be number 3, "You're not too big. The cars are just too small"). This type of gameplay option can be referred to as DELICATE PHRASING.

3.4.4 The Elder Scrolls III: Morrowind

The Elder Scrolls III: Morrowind (Bethesda Game Studios, 2002) (*Morrowind* henceforth) is a role-playing game (RPG) played from a first person perspective by a single player. The player starts by building the character, such as specifying

its race, class, gender, skills and a number of additional attributes. The actions the player then chooses to perform will have impact on how the game progresses and how the PC is perceived by the other characters in the game. The game challenges, as in most RPGs, involve solving quests, exploring, fighting, joining guilds, trading and interacting with other characters in the game. A player can approach an NPC to initiate a P-NPC dialogue. When the dialogue interface is activated, the game world freezes, making it a different GAME MODE. This gives consequences to game play such as being able to start a discussion with a guard while being chased by a monster (as long as the guard has not noticed the monster) and having no risk of being attacked until the discussion has ended. Similar to all the previous examples, players initiate and control the dialogue, thus making use of the SINGLE INITIATIVE DIALOGUE (2.4.1) and TURN-TAKING (2.4.2) patterns.

Morrowind uses a dialogue system based on hypertexts connected to a database in which all interface objects containing text are stored, for instance the PC's diary as well as all dialogue content. Dialogue management is thus restricted to selecting the correct database entry for a specific hypertext keyword based on the current game state, location, and the NPC currently addressed, which spawns the new patterns LOCATION-SPECIFIC DIALOGUE, CHARACTER-SPECIFIC DIALOGUE and a generated version of the CONTEXT-DEPENDENT DIALOGUE pattern (cf. elements constituting a social activity in 2.3). The selection is also dependent on the PC–NPC relationship, hence introducing RELATION-DEPENDENT DIALOGUE as a sub-pattern to CONTEXT-DEPENDENT DIALOGUE. Although some of the phrases are diegetically social interaction, nearly all conversations with NPCs are functionally attempts to complete GAIN INFORMATION (Björk & Holopainen, 2005) goals. Further, it is possible to use AFFECTIVE COMMUNICATION through the dialogue choices Admire, Intimidate, Taunt and Bribe to influence the NPC's disposition towards the PC and change preconditions for succeeding with a preferred action. AFFECTIVE COMMUNICATION can be regarded as a special form of affective actions – previously discussed in (Brusk & Eladhari, 2006) and (Eladhari, 2009).

3.4.5 The Elder Scrolls IV: Oblivion

Like its predecessor *Morrowind*, *The Elder Scrolls IV: Oblivion* (Bethesda Game Studios, 2006) (*Oblivion* henceforth) is a single-player computer role-playing game allowing players the freedom of controlling how their characters develop within a rich fantasy environment. Besides activities such as fighting, stealing and casting spells, players can interact with several hundred NPCs through a specialized interface for P-NPC dialogues accessible only when the relation between the PC and NPC is sufficiently good. In contrast to *Morrowind*, *Oblivion* makes use of branching dialogue trees in which the player engages by selecting a pre-scripted phrase, hearing the response and choosing a new phrase. The

selection of phrases depends on location, current relation between the PC and NPC, and status of quests, i.e. the sequel also uses the patterns LOCATION-SPECIFIC DIALOGUE, CHARACTER-SPECIFIC DIALOGUE, RELATION-DEPENDENT DIALOGUE, and CONTEXT-DEPENDENT DIALOGUE. As for *Morrowind*, starting a dialogue puts the player in another GAME MODE and the rest of the game world is paused until the dialogue is finished. In essence, this shows that the dialogue system is a separate system from the main game, both as the actions performed are different as well as the fact that these actions do not take up time in the game world. The dialogue interface also provides access to another interface where players can try to improve (or worsen) the NPC's perception of the PC through a mini-game that is focused upon recognizing facial expressions of the NPC.

Oblivion has previously been analysed for aspects of designing non-playing characters (Lankoski & Björk, 2007a) as well as their social networks (Lankoski & Björk, 2007b). During this process several patterns related to dialogues were identified, including CONTEXTUALIZED CONVERSATIONAL RESPONSES, FREE TEXT COMMUNICATION, GAMEPLAY INTEGRATED CONVERSATIONS, AMBIGUOUS RESPONSES, AWARENESS OF SURROUNDINGS, INITIATIVE, EMOTIONAL ATTACHMENT, ACTIONS HAVE SOCIAL CONSEQUENCES, EAVESDROPPING, SOCIAL NORM, and INFORMATION PASSING, and to a lesser degree OWN AGENDA, SENSE OF SELF, COMPETING FOR ATTENTION, and EITHER YOU ARE WITH ME OR AGAINST ME. Several of these have been mentioned in the earlier examples but for documentation purposes it should be noted that they were first identified in *Oblivion* although not studied in greater detail due to a different focus and space considerations.

The latest game in the Elder Scrolls series, *The Elder Scrolls V: Skyrim* (Bethesda Game Studios, 2011), uses basically the same patterns as *Oblivion*, but during the conversation the shift in game mode is seamless since the NPCs seem to continue with their activities. Also, instead of zooming in on the NPCs during the conversation the visual view remains more or less the same, which gives the illusion that the game supports the pattern GAMEPLAY INTEGRATED CONVERSATIONS.

3.4.6 Façade

As has already been mentioned, *Façade* (Mateas & Stern, 2003) is an interactive drama that uses natural language interaction to manipulate the game.²³ The player plays the role of an old-time friend of the married couple Grace and Trip and is invited to their home for a social visit. Standing outside their door, the PC involuntarily performs some EAVESDROPPING on the couple quarrelling and it becomes obvious that they are having some serious marital problems. This opening serves as a prelude to the plot and depending on how the player acts

²³ The game and associated publications are available for download at <<http://www.interactivestory.net>>.

from now on the story will take different turns. As for *Oblivion*, it has been a case study in an earlier paper (Lankoski & Björk, 2007a) where the patterns EMOTIONAL ATTACHMENT, COMPETING FOR ATTENTION, EAVESDROPPING, AWARENESS OF SURROUNDINGS, and EITHER YOU ARE WITH ME OR YOU ARE AGAINST ME were associated with it. The expressed design goal of *Façade* has been to create a drama in which agents interact socially with each other and with the PC (i.e. both PC-NPC and NPC-NPC dialogues), and where each action performed affects the behaviour and attitude of the other agents (Mateas & Stern, 2003). The agents in *Façade* therefore show that ACTIONS HAVE SOCIAL CONSEQUENCES, i.e. their behaviour and decisions depend upon the other participating agents. Grace and Trip start conversations with the PC to gain sympathy, showing examples of INITIATIVE and OWN AGENDA. Objects in the environment trigger conversation throughout gameplay, showing that the NPCs have EMOTIONAL ATTACHMENT to them and that they have LOCATION-SPECIFIC DIALOGUE as well as CHARACTER-SPECIFIC DIALOGUE, i.e. each character has its own set of possible responses.

In *Façade* the dialogue actually constitutes the major part of the story, and it seems reasonable to define a gameplay design pattern, DIALOGUE-BASED GAME CONSTRUCTION, explaining this type of design decision applied on games. The goal can be described as unlocking a RELATION-DEPENDENT DIALOGUE between Grace and Trip, signifying that the player has changed the context to create the right CONTEXT-DEPENDENT DIALOGUES (but it should be noted that players can set their own goals rather than the culturally implied one of helping the couple solve their marital issues).

As for *ELIZA* and *Zork*, players interact with the system through FREE TEXT COMMUNICATION. Although *Façade* only processes text after the return key has been pressed, i.e., by CHUNK-BASED DIALOGUE PROCESSING, NPCs can initiate actions regardless of what the player is doing and the system therefore has MIXED INITIATIVE DIALOGUES. That the time passes while dialogues take place and the people not involved in the discussion can perform other actions show that *Façade* also has GAMEPLAY INTEGRATED CONVERSATIONS. Furthermore, *Façade* handles MULTI-PARTY DIALOGUES, as all three characters can be engaged in the same dialogue. The size of the group talking is not completely in the player's control: one must consider that all characters can perform EAVESDROPPING. Further, characters can BARGE-IN on others and the NPCs show a SENSE OF SELF in becoming irritated when interrupted.

3.4.7 Mass Effect

Mass Effect (BioWare, 2008) is a space opera game that has received acclaim in the popular press for having a novel dialogue system. Although having responses available due to the presence of character traits (paragon, renegade or neutral) and players choosing type of response rather than exact phrasing have been present in earlier games, e.g. the *Fallout* series (Black Isle Studios, 1997-) and

Morrowind respectively, the quality of the writing and integration of the different parts may explain the positive reception. Another aspect given the positive response may in fact be related to the interface; dialogue options are organized in a pie menu with the same type of responses always appearing in the same place, which simplifies selection.

Mass Effect uses a SINGLE INITIATIVE, TURN-TAKING system. Although the dialogue system can be compared to *Morrowind* or *Oblivion* since they both use the LOCATION-SPECIFIC DIALOGUE, CHARACTER-SPECIFIC DIALOGUE, CONTEXT-DEPENDENT DIALOGUE, and RELATION-DEPENDENT DIALOGUE to CONTEXT-DEPENDENT DIALOGUE patterns, the experience is radically different. The reason is that the options presented in *Mass Effect* specify both what the PC will talk about as well as how, rather than the “database retrieval” style provided in *Morrowind*. Responses are also CANNED TEXT RESPONSES but support DIEGETICALLY CONSISTENT DIALOGUES since the responses are recorded pieces of voice acting with lip-synched avatars.

The dialogue system is in practice a subsystem of the *Mass Effect* game engine with only some information transfer to the overall game state. As such the dialogue system does not support patterns such as EMOTIONAL ATTACHMENT, COMPETING FOR ATTENTION, AWARENESS OF SURROUNDINGS, and ACTIONS HAVE SOCIAL CONSEQUENCES. However, the dialogues revolve around personal backgrounds and romances, which show that the mentioned patterns can occur on a narrative level rather than a gameplay level.

The second game in the *Mass Effect* series, *Mass Effect 2* (BioWare, 2010), uses the same dialogue interface and underlying mechanics. The developers have however rewarded dedicated players of the first game with certain bonuses based on the experience level of the PC and boosts for having played as either paragon or renegade. In the last game of the trilogy, *Mass Effect 3* (BioWare, 2012), players can issue commands and choose dialogue options using voice.

3.5 A Comparative Analysis

After having presented several examples of how dialogues are used in games separately, they may now be compared with each other to see how they relate to models of dialogues and dialogue systems as presented in the previous chapter. By doing so the potential design space between the games can be identified and this provides the basis for the discussion later on how the games could provide novel gameplay through design changes.

Grim Fandango, *Oblivion*, and *Mass Effect* use finite state-based tree-structures (see 2.6.1), where each player choice unfolds the branch of the selected node. *ELIZA*, *Zork*, *Morrowind*, and *Façade* on the other hand do not easily fit any of the models presented in section 2.6. The reason why *ELIZA* does not fit is due to

having a negligible internal state, for nearly all input the system remains in the same state as before.²⁴ *Zork* uses a FORM-BASED-like approach (2.6.2) when trying to disambiguate the user's input but does this for the current game state and not for the whole game. The dialogue engine in *Morrowind* resembles ordinary information retrieval systems that offer some kind of dialogue management, such as the *BirdQuest* system (Flyckt-Eriksson et al., 2003). The agents in *Façade* are built using a PLAN-BASED approach (2.6.3), but the method used differs from ordinary plan-based approaches in that it allows multiple agents (in this case Grace and Trip) to have joint goals and behaviours (Mateas and Stern, 2004).

Although *ELIZA*, *Zork*, and *Façade* have aspects of existing approaches in their design, it might be more correct to state that they have a GAME STATE-BASED APPROACH. That is, even if *ELIZA* and *Façade* are not games *per se*, the state of the dialogue is represented through the complete game state. *Morrowind* fits this model partly as it uses a different gameplay mode for dialogues but has some tree-based structures in it. However, the conditions determining what can be discussed in each conversation depend on the game state (primarily the NPCs' perception of the PC) and most dialogue can be retrieved in a random access manner.

3.5.1 Hypothetical Gameplay Design Patterns

By using the example games and identified patterns as starting point we now discuss how design choices can be transplanted or modified, and identify hypothetical (in the sense that they have not yet been found instantiated in a game) gameplay design patterns. As an initial observation, the examples show a range of possible uses for dialogues as interfaces to games. *ELIZA* and *Zork* use dialogues as the sole way to interact with the game and in the first case it is done completely within the diegesis of the game. *Grim Fandango*, *Morrowind*, *Oblivion*, and *Mass Effect* have dialogues as separate modes in the gameplay with no other activities occurring while the dialogue continues. *Façade* integrates the dialogue into the game interface and lets dialogue and other activities take place at the same time.

Allowing dialogues to take place simultaneously with other activities as *Façade* does, i.e. by using GAMEPLAY INTEGRATED CONVERSATIONS, is a conceptually easy way to change designs. This can add stress and tension, e.g. having to convince town guards in *Morrowind* or *Oblivion* to let the PC enter the city gate while monsters are approaching. Besides making verbal expressiveness a potential game skill, this also can cause speed of typing to be important for players (assuming that the dialogue is text-based rather than voice-based). It should be noted that this idea

²⁴ This is not entirely true since *ELIZA* in fact does need to detect and store keywords in the player's input in order to use it in the response.

can be applied in the case of *Façade* but at a higher level of detail; players can be challenged to provide NPCs with proper BASIC INPUT FEEDBACK by redesigning the system to support INCREMENTAL INPUT PROCESSING (see 2.4.3), i.e. handling input per key strokes or tokens separated by blanks instead of per complete utterances. The romantic and intimidating aspects of dialogue in *Mass Effect* could likewise be expanded with requiring players to have appropriate body language and distance to achieve the desired effect. To integrate game dialogues with the overall gameplay, one has to equate communicative actions with other game actions as well as provide a support for these actions to be performed simultaneously. In *Façade*, for example, they developed “A Behaviour Language” (ABL) to accomplish this (Mateas and Stern, 2004).

The personality of NPCs in *Grim Fandango* and *Oblivion* are expressed through dialogues, and in *Façade* and *Mass Effect* the character’s emotional state can be perceived as well. These features can be further explored to also include social behaviour, dependent on the characters’ interpersonal relationship and the role they play in the situation in question.

The games examined make use of dialogues for gameplay in different ways. In *ELIZA* the dialogue is the gameplay (in the sense that there is gameplay at all) while for *Zork* it is the interface that makes gameplay possible. *Grim Fandango*, *Morrowind*, *Oblivion*, and *Mass Effect* use dialogues to provide information to players about the game world and to progress the various plots. In one sense they all use dialogues as ILLOCUTIONARY INTERFACES since they can either change the game world or the progress of a narrative structure, but this may not be apparent to players before they make utterances. Taking an extreme view, one could finish any of these games without understanding the dialogues in the games. For *Zork* and *Façade* one would at least have to parse out the important words used, i.e. USE GAME INTERFACE AS PUZZLE. Interacting with *ELIZA* without understanding the language the system responds in is unlikely to give a meaningful experience for any longer period of time.

Zork and *Façade* show that requiring players to perform actions that express an understanding of the dialogue is one way of integrating dialogue and gameplay. For *Zork* this consists mainly of figuring out what verbs, adjectives, and nouns can be used in the interface. In *Façade* it is manifested through the vagueness of the goal and the openness of how players can express themselves.

The examples listed above may be perceived as only applicable to natural language interactions, but scripted dialogues can also be improved by allowing a wider range of utterance options in specific situations. These options may be available based on for instance the dialogue history, the character’s internal state as well as the interpersonal relationship. One example of a more complex system that uses a dialogue menu combined with CANNED TEXT RESPONSES is the Augmented Conversation Engine (Swain, 2008). In this system, the response

from the NPC is selected from a matrix of trust combined with randomness, which summarizes to about twelve different possible answers to a specific user input. The problem with this system is however that the number of player options available on the screen soon becomes overwhelming and tend to take up most part of the screen.

The combat system in *Monkey Island* (Lucasfilm Games, 1990), having to know the right insults to verbally defeat one's opponent, provides another alternative. This can be generalized into the pattern, COLLOQUIAL MASTERY, i.e. one has to learn the use of the language beyond simple information transferral so that one masters the idiosyncrasies of the current environment. For example, soldiers may need to begin and end every sentence with "sir" and outlaws may need to add curses to impress NPCs to fit the SOCIAL NORM. A technically more challenging option is to require players to use DELICATE PHRASING, formulating utterances without revealing sensitive information or causing insults. In this fashion, game dialogues can challenge players' skills in expressing themselves – either as a standalone game or as part of the overall gameplay.

3.6 Concluding Remarks

In chapter 2 speech act theory was introduced, which accords well with gameplay design. Searle (1969) bases his work on speech acts on the hypothesis that that "speaking a language is engaging in a rule-governed form of behavior" (p. 22) and just as "we can translate a chess game in one country into a chess game of another because they share the same underlying rules, so we can translate utterances of one language into another because they share the same underlying rules" (p. 40). Searle also makes a distinction between regulative rules, i.e., those rules that regulate the interpersonal relationship, for example rules of etiquette; and constitutive rules, explicit rules that define or create new forms of behaviour. In a similar manner Salen and Zimmerman (2004) have described game rules at three levels: operational rules, constitutive rules, and implicit rules. The operational rules are the description given to the players that specifies the "guidelines" for playing the game. The constitutive rules are, similar to Searle's description, "the underlying formal structures". The implicit rules correspond to Searle's regulative rules in that they define the "unwritten rules" of the game, which regulate things such as cheating and good sportsmanship.

Interacting with a game system is conceptually the same as interacting with another agent (human or artificial). A player performs an action and the game system responds to that action in some way. In a similar way, a speaker performs speech acts and receives responses to these actions. By treating the player's communicative acts on a par with other game actions, it is possible to achieve GAMEPLAY INTEGRATED DIALOGUES.

In this chapter an analysis of a number of games and game-like activities have been presented with regard to their dialogue systems by exploring how common dialogue features can be applied in games. Suggested uses have come from established techniques used for designing dialogue systems (see for example section 2.4 and 2.6) as well as by identifying uses in some examples and generalizing them. The specific suggestions have been identified as gameplay design patterns although no full descriptions of these have been given.

Even though natural language interaction adds possibilities and advantages in the design of the user interface, it also introduces new problems and difficulties. For example, the system may fail to create an interpretation or create an erroneous one, and repeated failures in understanding may cause frustration for the player. Some suggestions to decrease the potential negative effects of NLI include having a range of possible input modalities, letting the system choose reactions based on the global state as *Façade* does by using a beat system, (see e.g. Mateas & Stern, 2003), or trusting players' tendency to try to find meaning in utterances as *ELIZA* does. *FREE TEXT COMMUNICATION* gives players a large degree of freedom but it may still be important to make the range of reasonable options obvious to players (see 3.2.2). Another alternative could be to adapt the NPC's behaviour and match it to the player's behaviour or just let the system take control occasionally and help the player to learn how to interact. These solutions would make the NPCs behave cooperatively in the sense that they take the player (character) into cognitive consideration and make an attempt to understand and perceive the player's action (see section 2.3.3). However, they could still behave uncooperatively in other respects.

Despite such problems, Sali et al. (2010) presents a study showing that NL dialogue interfaces (NLI) are the most appreciated form of dialogue system when compared to the abstract form used in *Mass Effect* and the verbatim dialogue types such as the one used in *Oblivion* and *Grim Fandango*. The subjects preferred the NLI, even though they also found it the most difficult to use and received the lowest scores when measuring the player's sense of control. The NLI was also regarded as the most engaging of all the interfaces, indicating that engagement and appreciation are correlated to some extent. The story involvement was lower when using NLI, suggesting that story was less important for the players in the study in comparison to engagement.

Several new design patterns were identified through analyses of existing games (see the complete list in section 3.6, below). Further analyses of hypothetical re-designs (see 3.5.1) have provided the basis for the new patterns including *DELICATE PHRASING*, *COLLOQUIAL MASTERY*, and *INCREMENTAL INPUT PROCESSING*. These patterns have not been described to the level of detail in the original collection but by situating them in the context of dialogue systems and by grounding them in concrete game examples, the meaning and potential are hopefully evident.

Regarding the applicability of models presented in section 2.6 to games, the examples did not show perfect matches to existing categories. This may be due to specialization to the applications in question, but also due to the presence of characteristics typically not discussed in computational linguistics. An alternative approach, the GAME STATE-BASED APPROACH, has been introduced as has the observation that the intention of dialogue systems for games can differ from other dialogue systems in that cooperativeness is not always the intended design goal (see section 2.5.1).

One of the complexities associated with natural language understanding in story-based games is that the system must be able to handle a variety of conversation types as well as domains, and a range of unexpected user inputs. When faced with complex problems it is usually a good idea to break it down into smaller problems – to “divide-and-conquer”. Likewise, a dialogue management component can be broken down into smaller modules that can be combined to solve more complex problems. In the following chapter a method for creating modular based applications for game dialogue design will be presented, which in the chapters thereafter is used for implementing some of the patterns presented in the following section.

3.7 Collection of Patterns

Table 5 to Table 7, present a selection of the patterns mentioned in this chapter. Gameplay design patterns that have been defined in earlier studies will only be listed here if they occur in several examples or signify an important feature of game dialogues.

| New Patterns | |
|----------------------------------|--|
| Name | Definition |
| AFFECTIVE COMMUNICATION | Communicative act with the sole purpose of emotionally affecting the addressee |
| BARGE-IN | The possibility of interrupting a speaker |
| CANNED TEXT RESPONSES | Dialogues having pre-scripted recorded or text-based utterances |
| CHUNK-BASED DIALOGUE PROCESSING | Dialogue processing in which the complete utterance is treated as one input segment |
| CHARACTER-SPECIFIC DIALOGUE | Character has a unique set of possible responses |
| CONTEXT-DEPENDENT DIALOGUE | A dialogue that is dependent upon where in the narrative the dialogue takes place |
| DELICATE PHRASING | Dialogues requiring the player to choose the right phrasing in order to succeed |
| DIEGETICALLY CONSISTENT DIALOGUE | A dialogue in which all utterances are consistent with the represented environment the dialogue takes place in |

| New Patterns | |
|----------------------------------|--|
| Name | Definition |
| DIETETIC GAME HINTS | Hints provided by NPCs that help the player move forward in the game, but that are thematically within the game diegesis |
| DIALOGUE-BASED GAME CONSTRUCTION | Games design based primarily on dialogue |
| FREE TEXT COMMUNICATION | An interface that accepts natural language text input |
| GAMEPLAY INTEGRATED CONVERSATION | Conversation is an integral part of the game mechanics |
| GAME STATE-BASED APPROACH | The state of the dialogue is represented through the complete game state |
| ILLOCUTIONARY INTERFACE | Interface to the game is through communicative acts regardless of whether the generated actions within the game are represented as dialogue or not |
| LOCATION-SPECIFIC DIALOGUE | Character's set of responses is dependent upon the location where the dialogue takes place |
| MIXED INITIATIVE DIALOGUE | A dialogue in which either participant may direct and control the dialogue. Control may also switch during the course of the interaction |
| MULTI-PARTY DIALOGUES | Dialogues in which several participants are engaged and can contribute |
| RELATION-DEPENDENT DIALOGUE | Character's set of responses is determined by its relation to the other participating character(s) |
| RUMORS | Unconfirmed second hand information spread among the game characters |
| SINGLE INITIATIVE DIALOGUE | A dialogue in which one of the participants (typically the system) directs and controls the dialogue |
| TEMPLATE-FILLING | Dialogue systems lacking a dialogue management component and in which the system instead produces responses by filling templates |

Table 5: New Gameplay Design Patterns for Dialogue

| Patterns Found in Earlier Studies ²⁵ | |
|---|---|
| Name | Definition |
| ACTIONS HAVE SOCIAL CONSEQUENCES | The behaviour and decisions of a character depend upon the other participating character(s) |
| EAVESDROPPING | The ability to overhear other characters' conversations |
| EMOTIONAL ATTACHMENT | The ability of agents to have noticeable emotional relations inside the game world to the diegetic phenomena in that world. |
| INITIATIVE | The ability of agents to take actions not directly perceived as the consequence of game events. |

²⁵ See collection of patterns at <<http://gdp2.tii.se/index.php?title=Category:Patterns&until=No-Use+Bonus>>.

| | |
|-----------------|---|
| OWN AGENDA | The ability of agents to seem to strive towards personal goals. |
| SENSE OF SELF | The ability of agents to react to events concerning it or to its internal states. |
| TURN-BASED GAME | The players take turns to make their actions to change the game state, and the progress of time is not tied to the real time. |

Table 6: Gameplay Design Patterns for dialogues found in earlier studies

| Hypothetical Patterns | |
|------------------------------|--|
| Name | Definition |
| BASIC INPUT FEEDBACK | The ability to provide continuous feedback during conversation |
| INCREMENTAL INPUT PROCESSING | The ability of agents to process natural language input continuously, for example per key strokes or tokens separated by blanks, instead of after the complete utterance has been formulated |
| COLLOQUIAL MASTERY | The ability of agents to learn the use of the language beyond simple information transferral in order to master the idiosyncrasies of the current environment |
| SOCIAL NORM | The ability of agents to behave according to a specific norm set for a particular social situation |

Table 7: Hypothetical Gameplay Design Patterns for Dialogue

Chapter 4

Technological Framework

As has already been announced, the implementations presented in this thesis have been developed using SCXML and to some extent VoiceXML. This chapter introduces these technologies to the reader and follows up the rule-based approaches presented in chapter 2.6. A detailed description of Harel statecharts (Harel, 1987) will also be given as they form the semantics of SCXML. The last part of the chapter discusses other advantages in using Harel statecharts. Initially, however, an introduction to the Data Flow Presentation (DFP) framework will be given.²⁶

4.1 The DFP Framework

The World Wide Web Consortium (W3C) is an international organization that develops technical specifications and guidelines that “ensure the long-term growth of the Web” (W3C mission²⁷). Standards are free, accessible, stable, distributable (i.e., it works on different browsers and has backward and forward compatibility), can easily be validated, and provide consistency.

In 2006, W3C introduced the Data Flow Presentation (DFP) framework²⁸, where “computation and control flow are kept distinct from application data and from the way in which the application communicates with the outside world” (McGlashan et al., 2010). The DFP framework is an instance of the Model-View-Controller pattern (Burbeck, 1992) and consists similarly of three separate layers:

Data – A component that manages the data for the application. The data can for instance be the status of the dialog in collecting certain information, which prompts have just been played, and how many of various error conditions have

²⁶ Parts of this chapter has previously been published in (Brusk & Lager ,2008).

²⁷ <<http://www.w3.org/Consortium/mission.html>>.

²⁸< <http://www.w3.org/Voice/2006/DFP>>.

occurred so far, and the values entered by the user until they are transmitted to the back-end database or file system.

Flow – A component that controls the application flow by interacting with data and presentation layers. The flow layer is not intended to interact directly with the user. Rather, it requests user interaction by invoking a presentation component running in parallel with the SCXML process, and communicating with this component through asynchronous events.

Presentation – Components of the presentation layer interact with the user. Presentation components may support modalities of different kinds, including graphics, voice or gestures. VoiceXML 3.0, for example, is designed with the DFP Framework in mind.²⁹

The web standards and working drafts presented here are in particular two of the languages that have been chosen to specify the different layers of the Data Flow Presentation (DFP) Framework: SCXML and VoiceXML.³⁰

It should be noted that SCXML has evolved during this thesis project and that the implementations listed in the appendices were developed in accordance with the current version of SCXML when the programs were written. This means that the examples vary in syntax, but hopefully not that much that they will not make sense. Furthermore, there is still a lack of platforms and development environments for SCXML, but we have had the opportunity to use a platform developed by Spyderbrain Technologies³¹, a platform, which also has been under development during this time period. The code samples presented in this chapter have however been adjusted to the latest working draft, which according to W3C is the final one.

4.2 VoiceXML

VoiceXML is a framework for building interactive voice applications.³² It is part of the web-based voice infrastructure, which makes it possible to allow VoiceXML documents to access the Internet to exchange information with web

²⁹ It should be noted that the DFP document dates back to 2006 and no later reports have been published. Also, the latest working draft of VoiceXML 3.0 was published in 2010. It is therefore not clear exactly where W3C will take this. It should also be noted that W3C in collaboration with Google recently has released the WebSpeech API, which might make the DFP framework obsolete. However, VoiceXML 2 is an already existing standard and SCXML is in the final round of becoming one.

³⁰ The presentations of SCXML and VoiceXML 3.0 are based on the working drafts released in August 1, 2013 and December 16, 2010, respectively.

³¹ <http://spyderbrain.ling.gu.se/web_labs.html>.

³² VoiceXML is a World Wide Web Consortium (W3C) standard. The recommendations can be found here: <<http://www.w3.org/TR/voicexml20/>>.

sites and other Internet servers. W3C estimates that about 85% of all IVRs use VoiceXML.³³

A VoiceXML application consists of a set of VoiceXML documents, where each document forms a conversational finite state machine (McGlashan, et al., 2004). Each conversational state corresponds to a dialogue and the user can only be in one dialogue at a time. There are two types of dialogs in VoiceXML: *forms* and *menus*. Forms present information and gather input, similar to how forms in traditional web-based applications work. A form can contain one or more fields, each of which may specify a grammar that defines the allowable inputs for that field. The following example implements the finite state machine illustrated in Figure 9 (see section 2.6.1):

```
<vxml version="2.1">
  <form id="travel">
    <grammar src="from_to.grxml"
             type="application/srgs+xml"/>
    <field name="from">
      <prompt> What planet are you leaving from? </prompt>
    </field>
    <field name="to">
      <prompt> To which planet? </prompt>
    </field>
    <nomatch count="1">
      Sorry dude, but I didn't get that.
    </nomatch>
    <nomatch count="2">
      I'm sorry, I still don't understand.
    </nomatch>
  </form>
</vxml>
```

Dialogue control for filling a form in VoiceXML is managed through the Form Interpretation Algorithm (FIA), which combines the finite state-based and frame-based approaches and was previously described in chapter 2. FIA visits each unfilled form element in document order and collects the value of the user's input, one at a time. If the dialogue contains the control element `<initial>`, FIA can fill any of the matching slots from the user's input at the initial prompt as was mentioned in section 2.6.2. For this reason, VoiceXML is regarded as being able to support mixed initiative dialogues (McTear, 2004). If there are remaining slots to fill after `<initial>` has done its task, FIA revisits each of them in document order until it interprets a transfer of control statement, e.g. a command to go to another document or submitting data (McGlashan, et al., 2004). This also means that if FIA is unable to fill any slot from the initial input, it will start filling the form from the very beginning.

³³ <http://www.w3.org/standards/webofdevices/voice>.

The code snippet below shows how the previous example may be modified to handle mixed initiative using the `<initial>` element. The example also makes use of the built-in error handling provided in VoiceXML for no match events (`<nomatch>`) (see also the specification in Figure 9). The result may be read as follows. Initially, the player is expected to respond with (say) “From Mars to Venus”, in order to fill both fields in one shot. However, if the response is not recognized by the ASR (using the specified grammars) after two attempts, the dialogue engine tries instead to split the initial question into two parts, expecting responses such as “Mars” or “Venus”, filling one field at a time, as was the case in the first example:

```
<vxml version="2.1">
  <form id="get_from_and_to_planets">
    <grammar src="from_to.grxml"/>
    <initial name="bypass_init">
      <prompt>Fly from and to where?</prompt>
      <nomatch count="1">
        Sorry dude, but I didn't get that.
      </nomatch>
      <nomatch count="2">
        I'm sorry, I still don't understand.
        <assign name="bypass_init" expr="true"/>
        <reprompt/>
      </nomatch>
    </initial>
    <field name="from_planet">
      <prompt>From which planet?</prompt>
    </field>
    <field name="to_planet">
      <prompt>To which planet?</prompt>
    </field>
  </form>
</vxml>
```

4.2.1 SRGS and SISR

In the examples above, the grammar references pointed to an external file (using the extension “grxml”). Grammar is specified using one of the forms of the W3C Speech Recognition Grammar Specification³⁴ (SRGS) – ABNF (Augmented Backus–Naur Form) or XML. The examples given in this thesis have used XML syntax for parsing the user’s utterance into raw text format. However, SRGS also have syntactic support for a semantic interpretation of the input by means of the *tag* construct. How to specify the content of the tag element and then access the semantic information is defined in the W3C’s specification Semantic Interpretation for Speech Recognition³⁵ (SISR).

³⁴ <<http://www.w3.org/TR/speech-grammar/>>.

³⁵ <<http://www.w3.org/TR/semantic-interpretation/>>.

A grammar specifying acceptable planet destinations for the planet traveller could for example be as follows:

```
<grammar version="1.0" root="planet">
  <rule id="planet">
    <one-of>
      <item>Mars</item>
      <item>Pluto</item>
      <item>Venus</item>
    </one-of>
  </rule>
</grammar>
```

In this case, no semantic interpretation other than the actual input is given. In this particular example it would be redundant as we have three distinct answers with different meanings. However, if the grammar should accept different expressions for one and the same meaning, such as “hello” and “hi”, a semantic specification is needed. To exemplify, we revisit our waiter character from chapter 1. Since the dialogue manager accepts communicative acts as input, the parser needs to be able to interpret the player’s input as a particular communicative act. The grammar specified below allows the player to greet the waiter or order something from the menu (e.g. “I would like a coffee”). The example is stripped down to highlight the essentials.

```
<grammar version="1.0" root="start">
  <rule id="start">
    <one-of>
      <item> <ruleref uri="#greet"/>
        <tag>out.ca='greet';</tag></item>
      <item> <ruleref uri="#order"/>
        <tag>out.ca='order';
          out.drink=rules.drink; </tag></item>
    </one-of>
  </rule>
  <rule id="greet">
    <one-of>
      <item> hi </item>
      <item> hello </item>
    </one-of>
  </rule>
  <rule id="order">
    I would like <ruleref uri="#drink"/>
  </rule>
```

```

<rule id="drink">
  <one-of>
    <item>a coffee <tag> out='coffee';</tag></item>
    <item>a cup of tea <tag> out='tea';</tag></item>
  </one-of>
</rule>
</grammar>

```

In this example the tag construct is used to return the value of the type of communicative act (“ca”) that was performed – greet or order – as well as a specification of the order (coffee or tea). Later sections will clarify how this information can be used for managing the dialogue.

Tags can also be used to assign synonyms the same value, so that “yeah”, “ok”, “yupp” and “yes” all can be treated as a “yes” for example. The rule “greet” treats “hi” and “hello” as the communicative act “greet” and makes no semantic distinction between the two alternatives.

Going into more detail about VoiceXML, SRGS, and SISR is beyond the scope of this thesis. Suffice it to say that VoiceXML is a well-proven and mature technology for the design and implementation of practical dialogues.

4.3 State Chart XML (SCXML)

SCXML can be described as an attempt to render Harel statecharts (Harel, 1987) in XML. Harel (1987) introduced a number of (at the time) novel extensions to finite-state machines, including hierarchy, history, concurrency, and broadcast communication. In the following sections these features will be described in more detail supported by game-related examples.

Let us now begin with a very simple example in the form of a ‘game’ where a display will show “YOU WON! Play again?” if the player pushes a Play-button, and then “Push to Play” if he pushes the Again-button. Not much fun, but it serves to introduce some notation. The statechart controlling our game could simply be:

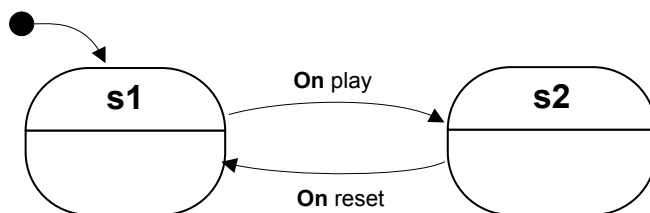


Figure 15: Game Statechart

Any statechart can be translated into a document written in the linear XML-based syntax of SCXML. Here, for example, is the SCXML document capturing the above statechart, and thus the logic of our simple game:

```
<scxml initial="s1">
  <state id="s1">
    <transition event="play" target="s2"/>
  </state>
  <state id="s2">
    <transition event="reset" target="s1"/>
  </state>
</scxml>
```

The document can be executed by an SCXML conforming processor, thus shortening the step from a game specification into a running game application that needs to be taken.

4.3.1 Hierarchy and History

As has been mentioned earlier, statecharts may be hierarchical, i.e. a state may contain another statechart down to an arbitrary depth. From a methodological point of view this is important, since it allows us to apply the principles of refinement (a top-down design process in which a state is refined into a number of sub-states and the transitions between the sub-states spelled out in detail) and clustering (a bottom-up design process in which a number of similar states are grouped together under the umbrella of a super-state). These principles are very general, and they are certainly relevant to game design as well.

To hierarchically group states reduces the number of transitions considerably since the sub-states can share transitions further up in the hierarchy rather than having separate transitions to the same target state. States that represent complex behaviours can also be decomposed into smaller, more understandable and maintainable units by use of hierarchy. Fu and Houlette (2004), for example, recommend the use of hierarchical FSMs for specifying the character AI in games for the following reasons:

Hierarchical FSMs have a couple of advantages that tend to make their added complexity worthwhile. For one thing, they allow large, complex character behaviours to be broken down into smaller, more understandable, and more maintainable chunks. As your FSMs grow and become more complex, the ability to refactor and decompose FSMs in this way can be invaluable. In addition hierarchical FSMs allow you to avoid duplication of FSM code (or script), since you can extract common sub-behaviours out into separate FSMs that can be referenced by many other FSMs. (Fu & Houlette, 2004, s. 297)

A complex state may contain a history state (H), which is a pseudo-state serving as a memory of which sub-state the complex state S was in, the last time it was left for another state. Transition to the history state implies a transition to S . The history state can be either deep (specified by an asterisk, H*) or shallow. If the history is deep, the target state will be the last visited state at the deepest level, whereas the shallow version will only find the state one way down in the hierarchy of the previous configuration.

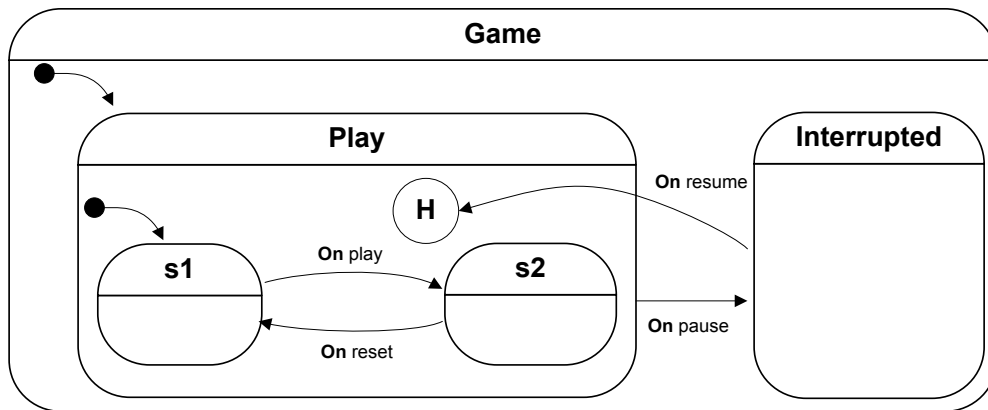


Figure 16: Pause-and-resume

Figure 16 shows an example of a game that can be played, paused and then resumed again. The compound state `Game` contains two sub-states: `Play`, which also is a compound state, and the atomic state `Interrupted`. The state `Play` starts in state `s1` where an event "play" is expected that will trigger a transition to `s2`. Suppose that the current state is `s2`, and that an event "pause" appears in the event queue. The transitions leaving `s2` are tried from the inside and out, and since "reset" does not match the first event in the queue, but "pause" does, a transition to the "Interrupted" state takes place. If a "resume" event shows up in the queue, the system transfers to the history state `H`, which implies a transition back to the `s2` state again.

The SCXML documents corresponding to Figure 16 would be:

```
<scxml initial="Game">
  <state id="Game" initial="Play">
    <state id="Play" initial="s1">
      <state id="s1">
        <transition event="play" target="s2"/>
      </state>
      <state id="s2">
        <transition event="reset" target="s1"/>
      </state>
      <history id="H"/>
      <transition event="pause" target="Interrupted"/>
    </state>
    <state id="Interrupted">
      <transition event="resume" target="H"/>
    </state>
  </state>
</scxml>
```

The attribute `initial` specifies the default start state for nested states.

4.3.2 Concurrency

Two or more statecharts may be run in parallel, which basically means that their parent statechart is in two or more states at the same time. This is an important mechanism for introducing independency and orthogonality into a design. It is well-known (cf. Harel, 1987; Horrocks, 1999; Fu & Houlette, 2004) that concurrency and to a some extent also hierarchy are the means by which this often cited problem with ordinary finite state machines – the exponential growth in the number of states and transitions – can be treated.

Concurrency may for example be useful when the flow of a game is not (only) modelled directly (or scripted), but when the NPCs – their states-of-mind, states-of-body, as well as their (verbal and non-verbal) behaviours – are modelled in the hope that a good game will emerge from the interaction between different NPCs and between NPCs and human players. In such cases it makes sense to model each NPC as a separate statechart, running in parallel with each other, and running in parallel with a statechart modelling the environment.

Just to give a hint of how this might look like in SCXML, we give a high-level view of an architecture where an NPC's 'emotions' and its reactive behaviours are working independently. First as a statechart in the graphical notation (Figure 17) and then translated into SCXML:

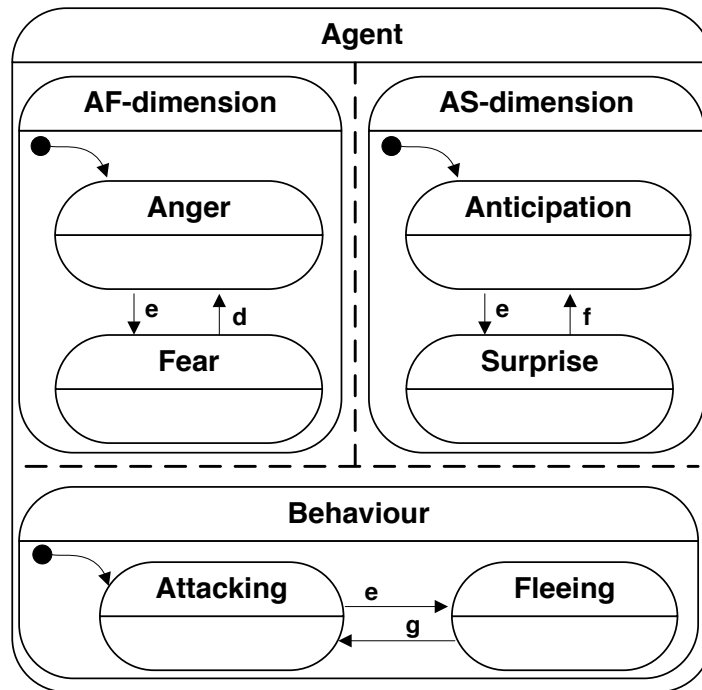


Figure 17: Emotion and behaviour

```

<scxml initial="agent">
  <parallel id="agent">
    <parallel id="emotions">
      <state id="AF-dimension" initial="anger">
        <state id="anger">
          <transition event="e" target="fear"/>
        </state>
        <state id="fear">
          <transition event="d" target="anger"/>
        </state>
      </state>
      <state id="AS-dimension" initial="anticipation">
        <state id="anticipation">
          <transition event="e" target="surprise"/>
        </state>
        <state id="surprise">
          <transition event="f" target="anticipation"/>
        </state>
      </state>
    </parallel>
    <state id="behaviour" initial="attacking">
      <state id="attacking">
        <transition event="e" target="fleeing"/>
      </state>
      <state id="fleeing">
        <transition event="g" target="attacking"/>
      </state>
    </state>
  </parallel>
</scxml>

```



```

<state id="behaviour" initial="attacking">
  <state id="attacking">
    <transition event="e" target="fleeing"/>
  </state>
  <state id="fleeing">
    <transition event="g" target="attacking"/>
  </state>
</state>
</parallel>
</scxml>

```

The idea is that in the event of (say) an explosion (the event named “e”) the NPC will end up in the states of fear, surprise and flight behaviour. Then, if his friend dies (the event named “d”), fear will turn into anger, and he will attack instead. Note that if we did not have access to concurrency, we would have had to distinguish the atomic state `FearSurpriseFleeing` from other atomic states such as `AngerSurpriseFleeing`, `AngerAttackFleeing` and so on. And as soon as we wanted to extend (say) the emotional dimensions from two into (say) three, we would see a large increase in the number of states and transitions required.

Another way to coordinate the behaviours in concurrent states is to check the active state of a parallel process by use of the “In()” predicate, which takes a state id as its argument and returns true if the state machine is in that state (Barnett, et al., 2012). One of the first examples given in this thesis (see Figure 3) used this predicate to test whether the waiter was in the state `TakeOrderAct` – a substate of `ActionAM` – which would trigger the dialogue to be initiated if evaluated to true.

4.3.3 Broadcast Communication

One statechart S_1 may communicate with another statechart S_2 (running in parallel with S_1) by placing, in the global event queue, an event that triggers a transition in S_2 . Since the event in principle can be detected by any transition in any state in the statechart, this is often referred to as “broadcast communication”.

We exemplify parallel statecharts and the communication between their substates with an SCXML document featuring two ‘agents’ playing ping-pong.

```

<scxml initial="start">
  <parallel id="start">
    <state id="Pinger">
      <onentry>
        <send event="ping"/>
      </onentry>
      <transition event="pong">
        <send event="ping" delay="1s"/>
      </transition>
    </state>
    <state id="Ponger">
      <transition event="ping">
        <send event="pong" delay="1s"/>
      </transition>
    </state>
  </parallel>
</scxml>

```

Note the use of ‘targetless’ `<transition>` elements here, where the matching of an event results in the running of the transition’s actions (executable content such as the `<send>` elements) but not in any actual transitions. Note also that we have delayed each sending of an event with one second, just to make the speed of the ping-pong game a bit more realistic.

4.3.4 Data Model

SCXML gives authors the ability to define a data model as part of an SCXML document. A data model consists of a `<datamodel>` element containing one or more `<data>` elements, each of which may contain a XML description of data. The value of the ‘cond’ attribute in a `<transition>` element may be an expression referencing the data, and transitions may thus be conditioned on the data. The `<assign>` element may be used in actions, or in `<onentry>` or `<onexit>` elements, to modify the data. A data element always consists of the attribute `id`, specifying the name of the data item, and optionally a value specified by the attribute `expr` (for references to data stored outside the SCXML document, `src` can be used to locate the value of the current data item). As a simple illustration, the following state (which could be one of several parallel states) serves as a score counter, transferring to the “GameOver” state when the count is 3, and incrementing the count each time the “point” event shows up in the global event queue.

```

<scxml initial="Scorer">
  <state id="Scorer">
    <datamodel>
      <data id="Score" expr="0"/>
    </datamodel>
    <transition event="point">
      <assign location="Score" expr="Score + 1"/>
    </transition>
    <transition cond="Score == 3" target="GameOver"/>
  </state>
  <final id="GameOver"/>
</scxml>

```

4.3.5 External Communication

SCXML also has support for communicating with external services, which is very valuable for many reasons. Consider for example *Siri* (see 2.1), in order to accommodate the user's request she must communicate with one or several external applications first and to accomplish this, the dialogue system must have support for that type of communication.

SCXML offers two different ways of communicating with external processes, `<send>` and `<invoke>`. `<send>` is an element that can fire away an event to either another SCXML process or some other process. A "delay" attribute allows the delivery of the event to be postponed the specified amount of time.

The `<invoke>` construct offers a tighter way of communication as it enables the invocation of external processes, for instance a VoiceXML process or another SCXML process, from within an SCXML session. An invoked process runs in parallel with the invoking process and does not share data with it unless explicitly requested through the `<param>` and `<content>` elements as well as the "src" and "namelist" attributes. The processes can communicate by sending and receiving events, but in order for the invoked process to get access to the events that appear on the global event queue of the invoking process, its "autoforward" attribute must be set to true. The invoking process can also send events to the invoked process by using the `<send>` element. The id of the invoked process must be specified in the target using the special form "#_invokeid". An invoked SCXML process may also use the `<send>` element to communicate with the parent session. In that case the target must be specified using the form "#_parent".

4.4 SCXML in the Bigger Picture

SCXML is intended to control the flow of an application, be it a game or a (multimodal) dialogue system. It is not intended to manage lots of data, or to directly interact with the user. In this section we will look closer at one particular kind of user interaction – dialogue using natural language and in particular dialogues between the player and an NPC in the game.

Conversations between humans often are of the multimodal kind, and so should be realistic dialogue with NPCs. An NPC should be able to nod instead of saying “yes”, or nod and say “yes” at the same time. Thus, the boundary between controlling the visual appearance and behaviour of an NPC – how it looks and what it does – and its natural language capabilities – what it says – is not very clear-cut. Therefore there are advantages in controlling and synchronizing them using one and the same mechanism.

As we have indicated already, SCXML is not supposed to directly interact with the user. Rather, it requests user interaction by invoking a presentation component running in parallel with the SCXML process, and communicating with this component through asynchronous events. Presentation components may support modalities of different kinds, including graphics, voice or gestures. Concentrating on presentation components for spoken language dialogue (a.k.a. “voice widgets”) we may assume that they include things like:

- A Text-To-Speech (TTS) component for presenting the player with spoken information.
- An Automatic Speech Recognition (ASR) component to collect spoken information from the player.
- A combination of TTS and ASR to implement something akin to a field, prompting for, and collecting, a value of one single parameter from the user.
- A form-filling algorithm (a.k.a. FIA – the Form Interpretation Algorithm) running over an (internal) datamodel, and using TTS and ASR for output and input, respectively, and thus implementing something akin to a form, collecting values for a set of parameters from the user.
- Other external interaction components, tailored to particular conversational modes, e.g. social talk.

Note that presentation components may be simple, as the first two components in the above list, or complex, as the last three. (The very last one is probably very complex.) A complex component may be made up of other (simple or complex) components (perhaps using SCXML for controlling the interplay between their parts, perhaps not), but for all intents and purposes their complexity is hidden

from the developer, and the only way to communicate with them is through the global event queue that they share with the invoking SCXML document.

In a game setting it could be useful to treat each level as a separate process invoked from a meta-game controller, and each agent – NPCs as well as the PC(s) – within that level to be invoked from within the active game level. Since invoked processes have private data models, the agents will have their own internal memory, which means that the agents’ common ground will be based entirely on the information that is available in the shared environment. Using statechart notation it could be visualized as Figure 18, below. The PC and NPC are here invoked processes and as can be seen, they have separate data models.

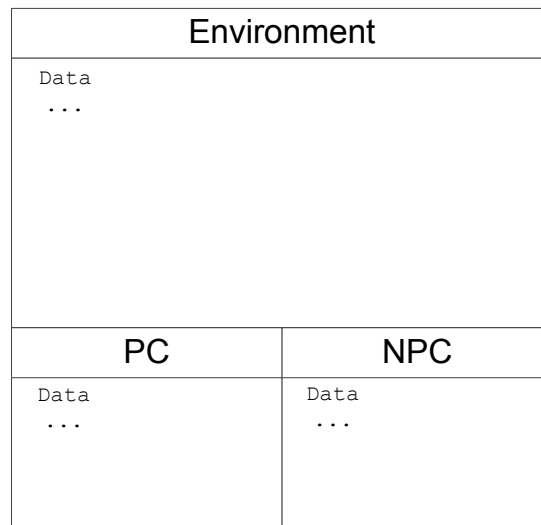


Figure 18: Game environment with invoked characters

As long as the invoked agents have set the “autoforward” attribute to true, they will have access to all events appearing on the global event queue of the environment. The agents may thus communicate by sending and receiving events via the parent state, *Environment*. This way of treating the agents in the world would make communication more believable as it would go beyond the traditional solution of reading each other’s mind and instead allow all current activated agents to capture the same events, which for instance could support the notion of eavesdropping. These agents may also be aborted as soon as they have served their purposes.

SCXML+VoiceXML is not just a framework for building spoken dialogue systems, but also for controlling telephony – a framework in which technologies for voice recognition, voice-based web pages, touch-tone control, capture of phone call audio, outbound calling (i.e. initiate a call to another phone) all come together, creating a marvellous playground for building new and innovative games.

4.5 Implementing Dialogue Management Strategies in SCXML

SCXML is thus intended to constitute the dialogue management component in a spoken dialogue system within the DFP framework. It communicates with the presentation layer, for example a VoiceXML component, through asynchronous events. In more concrete terms, when VoiceXML has successfully parsed the user's input, a "filled" event is raised that can be captured by the SCXML component. The event data may contain semantic information specified in the grammar specification. In the following sections it is assumed that the user interacts with a VoiceXML application and that the grammar specification (see section 4.2.1) outputs a semantic representation in the form of a communicative act, perhaps with some additional information such as the propositional content and/or attitude. In the example below, the user from our previous restaurant example can either greet the waiter by saying "hello" or "good afternoon", or order a beer (or coke) using the exact phrase "I would like a beer please". The grammar passes on the type of communicative act, corresponding to the value of the tag "out.ca", to the dialogue manager:

```
<grammar version="1.0" root="start">
  <rule id="start">
    <one-of>
      <item><ruleref uri="#greet"/>
        <tag>out.ca='greet' </tag></item>
      <item><ruleref uri="#request"/>
        <tag>out.ca='request';out.request=rules.request;</tag>
      </item>
    </one-of>
  </rule>
  <rule id="greet">
    <one-of>
      <item>hello</item>
      <item>good afternoon</item>
    </one-of>
  </rule>

  <rule id="request">
    <one-of>
      <item>I would like a coke please
        <tag> out='coke';</tag></item>
      <item>I would like a beer please
        <tag> out='beer';</tag></item>
    </one-of>
  </rule>
</grammar>
```

The VoiceXML layer is invoked on start up and runs in parallel with the dialogue manager controlled by the SCXML application, as in Figure 19, below.

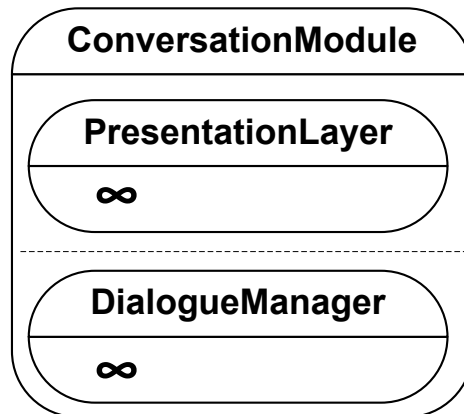


Figure 19: Conversation module

This way the presentation layer need not be invoked every time the system expects a user input, as in the following code skeleton.

```

<parallel id="ConversationModule">
  <state id="PresentationLayer">
    <invoke id="conversation" type="x-vxml-
      interpreter">
      ...
    </invoke>
    ...
  </state>
  <state id="DialogueManager" initial="S"/>
    <state id="S">
      <transition event="filled"
      ...
    </transition>
    ...
  </state>
  ...
</state>
</parallel>

```

Some of the examples listed in the appendix use this structure, but follow the SCXML syntax at the time of writing. Using VoiceXML as the presentation layer and SCXML for controlling the flow is in line with the intention of the DFP framework.

It could also be useful to design a “meta-dialogue manager” in SCXML that is responsible for selecting which dialogue manager to invoke (or close down) in a given situation. Since an SCXML process can invoke processes implemented in languages other than SCXML, it would be possible for example to invoke an AIML-process for social chats and then a VoiceXML process for handling a practical dialogue. Thus, such a meta-dialogue manager would provide the developer with much freedom in implementing a dialogue manager in a language

that is the most suitable for the task as well as allowing several dialogue managers to run in parallel.

Next we will discuss how SCXML can implement the strategies for dialogue management discussed in chapter 2.

4.5.1 FSMs in SCXML

It should be clear by now that statecharts really are extended finite state machines, and consequently they have the ability to model ordinary finite state machines. Translating the finite state-based approach to dialogue management into SCXML code is therefore rather straightforward.

A dialogue such as the one presented in Dialogue 3 could for example be managed in the following way:³⁶

```
<state id="travel_from_to_planet" initial="from">
  <datamodel>
    <data id="fromPlanet"/>
    <data id="toPlanet"/>
  </datamodel>
  <state id="from">
    <onentry>
      <send event="from_dest" target="tts"/>
    </onentry>
    <transition event="filled"
      cond="_event.data.from_planet!= undefined"
      target="to">
      <assign location="fromPlanet"
        expr="_event.data.from_planet"/>
    </transition>
  </state>

  <state id="to">
    <onentry>
      <send event="to_dest" target="tts"/>
    </onentry>
    <transition event="filled"
      cond="_event.data.to_planet != ''"
      target="wrapup">
      <assign location="toPlanet"
        expr="_event.data.to_planet"/>
    </transition>
  </state>
</state>
```

³⁶ It should be noted that the example is simplified to the extent that it does not have any error handling and that it accepts any input from the user.


```
<state id="wrapup">
  <onentry>
    <!-- confirm and wrap up dialogue -->
  </onentry>
</state>
</state>
```

4.5.2 Frame-based Approach

Frame-based dialogue systems are better at creating flexibility in the dialogue in comparison to finite state-based systems, but just like finite state-based dialogue systems they are suitable for simpler tasks, in which the system collects information from the user in order to fill empty slots necessary for accomplishing the task. It ignores the user's behaviour as long as the relevant information is provided, and the agent is expected to behave consistently and predictably. Moreover, frame-based systems have limited dialogue management capabilities in that they cannot handle topic switches unless they can be regarded as a sub-dialogue. That is, once left, the dialogue must be restarted rather than resumed if a topic change occurs. Statecharts, on the other hand, allow the application to return to a previous dialogue state by means of the history state. This means that its different task agents may be combined to solve more complex tasks, which Allen et al. (2001) refer to as "sets of contexts", where shifts between predetermined topics are made possible. The state of the current dialogue may be saved in case of a topic shift and resumed again.

In the finite state-based example above, the user's input was stored in a datamodel, basically functioning as the fields in VoiceXML. Now, this datamodel can also be used to fill several slots in one user turn. The information must first be parsed through the grammar, which returns a semantic interpretation in the form of the 'tag'-construct. Assuming the grammar returns values for "from_planet" and "to_planet", as in the previous example, the SCXML dialogue manager using a frame-based approach could be implemented as in the code example below.

```

<state id="travel" initial="bypass_init">
  <datamodel>
    <data id="fromPlanet"/>
    <data id="toPlanet"/>
    <data id="count" expr="0"/>
  </datamodel>
  <state id="bypass_init">
    <transition cond="count==0"
      target="get_from_and_to_planets">
      <assign location="count" expr="count+1"/>
    </transition>
    <transition cond="fromPlanet != undefined && toPlanet!=undefined" target="wrapup">
    </transition>
    <transition cond="fromPlanet == undefined"
      target="from_planet"/>
    <transition cond="toPlanet == undefined"
      target="to_planet"/>
  </state>
  <state id="get_from_and_to_planets">
    <onentry>
      <send event="from_to" target="tts"/>
    </onentry>
    <transition event="filled" target="bypass_init">
      <assign location="fromPlanet"
        expr="_event.data.travel_from"/>
      <assign location="toPlanet"
        expr="_event.data.travel_to"/>
    </transition>
  </state>
  <state id="from_planet">
    <onentry>
      <send event="from" target="tts"/>
    </onentry>
    <transition event="filled" target="bypass_init">
      <assign location="fromPlanet"
        expr="_event.data.travel_from"/>
    </transition>
  </state>
  <state id="to_planet">
    <onentry>
      <send event="to" target="tts"/>
    </onentry>
    <transition event="filled"
      target="bypass_init">
      <assign location="toPlanet"
        expr="_event.data.travel_to"/>
    </transition>
  </state>
  <final id="wrapup"/>
</state>

```

The state `bypass_init` is responsible for keeping track of the dialogue state, i.e. which slots have been filled in the form, and follows the FIA algorithm in

determining the next dialogue move.³⁷ This means that in the first iteration it will transition to `get_from_and_to_planets`, which is a state that will attempt to fill every slot that matches the user’s input before returning to `bypass_init`. If there are remaining slots to fill, `bypass_init` will check visit each state that request the missing information in a finite state-based manner, in this case either `from_planet`, `to_planet`, or both, in consecutive order. When the slots have been successfully filled, the dialogue can be wrapped up and terminated.

This is of course a very simple example and should be read as such, but it suffices to show that the form-based approach can be implemented in SCXML.

4.5.3 Plan-based Approach

A plan is really a sequence of behaviours leading to a specific goal. In a plan-based approach, such a plan can be generated in real-time by means of logical reasoning based on the agent’s goal (desire) and its perception of the world (belief) (recall the BDI-model presented earlier, the plan corresponds to the agent’s intention). Plans may be constructed as a sequence of states but a state machine cannot generate a plan. The `Serve` state introduced in chapter 1 (see Figure 2) constitutes such a plan and consists of a sequence of behaviours that the waiter must traverse in order to execute the `serve`-behaviour. It is also possible to observe the waiter’s current behaviour in the plan by the “In”-predicate. SCXML thus enables the implementation of plans, but is currently incapable of planning, that is, of generating a plan in real-time. It would however be possible for an SCXML application to interact with a planner, for example a Prolog program as suggested by Radomski and colleagues (Radomski, Schnelle-Walka, & Radeck-Arneth, 2013)

4.5.4 Information State Update Approach

In chapter 2.6.4 we presented the information state update approach as an abstract framework that needs to be filled with theoretical content. It was also mentioned that an information state contains all information available to an agent during the dialogue, of which some is private and some is shared. The question is whether SCXML is capable of implementing the ISU approach and according to Kronlid and Lager (2007) this is possible and they suggest the following correspondences:

Information state – datamodel. The statechart can manipulate external data through the use of datamodels, in which parameters for specifying the circumstance and conditions during which the dialogue takes place can be set.

³⁷ Note that “`bypass_init`” previously was the name of the `<initial>` element, which keeps track of the fields that could be filled from the user’s initial input. The element `bypass_init` is used in a similar way in this example.

The datamodel may therefore correspond to the contextual information stored in the information state in the IS approach.

Update rule - transition. Transitions can be regarded as “update rules” (see also Gandhe et al., 2008), i.e. “a set of applicability conditions and a set of effects” (Larsson & Traum, 2000, p. 5) that change the state of the dialogue.

Dialogue move – event. Just as an event in SCXML may trigger a transition from one state to another and possibly assigning new values to the data objects stored in the datamodel, a dialogue move in ISU may trigger an update of the information state.

Gandhe and colleagues (Gandhe, DeVault, Roque, Martinovski, Artstein, Leuski, Gerten, & Traum, 2008) have implemented a dialogue manager using the ISU approach in SCXML. The state diagrams model the virtual character’s conversational obligations, with the different possible sub-dialogues running in parallel. Just as Kronlid and Lager suggest, the dialogue moves from either participants are treated as events that can trigger a transition to another state (for example from `question_resolved` to `question_not_resolved`, when the player’s latest dialogue move was interpreted as a question). The current data model configuration represents the information state and can also be used for conditional transitions.

4.6 Advantages of Using Harel Statecharts

Harel statecharts have previously been promoted as a suitable tool for describing dialogue flow (Kölzer, 1999) and the fact that statecharts have been endorsed by the World Wide Web Consortium (W3C) to constitute the semantics of the SCXML makes them particularly interesting. When control is managed by SCXML it allows an application developer to invoke several simultaneous presentation components, for example a voice-based component together with a multi-modal component, thus making it possible to synchronize speech with gestures for example.

4.6.1 Visual Representation

System developers are already familiar with the concept behind statecharts through Unified Modelling Language (UML) diagrams, which constitute an established methodology for system design (see for example Pressman, 2010). Horrocks (1999) argues that statecharts can also be used with advantage in the design of user interfaces. It is for example both simpler as well as faster to describe the flow using statecharts in comparison to describing it using text. He

also stresses that a visual representation is easier to understand and follow and can thereby simplify communication. Another advantage he lists is the fact that statechart design does not require a specific programming language; instead statecharts can be created using a simple drawing tool.

A visual representation makes the system easy to get an overview of, manage and communicate to others. There is however no standard way for game designers to illustrate game flow, but attempts have been made to create a visual language for describing game flow, for example “Computer game-flow design” (Taylor, Gresty, & Baskett, 2006). Another example is BrainFrame, a visual authoring tool for creating game AI created by Fu & Houlette (2002) that is based on FSMs. Other visualization tools that are used are flowcharts (promoted by for example Fullerton, 2008), storyboards (see for example Cristiano, 2008) and use case diagrams (see also Hammersley, 2009). These tools are sometimes used in combination to serve different demands.

Graphical representations let designers create a blueprint of the game that may support communication between the members of the development team. Depending on the game designer’s background and skills, tools may either be too technical (e.g. UML class diagrams) or too oriented towards the visual and/or narrative aspect of the game (e.g. storyboards). For a game designer it is important to be able to describe the game at different levels of detail. A game designer need not, for example, determine how the game is actually coded, but they need to be able to communicate how the game is supposed to work so that it can be translated accurately into code.

FSMs are extensively used for implementing the game AI since they are “conceptually simple, efficient, easily extensible, and yet powerful enough to handle a wide variety of situations” (Fu & Houlette, 2004, p. 283). Another advantage is that they have a visual representation that directly corresponds to the actual implementation. Behaviours are expressed in terms of states and changes in behaviours are expressed by transitions between states. We have already mentioned that FSMs only can be in one state at a time, which is problematic when actions are to be combined to form more complex behaviours. An NPC that has an action “aim” as well as an action “shoot”, for example, cannot aim and shoot at the same time unless there is a separate state for “aim-and-shoot”. This is because the NPC cannot be in the atomic states “aim” and “shoot” simultaneously. Statecharts have the advantages of FSMs but due to the hierarchical and concurrent features they avoid the most pressing problem, the state explosion problem. Statecharts are also easier to get an overview of and manage due to their “zoom-in” and “zoom-out” features.

4.6.2 Statecharts and the Iterative Design Process

In software engineering as well as within game development the iterative design approach is often promoted (see e.g. Pressman, 2010; Salen & Zimmerman, 2004). Typically a project is divided into a number of independent deliverables that each iterates through 4 main processes in each cycle: design, prototype, test and revise. Projects that use the iterative approach typically apply agile process models, such as SCRUM and eXtreme Programming (XP), where small groups are responsible for a specific part of the project and have regular “builds” when the prototype is tested and evaluated. This process model has become more and more common within game development since it gives the team the opportunity to test and correct potential design flaws early on in the project. The quality is assured and since regular updates are provided it is easier to have control over the project’s development.

Statecharts are well suited for the iterative design approach since the design can be broken down into small manageable pieces, each representing separate modules that can be worked on in isolation by different members of the team. And since statecharts offer an abstract representation of the software design, it can be iterated, tested and revised several times before the actual code is written (Horrocks, 1999).

4.6.3 Summary

Several advantages in using statecharts have been mentioned in this section and they can be summarized as follows:

- They are graphical representations that simplify communication between team members
- They support a modular-based approach to system development, meaning that design and implementation can more easily be distributed among team members.
- Statecharts can describe a system at different levels of granularity, thus supporting communication at different levels of abstraction.
- Statecharts support the iterative refinement approach to game design (e.g. Kreimeier, 2003; Salen & Zimmerman, 2004) typically used in for example agile process models, such as SCRUM, as well as bottom-up clustering – allowing the designers to specify and discuss concepts in isolation. (We will discuss this further in the section below)
- They are finite-state based (and most games are finite-state based)

4.7 Examples of Interactive Systems Using a Similar Approach

In order to create agents capable of expressing individual behaviour, Egges and colleagues conducted an experiment “to realistically connect personality and 3D characters not only on an expressive level [...], but also on a dialogue level (generating responses that actually correspond to what a certain personality in a certain emotional state would say)” (Egges, Zhang, Kshirsagar, & Magnenat-Thalmann, 2003, p. 1). They regard a dialogue system as a collection of dialogue units, each handling a specific dialogue between the user and the computer. They have implemented such dialogue units as small finite state machines (FSMs) that run concurrently under an FSM kernel. The different dialogue units may have transitions that correspond to a certain input, but instead of allowing several transitions to take place simultaneously, an algorithm selects which transition is to be chosen based on priority. The dialogue system also has an interface to a mental model that formulates the mental state as a condition specified in XML. The data from the emotional state then tags the output text of a response. Even though their solution may support individual behaviours in terms of a variety of combinations, it still lacks the possibility of allowing the agent to perform several forms of expressive behaviour simultaneously and independently.

Gebhard et al. (Gebhard, Kipp, Klesen, & Rist, 2003) have developed a toolkit called SceneMaker for creating interactive stories that uses cascaded FSMs (statecharts in effect) to model the scene flow. Every story contains a logical scene flow that defines the transitions between scenes, which is provided with content that is either pre-scripted or automatically generated. They call the active state a node, which can be of one of two types depending on whether it is a super-state or an atomic state:

- Scene node - represents a state in which a pre-scripted scene is performed
- Super node - represents a state in which a pre-scripted or automatically generated scene is performed. Super-nodes may contain sub-nodes of either node-type, and one of the sub-nodes must be declared the starting node. The sub-nodes inherit all the edges that are connected to the super-node.

An edge defines the transition between states and like nodes they may have pre-scripted scenes attached, which will be invoked when the edge is traversed.

The first system that has been implemented using SceneMaker is the CrossTalk System (Gebhard, Kipp, Klesen, & Rist, 2003), in which the user can observe and provide feedback in a conversation between two agents, a salesperson and a customer, displayed on separate screens. By using simple means, such as

applauding or booing, the user can influence agents' behaviour and the progression of the story.

Zubek (2005) uses a stochastic-based approach (see section 2.6) for modelling game dialogues and extends ordinary Hidden Markov Models (HMMs) to support hierarchy and parallelism. He applies the method in two different "games": *Xenos the innkeeper*, a shopkeeper NPC that sells items and delivers quests to the player, and *Break-up conversation*, simulating a break-up conversation over chat between the player and the system in the role of the "soon-to-be-ex".

The dialogue engine iterates over the following consecutive steps: (1) parsing and pre-processing of the user's input; (2) an evidence estimation based on the output from (1); (3) state estimation "computes the new belief distributions for each parallel space, given the evidence and the last known state"; and (4) action production and generation, translated into utterances or emotes (Zubek, 2005, p. 109).

The result is an interaction system that exhibits the desirable hierarchical parallel property: the interaction retains hierarchical composition of sub-interactions that can affect over each other, but at the same time allows for parallel and independent engagement on different levels of generality. The redundant representation softens the performance drop-off at the edge of competence, and can be used to mimic some of the human ways of dealing with error. (Zubek & Horswill, 2005, p. 145)

Since SCXML is Turing complete, it would be possible to implement Zubek's approach using SCXML.

Chapter 5

Face Management for NPCs

This chapter revisits the discussion concerning cooperative agents introduced in chapter 2, and attempts to model an NPC with the ability to take another game character into “ethical consideration” by means of “face management” (see Goffman, 1967). Face management can be described as the strategies we use in order to maintain our own and the other participants’ public self-image. In this chapter we will investigate ways to equip an NPC with face management skills in order to make it appear more socially competent. Also, by knowing the appropriate way to act in a given situation, the NPC can choose whether to be socially compliant or not. This chapter thus aims to present a model for implementing the hypothetical game design pattern SOCIAL NORM (see Table 7 in section 3.7), which will be applied to the waiter character presented earlier in 1.3.1.³⁸

We take Brown and Levinson’s (1987) theory of politeness as a starting point, and in particular their algorithm for determining the threat to face given a specific communicative act in a particular situation. First, the theoretical background is given and thereafter the waiter character presented earlier will be revisited and updated.

5.1 Face Management

Being able to take an order and serve a customer using natural language would offer a new gameplay challenge in games such as the waiter character presented in chapter 1. However, in the example the waiter will behave in the exact same way each time a customer enters the restaurant, which is at odds with Hayes-Roth and Doyle’s (1998) suggestion that a character needs to behave variably and unpredictably in order to be believable. This section presents an attempt to

³⁸ Parts of this chapter have previously been published in (Brusk, 2008) and (Brusk, 2010).

improve the waiter's behaviour by taking ethical considerations into account, here in the form of politeness expressed by means of face management actions.

5.1.1 Threats to Face

“Face” is often used in metaphorical expressions related to our self-image, such as “lose face” which is used as a metaphor for being humiliated. It can be described as the mask we use in social situations, the appearance we want to keep in the interaction with others. Goffman (1967), for example, defines face as “the positive social value a person effectively claims for himself by the line others assume he has taken during a particular contact [...], an image of self”. This implies that we are expected to use different “masks” for different occasions, for instance wear a “professional mask” when interacting with people in the work environment, and a “parent mask” when interacting with other parents or our children's teachers for example. As professionals we probably want to be perceived of as proficient, knowledgeable, and the like, while we as parents may want to be perceived of as reliable and affectionate. In the former situation it might not be important whether we are affectionate parents, and in the latter proficiency in our job may be of less importance. Hence, it is the context that determines whether a certain behaviour is face threatening.

We can assume that all participants in a social situation wish to preserve their face, but this is only possible if the other participants cooperate in this attempt since every action towards another participant is a potential threat to someone's face. Hence, face preservation is a cooperative effort (see Brown & Levinson, 1987, p. 61) and an example of the ethical considerations the participants trust each other to have – at least in order to achieve *Ideal Cooperation* (see chapter 2.3.3). Consequently, participants try to avoid actions and situations that may constitute a threat to face, something Goffman (1967) refers to as “the avoidance process”. Participants may for example use hedges to diminish the effect of the face threatening action (FTA), i.e. a verbal or non-verbal communicative act that intrinsically threaten face (see Brown & Levinson, 1987, p. 65). Sometimes, though, the FTA is unavoidable and in order to preserve the face of the participants, attempts are made to correct for its effect, something Goffman calls “the corrective process”. Here, the strategies we use in order to maintain our own and others' face will be referred to as face management actions.

5.1.2 Brown and Levinson: Politeness Theory

Brown and Levinson (1987) attempt to provide a more detailed understanding of the mechanisms that control the choice of strategy an agent is confronted with when a threat to face is unavoidable. First, when an agent is about to make an FTA, they assume that the agent considers the relative weightings of the

following three desires: (1) the desire to communicate the FTA, (2) the desire to be efficient and urgent, and (3) the desire to maintain the hearer's (H) face to any degree. Second, the agent is assumed to evaluate the interpersonal relationship and estimate the value of the FTA in the current context. To do this, Brown and Levinson suggest that the agent makes an estimation of the values of three socially determined variables: the social distance between the speaker (S) and the hearer (H), $SD(S, H)$; the power the hearer has over the speaker, $P(H, S)$: "the degree to which H can impose his own plans and his own self-evaluation (face) at the expense of S's plans and self-evaluation"; and, finally, a "culturally and situationally defined ranking of imposition" for the act under discussion, I , i.e., the extent to which the act interferes with the agent's self-determination or approval. The calculation is then made by a simple addition of the individual factors:³⁹

$$(1) \quad SD(S, H) + P(H, S) + I$$

Since these values reflect the individual agent's *own* perception of the interpersonal relationship, they can only be approximations. Brown and Levinson however loosely propose that the value of each factor is an integer between 1 and n , where n is some "low number". They furthermore consider the possibility that the algorithm is enhanced with other (e.g. cultural) factors that may be of importance for determining the face threat in a specific community.

The evaluation will result in one of the following strategies (Brown & Levinson, 1987, p. 68-70): 1) *to do the FTA*, either (a) "*on record*", i.e. to unambiguously express the intention, or (b) "*off record*", by using for example irony or metaphors thus by hinting the intention rather than expressing it directly. 2) *Not to do the FTA*. Several different strategies can furthermore be used when going "on record" (1a), for example to do it in the most direct and precise way ("baldly, without redress"), or by using negative or positive politeness, where negative politeness is essentially avoidance-based (S might for example use hedges and other softening mechanisms to avoid putting pressure on H) while positive politeness is oriented to the positive self-image H claims for himself.

In the following section we will present a couple of implementations using Brown and Levinson's framework and in chapter 7 we will return to these face-threatening acts and discuss how they potentially may impact how gossip is produced and perceived.

³⁹ Brown and Levinson actually use R_x for denoting the ranking of imposition, but we use I instead, following Walker et al.'s notation, which is presented next.

5.1.3 Walker, Cahn and Whittaker: Linguistic Style Improvisation

Walker and colleagues (1996, 1997) use Brown and Levinson's algorithm combined with speech act theory to produce *Linguistic Style Improvisation* (LSI). LSI is described as the choices a speaker makes about the semantic content, syntactic format and the acoustic realization of their spoken utterances. A particular speech act is given a specific ranking of imposition based on how they relate to the agents' desire for autonomy and approval. Speech acts that can function as a threat to H's desire for autonomy include those that predicate some future act of H, as well as speech acts that predicate some future act of S toward H, such as offers, which put pressure on H to accept or reject them. Speech acts that threaten H's desire for approval include all rejections. The ranking of imposition is thus based on the speech act type where each speech act has been assigned a value between 1 and 50. The threat value, which they denote with Θ , is then calculated in accordance with Brown and Levinson's algorithm:

$$(2) \quad \Theta = SD(S, H) + P(H, S) + I_{(speech\ act)}$$

The agent evaluates the social distance (which roughly corresponds to how well S and H know each other) and assigns it a value between 1 and 50, and the power (P) is evaluated similarly (where power is described in a similar fashion as proposed by Brown and Levinson). An addition of these values thus generates a maximum value of 150. The sum is used to choose one of the following four strategies for executing a speech act, selected in order from lowest threat value to highest:

- DIRECT: Do the act directly (corresponds to Brown & Levinson's "on record strategy – baldly");
- APPROVAL-ORIENTED: Orient the realization of the act to H's desire for approval (Brown & Levinson's "on record – positive politeness");
- AUTONOMY-ORIENTED: Orient the realization of the act to H's desire for autonomy (Brown & Levinson's "on record – negative politeness");
- OFF-RECORD: Do the act off-record by hinting, and/or by ensuring that the interpretation of the utterance is ambiguous (Brown & Levinson's "off record strategy").

Walker and colleagues have thus chosen an algorithm that for the most part follows Brown and Levinson's proposal. As noted in (Walker, Cahn, & Whittaker, 1996) one weakness in their model is that they only consider the imposition of the speech act itself and disregard the propositional content. Meaning that the following requests are treated equally: "can I borrow a dollar" and "can you lend me \$10 000?". The values they have assigned each speech act must also be assumed to be specific for the particular scenario they use (a

restaurant scene from Casablanca). A request for X may for example be more or less imposing depending on whether it is in line with H 's role to provide S with X or not. Acts that are associated with different roles thus ought to be treated as less threatening in general than the acts that are not.

5.1.4 Prendinger and Ishizuka: Social Filter

Prendinger & Ishizuka (2001a, 2001b) present a program that acts as a social filter between the agent's affective state and its rendering in a social situation. They calculate the agent's emotional state based on the agent's beliefs, goals, and preferences using an inference machine referred to as the "affective reasoner". The social filter program takes as input the agent's emotional state and outputs the conditions for the emotion expression based on the agent's personality and social role. We will return to their implementation in section 5.3.2 and forward.

Prendinger and Ishizuka's implementation differ from Walker and colleagues' application on two main accounts: First, they explicitly prioritize the mental state of the agent, i.e. its personality type and emotional state, and how it affects the agent's behaviour. Walker and colleagues have not taken into account the agent's mental state. Second, they simplify Brown & Levinson's algorithm by discarding the ranking of imposition and instead consider the agent's social role awareness, which they summarize as the agent's tendency to accept or violate social conventions, for instance those conventions associated with a particular role. The agents' social role awareness becomes especially prominent in socio-organizational settings where the roles have asymmetrical power built-in, such as doctor-patient, teacher-pupil.

Prendinger and Ishizuka also use a smaller span for determining the social distance and power, limited to a value ranging from 0 to 3 (which in itself is a modification of Brown and Levinson's suggestion, as they specified it to be a positive number), where 0 corresponds to a very close relationship.

5.2 Introducing ρ

We have also used Brown and Levinson's algorithm as a starting point for dealing with face preservation in social interaction. We have furthermore been inspired by Walker and colleagues' application of the algorithm and even reconstructed parts of Prendinger and Ishizuka's implementation in (Brusk, 2008) (see appendix I and II).

In essence, Brown and Levinson's algorithm covers and supports the possibility to take cultural and situational factors into account, but does not specify which they are. This means that the parameter I most likely will stand for different

things in different situations and therefore cannot be determined using a simple calculation similar to SD and P. Walker and colleagues, for example, chose to use it to evaluate the degree to which a particular speech act may interfere with the agents' approval or self-determination in one particular context and given one particular role (in this case a waiter). Prendinger and Ishizuka chose instead to exchange the parameter I with a factor for determining the agent's inclination to adhere to social conventions. We do not consider I to be of less importance than SD and P, but we find its role in the algorithm to be too vaguely defined to be useful for our purposes. Even though it is context-dependent just like the other two variables, it is much more difficult to define since it is a variable that possibly combine several different and independent values, e.g. the ranking of imposition of a particular speech act with the propositional content uttered in a particular situation with a given role distribution. It would thus be very difficult to calculate this value appropriately in an implementation. Instead, we propose an algorithm that simply measures the interpersonal relationship between the participants, which in turn determines the *preconditions* for a certain type of dialogue exchange, denoted ρ :

$$(3) \quad \rho = SD(S, H) + P(H, S)$$

It is thus Brown and Levinson's algorithm presented in (1) but with the ranking of imposition (I) omitted.

In the social filter rules, described later in this chapter, ρ is used in combination with the agent's personality type and emotional state to determine the linguistic style (neutral, rude or polite) of an utterance. In Chapter 7 ρ is used to determine whether it is appropriate to initiate a gossip conversation.

5.2.1 Calculating SD and P

It is difficult to accurately specify how well we know another person on an arbitrary scale – the question is whether it is easier to use a rough estimation, using a scale consisting of a fairly low number of degrees, say from 0 to 3, like Prendinger and Ishizuka, or using Walker and colleagues' scale ranging using a much larger span. To have some guideline for specifying the range of the value of SD we use the same scale as Prendinger and Ishizuka as given in the previous section, but we also specify the meaning of the values by the following approximate correspondences:

- 0: intimate, such as partners or parent-child
- 1: (close) friends
- 2: acquaintances
- 3: strangers or distant acquaintances

The power is also a value between 0 and 3, but is not as easily translated into specific relationships. The power H has over S is typically related to the degree to which H can control money or goods that S wants (Brown & Levinson, 1987; Walker et al., 1996, 1997), and corresponds roughly to how the participants are hierarchically organized. A high value of P indicates an asymmetric power relationship.

Both P and SD are context-relative but the meaning of ρ should be rather obvious; a low ρ value means that the risk of losing face is low, while a high value of ρ increase the risk and in reality delimits the scope of what the agents can talk about, assuming that they all are equally interested in preserving their own and the other participant(s)' face.

5.3 Applying ρ in a Practical Dialogue

We now aim to show how ρ can be used to specify a conversational agent's dialogue behaviour. In addition, we will allow the agent to let its personality traits and emotional state influence the behaviour. We will continue to use the game scenario presented in chapter 1 as an example, which means that the waiter's statechart will now be extended with a parallel statechart describing the waiter's emotional state. Since this thesis is not about how to model emotions in artificial agents we will only use a very simple model for simulating emotions. The personality model is also for demonstrational purposes. The main point is to show that different factors can determine and affect the agent's behaviour and that we do not need a lot of information to make the agent appear to be more complex than it really is.

As has already been mentioned, the algorithm for specifying ρ does not take the actual speech act into account. In the example given here this is not necessarily a problem since the roles are associated with expected communicative acts, which, even though they might be regarded as having a high imposition (for example requests), are instances of the behaviour each role is expected to have (in this case waiter-customer).

The emotional state of the waiter character and in particular its relationship to the customer may be determined by the situation in which the dialogue takes place. It should be noted that these factors are both CHARACTER SPECIFIC as well as CONTEXT DEPENDENT (see 3.7) so the model that will be presented later is a reflection of a specific dialogue situation, but a generalization is made about the use of these factors *per se*.

The dialogue between a waiter and a customer is a typical example of a practical dialogue and practical dialogues have been successfully implemented in

commercial systems (see e.g. Allen et al., 2001). Some discussion of this have been presented earlier, for example:

- They operate in a well-defined and limited domain – which decreases the number of potential interpretations of the input.
- They have a fairly predictable structure.

These limitations have consequences that also may contribute to the success:

- The interaction has a natural termination, which is when the task has been completed.
- The duration of the dialogue is expected to be in parity with the required steps to complete the task, so efficiency is often counted as a measure of success.

These two points affect the overall length of the dialogue, meaning that a success criterion is to keep the dialogue short and simple.

5.3.1 Activity Analysis

Following Allwood's (e.g. Allwood, 1995; Allwood, Traum, & Jokinen, 2000) schema for an activity-based communication analysis, the waiter-customer situation can be described as follows:

Purpose: The activity is a simulation of a restaurant visit in which the goal is to get something to eat or drink.

Roles: the NPC has the waiter role and is expected to take an order and serve the customers that enter the restaurant. The user plays the role of a customer who visits the restaurant. The NPC is expected to have knowledge of the domain, i.e., to know what is and what is not an appropriate order, even though the restaurant has a limited supply. If the restaurant does not supply alcoholic drinks, for example, the waiter must nevertheless be able to understand such a request and inform the guest that the order is rejected.

Physical and social environment: As has previously been indicated, we will not consider the user's physical environment. Instead the perspective is the player's character, the virtual customer. Therefore, the restaurant is a virtual simulation of a physical and social environment.

Artefacts: The artefacts are likewise virtual representations of real objects, such as tables, chairs, a counter and other typical artefacts existing in a restaurant environment.

5.3.2 Mental Model

The waiter’s mental model, here referring to a model of the waiter’s personality traits and emotions, has been designed with Prendinger and Ishizuka’s (2001a, 2001b) social filter program in mind (see 5.1.4). As was mentioned earlier, the program acts as a social filter between the agent’s affective state and its rendering in a social situation.

In their model, they consider the following two personality dimensions from the five-factor model (e.g. McCrae & Costa, 1999) to be relevant for explaining social behaviour: *Extroversion* (outgoing, neutral and introverted), which affects the choice of communicative act, and *agreeableness* (friendly, indifferent and unfriendly), which determines the linguistic style of the utterance (c.f. NPC roles in 2.2.2).

For a “cognitively intact” adult, personality is something that is preserved over time, or at least changes very slowly (McCrae & Costa, 1999). Therefore Prendinger and Ishizuka have treated the values of the personality dimensions as facts in their affective reasoner.

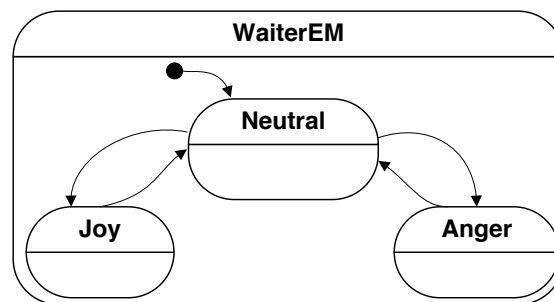


Figure 20: Emotions manager

We will not use an inference machine such as their affective reasoner; instead, the personality traits are integrated in the statechart itself. This means that characters with different personality traits are designed with a unique set of behaviours (in statecharts specified through states and transitions). An outgoing character may for example be inclined to talk more than an introvert character (see for example 2.4.2), which may be controlled by means of specifying conditions on the transitions for example. Another factor that will affect the waiter’s behaviour is his current emotional state. For managing the waiter’s different emotions we have created an emotions manager modelled as an independent statechart called `WaiterEM` (see Figure 20, above). Since this is just for demonstrating how the waiter can vary his verbal behaviour depending on his current emotional state we have chosen a simple model consisting of two basic emotions, Joy and Anger, and one state in which the agent is indifferent (here

referred to as Neutral).⁴⁰ The `WaiterEM` runs in parallel with `WaiterAM` and the `WaiterDM`.

5.3.3 The Effects of the Social Filter

The choice of linguistic style of the utterance is here regarded as an expression of politeness (or lack thereof), but the agent's emotional state and personality type can affect how inclined the agent is to accept the rules of politeness, i.e. its willingness and/or ability to adjust to a certain `SOCIAL NORM`. In our model we have used three different styles in which a communicative act can be expressed: Rude, neutral, and polite. We will assume that some alignment occurs, i.e. if the customer is rude, the waiter is more inclined to be rude as well.

The emotional state (see Figure 20, above) also affects the waiter's behaviour. An angry waiter, for example, may be less patient towards a rude customer or a customer that cannot make up his mind and thereby be more inclined to respond rudely (or perhaps even aggressively). A joyful waiter, on the other hand, may be more polite and patient in general and avoid adapting to a customer that behaves badly.

The following dialogue sample is extracted from running our simple text-based prototype that was based on Prendinger and Ishizuka's application (W=waiter, C=customer):

W: Welcome to our coffee shop!
C: Hello
W: May I take your order?
C: I would like a regular coffee, please
W: Very well, that will be 2 dollars please

Dialogue 6: Dialogue retrieved from running our social filter prototype with a friendly waiter in a coffeeshop

The example illustrates a waiter in the emotional state `Neutral` with a friendly and extrovert personality type. We used `CANNED TEXT RESPONSES` (see 3.7) that are stored as data in the datamodel, tagged only with linguistic style (`rude`, `polite` and `neutral` tags) and type of communicative act (value of `<data id>`), as in the following data structure for specifying the initial take order phrases:

⁴⁰ An overview of the current state of computational models of emotion can be found in (Marsella, Gratch, & Petta, 2010)

```

<data id="take_order_phrases">
  <phrases>
    <rude>What do you want?</rude>
    <polite>May I take your order?</polite>
    <neutral>What are you having?</neutral>
  </phrases>
</data>

```

In the current implementation we used a simulated user but assuming a human user interacting through a presentation layer implemented in VoiceXML, the user's linguistic style would have to be specified in the grammar, as in the code sample below for example:

```

<grammar version="1.0" root="start">
  <rule id="start">
    <ruleref uri="#requestOrder"/>
    <tag> out.order=rules.order;
      out.ca='order'</tag>
  </rule>
  <rule id="requestOrder">
    <ruleref uri="#requestMarker"/>
    ...
  </rule>
  <rule id="requestMarker">
    <one-of>
      <item> I would like
        <tag> out.ls='polite' </tag></item>
      <item> I want
        <tag> out.ls='neutral' </tag> </item>
      <item> Give me
        <tag> out.ls='rude' </tag> </item>
    </one-of>
  </rule>
</grammar>

```

Dialogue 7 and Dialogue 8, below, illustrate how the linguistic style can be affected depending on whether the agent is on the friendly or the unfriendly scale of the personality dimension *agreeableness*. If the customer greets the friendly waiter in a rude way and their interpersonal relationship does not in itself constitute a potential threat to face (ρ is low), the waiter will choose to answer in a neutral tone. However, if the waiter is in the unfriendly dimension of *agreeableness*, it will adapt the customer's behaviour and use a rude expression as well. Similar to Prendinger and Ishizuka's implementation, agreeableness is expressed in the linguistic style (LS). The agent's personality will thus define the behaviour of the statechart, i.e. different personality types will have a different set of possible behaviours.

W: Welcome!
W: May I take your order?
C: Give me a coffee right away!
W: A coffee, coming right up!

Dialogue 7: Friendly waiter

W: Hello.
W: What are you having?
C: Give me a coffee right away!
W: You will just have to wait for your turn, which possibly is never.

Dialogue 8: Unfriendly waiter

The personality traits thus affect the waiter’s behaviour in two ways: by choice of communicative act, i.e. what they say, and by choice of linguistic style, i.e. how they say it.

| S_1 | Event | LS | ρ | Agreeableness | S_2 | LS |
|-------|-------|------|--------|---------------|-------|---------|
| S1 | ca.* | Rude | 2 | friendly | S2 | Neutral |
| S1 | ca.* | Rude | 2 | unfriendly | S2 | Rude |

Table 8: Effects of agreeableness personality dimension

Possible transitions are specified in Table 8 and can be read as follows: The waiter is expecting an incoming “Event” in terms of a communicative act (denoted as “ca.*”, where * represents different types of communicative acts, e.g. “order”), encoded with the linguistic style “LS” in which it was expressed, and transferred in the event data. Depending on the interpersonal relationship (ρ) and the waiter’s “Agreeableness”, the waiter will perform a communicative act (in state “ S_2 ”) expressed in a specific LS.

An example of how a social filter rule can be coded in SCXML is given below and should be read as if the waiter is in the state `Anger` and ρ (the interpersonal relationship) is below 4, then go to state `TakeOrder`, otherwise go to the state `Greet`:

```
<transition cond="In('Anger') && rho < 4"
            target="TakeOrder"/>
<transition target="Greet"/>
```

This example is an illustration of the gameplay design pattern CHARACTER-SPECIFIC DIALOGUE – another waiter might always greet the customer whatever the current emotional state.

Similar to Prendinger and Ishizuka’s implementation, the personality dimension *Extroversion* affects the communicative act rather than the linguistic style. If ρ is

low, an introvert waiter may for example take the order directly instead of greeting the customer first (as shown in Table 9, below).

| S_1 | <i>Event</i> | ρ | <i>Extroversion</i> | S_2 |
|----------------|--------------|--------|---------------------|-----------|
| W_DialogueIdle | ca.greet | 2 | Extrovert | Greet |
| W_DialogueIdle | ca.greet | 2 | Introvert | TakeOrder |

Table 9: Effects of extroversion personality dimension

Dialogue 9 and Dialogue 10, below, exemplify how the introvert and extrovert personality dimensions, respectively, may be shown in the conversational behaviour:

W: What are you having?
C: A coffee

Dialogue 9: Introvert waiter approaches the customer

W: Welcome !
W: May I take your order?
C: A coffee

Dialogue 10: Extrovert waiter approaches the customer

If the character should be able to skip a stage, such as `Greet`, we need to update the previous model by adding a transition from `W_DialogueIdle` to `TakeOrder`. Assuming that the waiter only skips the greeting when not feeling threatened by the customer (i.e., ρ is low), the waiter’s dialogue manager could be modelled as the statechart in Figure 21, below. If the waiter never greets a customer, the state could of course be removed altogether. One could also design a character that only greets friends and acquaintances (low ρ) and never the customers that generate a high ρ -value. If the waiter’s emotional state determines his behaviour, such that he does not greet the customer when angry, the condition “In(‘Anger’)” must be added to the transition between `W_DialogueIdle` and `TakeOrder` (see previous code snippet). The choice of communicative act and the linguistic style used in the expression are thus dependent on the character’s personality, current emotional state and the interpersonal relationship – here represented by ρ . How to use these parameters is really up to the designer, what’s important here is that we can vary a character’s social behaviour by means of states, transitions between states, events, and conditions.

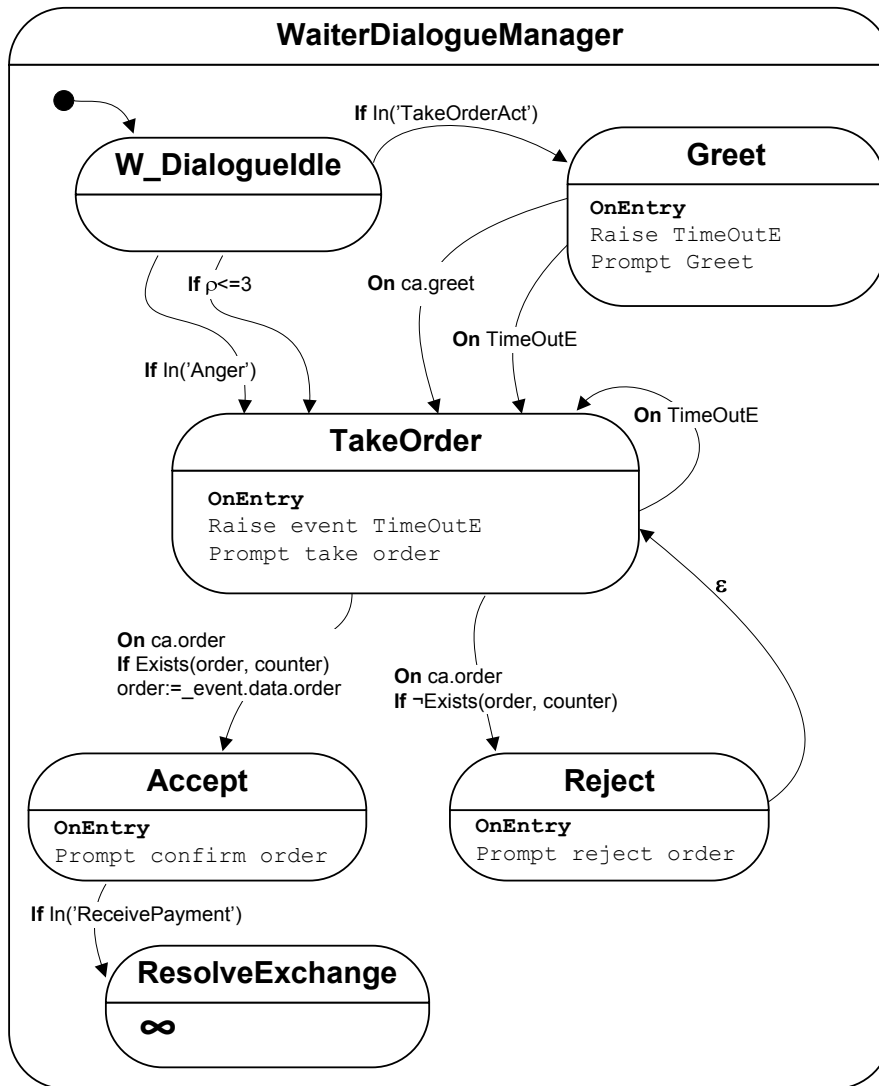


Figure 21: Dialogue manager for introvert waiter

5.3.4 Parallel Dialogues

Prendinger and Ishizuka (2001b) also implemented a boss character in order to demonstrate that the waiter’s behaviour is determined by the interpersonal relationship, in particular the equivalent they use to Brown and Levinson’s (1987) ranking of imposition, namely whether the character accepts or violates social conventions. The dialogue between the waiter and the boss shows how the threat value and the waiter’s attitude towards conventions come into effect in the waiter’s behaviour. In the example, the waiter tells the boss that he wants to take a day off. Depending on the result from the “affective reasoner” and the social filtering, the request is expressed politely as “Good afternoon boss. May I take a day off tomorrow?” or more casual as a statement in “Good to see you, boss. Tomorrow I will take the day off”. In both their examples, the boss rejects the waiter’s request, with a socially filtered expression. The waiter will obey the boss

or not depending on whether he respects or violates conventional practices toward him.

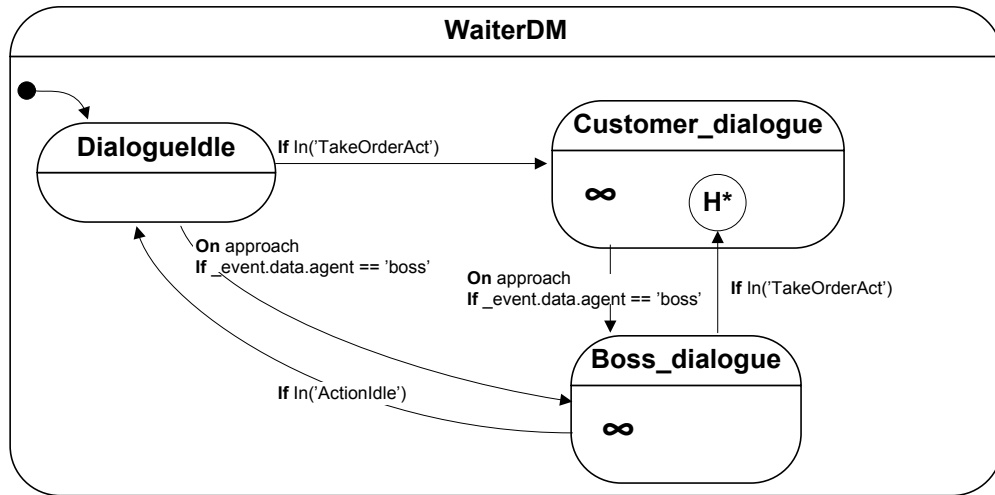


Figure 22: Parallel dialogue

As an extension to our version of the waiter character, the waiter can address the boss while being engaged in a dialogue with a customer. Rather than conducting the dialogue when the dialogue with the customer is terminated, as in their example, the waiter in our example addresses the boss when he passes by, signalled by the event ‘approach’ with a data payload specifying the agent to be the boss (see Figure 22, above). The dialogue is then managed in a module separate from the practical dialogue. Also, the waiter’s interaction with the customer is now specified in a new sub-state to `WaiterDM`, that we have chosen to call `Customer_dialogue`. The dialogue with the boss can then be initiated from either the `Customer_dialogue` or `DialogueIdle`, and this source state determines which state to return to when the dialogue with the boss is finished. If the active state is `Customer_dialogue`, the dialogue with the customer will be resumed as soon as the waiter is done talking to the boss. This is possible due to the deep history feature (H^*) in statecharts, which allows us to save the current dialogue state before it is exited. If the source state is `DialogueIdle`, there is no need to save the previous configuration.

Parallel dialogues are also possible to achieve by means of parallel states or by running several invoked dialogue managers simultaneously using the meta-dialogue manager mentioned earlier.

5.4 Summary and Conclusion

This chapter has presented a number of ways to create NPCs with the ability to manage face in social situations, i.e. NPCs applying the pattern `SOCIAL NORM`. The

first example showed how a character could be modelled to vary its behaviour according to the perceived interpersonal relationship. In order to do this, we introduced a parameter ρ , which is calculated by a modified version of Brown and Levinson's (1987) algorithm for calculating the weightiness of a face threatening action (FTA). The algorithm for calculating ρ was used in combination with an evaluation of the customer's behaviour to determine how the waiter should act. The model also took into account the waiter's current emotional state and personality type. In this regard, the models were simple and served merely to point out *how* personality and emotions can be represented using statecharts and how these differences may generate different behaviours.

Walker et al (1996, 1997) calculate the threat to face using Brown and Levinson's original algorithm, but where the factor I specifically corresponds to the imposition of a specific speech act. Prendinger and Ishizuka (2001a, 2001b), on the other hand, replaced I with "violating or respecting conventional practice". The algorithm presented here thus supports Brown and Levinson's algorithm as well as the variants proposed by Walker and colleagues and Prendinger and Ishizuka. The modification allows us to tailor the algorithm for different contexts, such as will be noticed in section 7.9, where we introduce sensitivity as an evaluation of the propositional content of a particular gossip story.

Chapter 6

DEAL

So far we hope to have presented convincing arguments for using a modular-based approach for designing natural language game dialogues, which is the main advantage in using statecharts. Now we also want to show that the approach is useful in other contexts and within other frameworks as well. This chapter presents our contribution to the DEAL project, a joint project for investigating the possibilities of creating a language learning system for conversational training that uses gameplay elements to create an immersive learning environment (see also Wik, Hjalmarsson & Brusk, 2007a, 2007b).⁴¹

6.1 Introduction to DEAL

DEAL is an example of a serious game, i.e. a game that has a purpose other than to solely entertain, such as teaching, training or advertising (Iuppa & Borst, 2007). The purpose of the project was to create a language learning system in which the user was encouraged to practice conversation by playing a simple trading game.

DEAL sets the scene of a flea market where a talking animated agent (a non-playable character) is the owner of a shop where used objects are sold. The objects sold at a flea market can be a diverse set of items, which can be tailored to suit the vocabulary mastered by a language learning student. A flea market is also a place where it often is acceptable to negotiate the price.

6.1.1 Framework

DEAL is implemented by using components from the *Higgins* project (see e.g. (Edlund et al., 2004): an off-the-shelf automated speech recognition (ASR) system, a dialogue manager developed for DEAL purposes, and a GUI with an embodied conversational agent (ECA).

⁴¹ This chapter has previously been published in (Brusk, Lager, Hjalmarsson, & Wik, 2007)

The *Higgins* project aims at developing “a collaborative dialogue system in which error handling can be tested empirically” (Edlund et al., 2004). *Higgins* is a module-based system consisting of an interpreter, *Pickering*, which takes the result from the ASR as input and creates a semantic representation of the user's communicative act as output. The communicative act is then sent to the discourse modeller, *Galatea* (Skantze, 2005). The output from *Galatea* constitutes the input to the dialogue manager implemented in SCXML.

The module presented in this chapter has been tested in the SCXML Web Laboratory⁴², a web-based interface for SCXML-applications. The dialogue manager (DM) has thus been developed separately from *Higgins*, i.e., there is no *real* connection between *Galatea* and the DM. Instead, we have made minor adjustments of the output from *Galatea* to fit the syntax of the event data as formalised in the SCXML Web Lab.

The communicative act retrieved from the output of *Galatea* is treated as an incoming event in our application. The SCXML DM responds with a new communicative act that in theory is passed to the generator in *Higgins* and then sent back to *Galatea* again.

6.2 Trade

Trade is a common game mechanics used in most role-playing games, such as *Morrowind* (Bethesda Game Studios, 2002) and other types of adventure games, for example *Animal Crossing* (Nintendo, 2001). A shopkeeper in a game is typically represented by an NPC, and a trade dialogue may be both practical (to successfully complete a transaction) as well as socially oriented, involving elements of negotiation and small talk. Trade is thus a good candidate for exemplifying how natural language dialogue can be used in a game setting.

There are thus several reasons why we chose trade for the DEAL project:

- It involves explicit and well-known roles of the participants.
- A trading situation is a fairly restricted and universally well-known domain. It is something everyone is conceptually familiar with, regardless of cultural and linguistic background.
- Trade may involve bargaining. The negotiation process is in itself an interesting research area due to its complexity containing both rational and emotional elements.

⁴² <<http://spyderbrain.ling.gu.se/index.html>>. The SCXML interpreter is written in Oz and developed by Spyderbrain technologies.

The aim is to create a believable interaction involving elements of freedom for the player combined with an unpredictable outcome associated with a negotiation process.

6.2.1 The Activity

Just like the activity of ordering something from a waiter at a coffeeshop, the DEAL scenario can be analysed using the parameters for “activity-based communication analysis” (see section 2.3), as follows.

Purpose: In serious games there is typically a higher purpose that goes beyond the intended player experience provided by the game itself. In this example the higher purpose is to be given the possibility to practice conversational skills in a target language. To fulfil this goal, the user is encouraged to talk as much as possible, e.g. request different items of various colours, forms and sizes, negotiate the price and so on. Seen from this perspective, the shopkeeper behaves like other task-oriented ECAs and the user interacts using his or her real identity. However, by creating a game-like activity around the task, it is assumed that the interaction becomes entertaining enough to motivate the user to *want* to use the system and use the system repeatedly.

The purpose for interacting with a shopkeeper is usually to buy something one needs, but it could also be to do a “good deal”, for example to buy something valuable for a cheap price. To make the activity into a game, various goals can be specified in advance, for instance “you have 30 SEK, buy a red clock and a green hammer from the shopkeeper”.

Roles: There are two well defined, and asymmetric, roles – a shopkeeper, who is represented by the NPC, and a customer, the role assigned to the user. Each role is associated with certain obligations; the buyer is for example expected to request an object while the shopkeeper’s obligation would be to respond to that request by presenting an object that matches the customer’s wishes.

In considering the interpersonal relationship, we can assume that the social distance is relatively high since this is a task-motivated interaction rather than personally driven. They both have some power over each other since they both have something that the other one wants – but the customer is more likely to have an advantage as the shopkeeper must keep business going in order to stay in business, while the customer most likely can do without the items.

Artefacts: The setting is a flea market in which used objects are sold. The customer (user) can see the objects for sale on the screen, located behind the shopkeeper. Between the customer and the shopkeeper is a counter on which the shopkeeper may place the requested item. When the shopkeeper and the buyer

have come to an agreement, payment is conducted by placing money on the counter.

Physical and social environment: The game uses a first person perspective in a single-screen 2D-display (see Figure 23). The imagined player character stands in a fixed position in front of a counter face to face to the shopkeeper. Behind the shopkeeper is a wall showing the objects for sale and the buyer's assets are also visible in the graphical user interface (GUI).

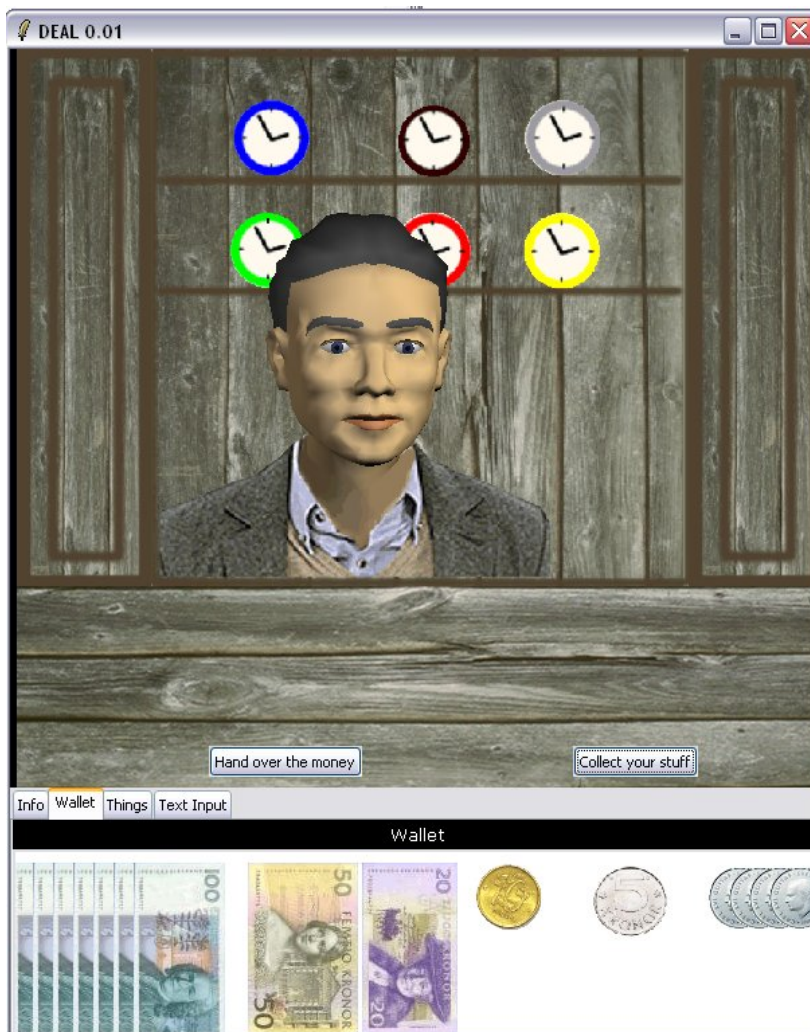


Figure 23: DEAL interface

6.3 Implementation in SCXML⁴³

A transaction can be modelled as consisting of three phases: An opening phase, in which the participants acknowledge each other, a trading phase, when the

⁴³ The original SCXML implementation is listed in Appendix III.

actual transaction takes place – including an optional bargaining phase, and finally an end phase, in which the deal is closed and the participants end the interaction. As in the following dialogue sample (where P represents the player and S the shopkeeper):

P1: Hi! I would like to buy a clock
S1: A clock.
S2: What colour did you have in mind?
P2: Do you have any green ones?
S3: Sure, what size?
P3: A small one, please.
S4: How about this clock?
P4: How much is it?
S5: It costs 250
P5: That is too expensive. I can pay 125
S6: How about 225?
P6: 150
...
S7: You can get it for 200, no less.
P7: Ok, that is a fair price, I'll take it.
S8: Fine, 200 then. [The user hands over the money, the shopkeeper hands over the object.]
S9: Thank you and welcome back!

Dialogue 11: A typical trade dialogue in DEAL

In Figure 24, below, an overview of the shopkeeper's dialogue manager is presented. The top-level state, *ShopkeeperDM*, contains three sub-states, one for each phase: *Opening*, *Trading* and *Ending*. Both *Opening* and *Trading* are compound states.

When the player utters P1, the shopkeeper's active state is *Opening*. After having established contact with the shopkeeper by a greeting, the customer expresses a request for a specific object, a clock, which initiates the trading phase. The shopkeeper checks the inventory, finds several matches and requests additional information (S2 and S3) in order to narrow down the search space. When the shopkeeper finally retrieves one unique match, the item is presented to the customer (S4).

Having agreed upon the item to trade, a negotiation process may begin, in this case initiated by the player in P5. After a series of offers and counteroffers, the two participants come to an agreement (S7 followed by P7). The shopkeeper confirms the price in S8, and then wraps up the dialogue when the customer has handed over the money.

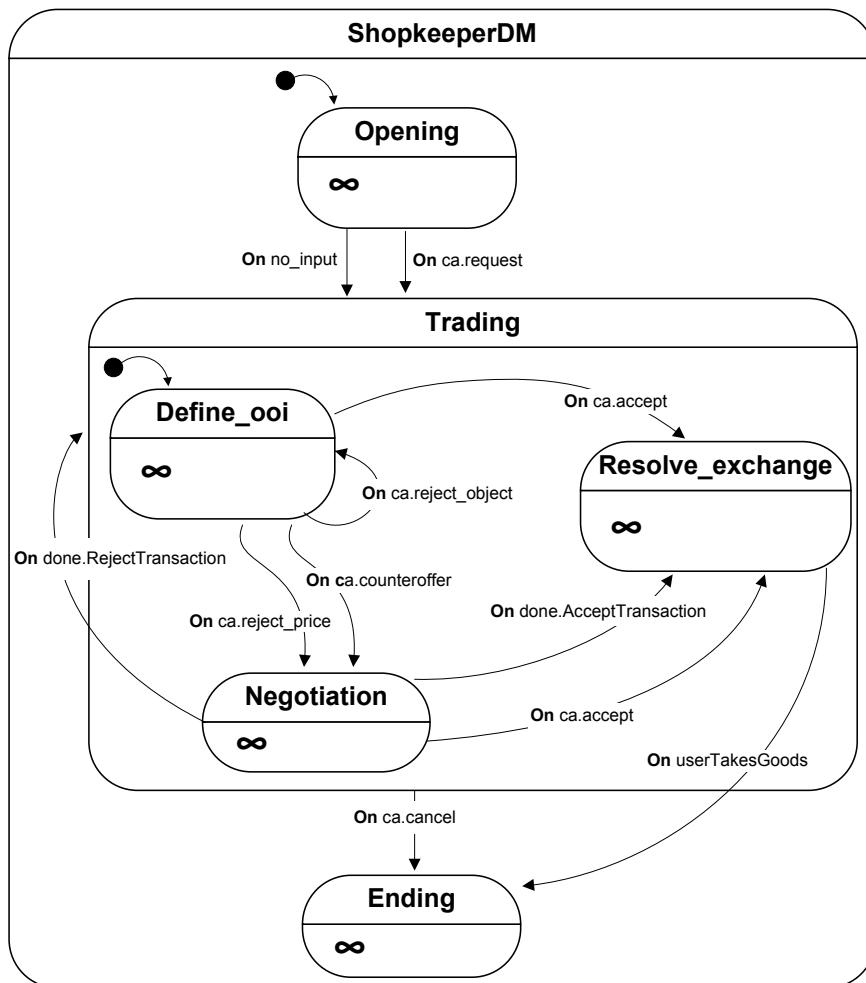


Figure 24: Dialogue manager for Shopkeeper in DEAL

Below, a detailed description of the different states – what they do and how they work – is given.

6.3.1 Opening

Figure 25, below, zooms in on the opening state of the shopkeeper’s dialogue manager. Either participant may initiate the dialogue, for instance with a greeting, but the player may also choose to immediately request an item which is an event that will trigger a transition to `Trading`. If the player is passive (detected by a so called “no input” event), the system may attempt to encourage the player to respond by offering assistance, which is done in the state `Offer_assistance`. The player can also specifically ask for help to get assistance, which also triggers a transition to `Offer_assistance`.

A transition to `Trading` is triggered when the user requests an object or does not provide an input at all. The former event is user-generated and the latter is

generated automatically if the player hasn't responded within a certain amount of time.

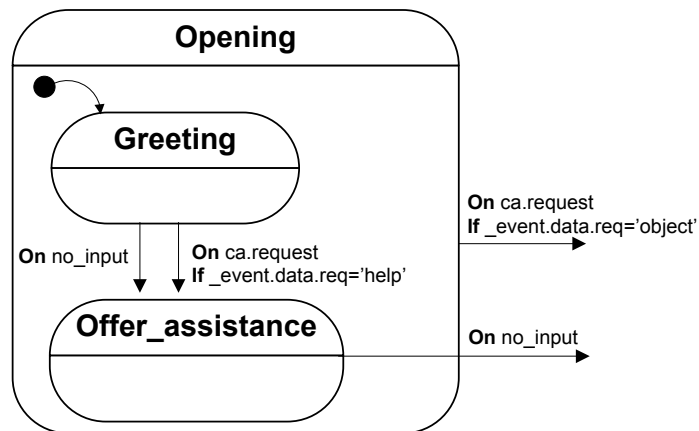


Figure 25: Opening

6.3.2 Trading

In the original version of the model that was published in (Brusk, Lager, Hjalmarsson, & Wik, 2007), the complex state `Trading` contained three substates: `Define_ooi` (an abbreviation of “define object of interest”), `Negotiation` and `Resolve_exchange` (see Figure 26, below), all of which are complex states themselves. Later, the model has been slightly modified to make better use of the advantages that statecharts offer. Below, a presentation of the original construction will be given, followed by a section presenting the improvement.

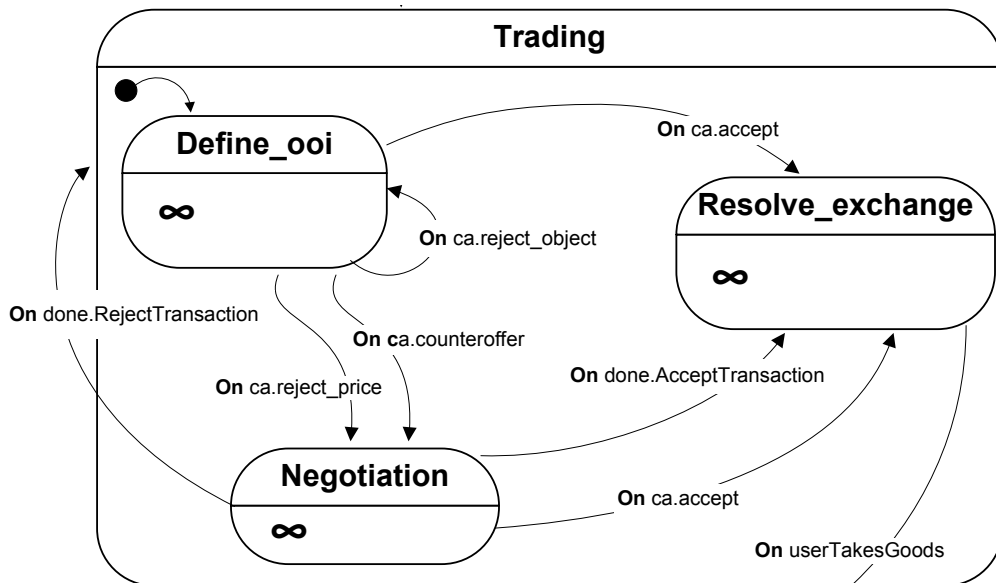


Figure 26: Trading

Define object of interest

A transaction can only be conducted if the shopkeeper is in possession of a specific item that the buyer is interested in purchasing and the buyer has something to trade in return (in this case money). The initial step in a trading situation is thus to find out what object the customer is interested in buying, here referred to as “define object of interest”, which in Figure 27 corresponds to the complex state `Define_ooi`.

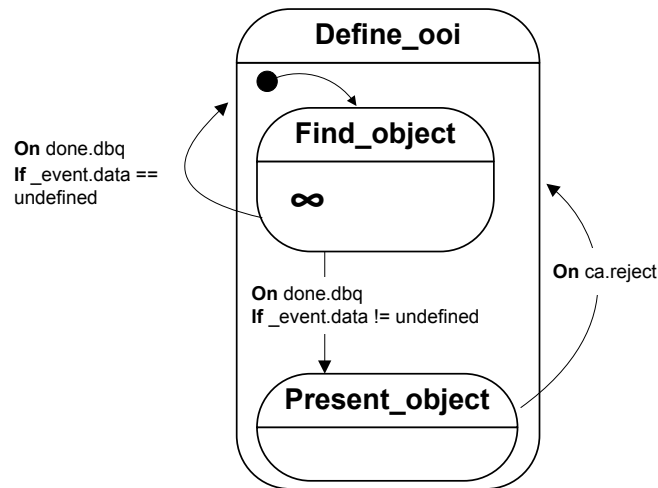


Figure 27: Define object of interest

`Define_ooi` contains two states: `Find_object` – a state for retrieving an object from the database, and `Present_object` – for presenting the object to the customer.

Figure 28, below, presents a more detailed view of `Define_ooi`. As can be seen in the picture `Find_object` is a complex state containing two states: `Get_info`, which is responsible for collecting the necessary data from the user; and `Search_object`, which does the actual database search based on this data. The model is inspired by form-based dialogue systems, such as VoiceXML, with regard to collecting the necessary information from the player. This means that when the system retrieves too many matches in a search, it tries to narrow down the search space by asking the player to specify the missing properties one by one until a match is found (`Get_info`) in a finite state-based manner. A more sophisticated system would of course try to ask only relevant questions, i.e. questions of distinctive features, but in this prototype we used a database with some limitations that prevented us from doing a search with more than one parameter at a time. Furthermore, the search is conducted in document order (see Appendix III). When a matching object has been found in the database a transition to `Present_object` is triggered, in which the shopkeeper presents the object to the user.

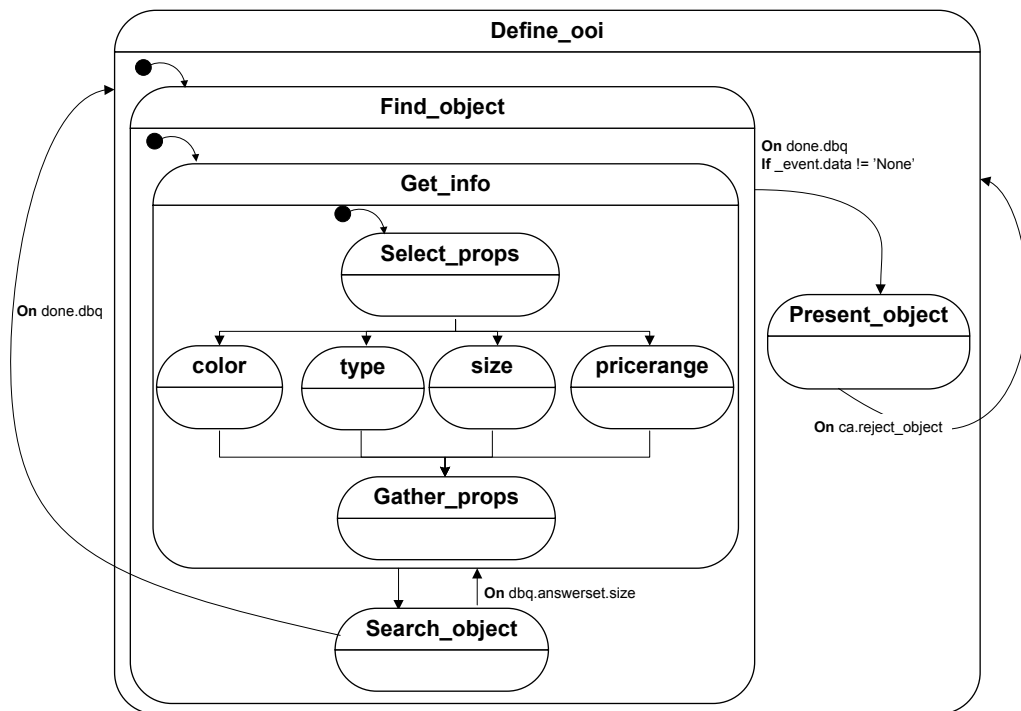


Figure 28: A detailed view of Define object of interest

There are three possible outcomes of `Define_ooi`: the user rejects the object or price, accepts the deal or responds with a counteroffer. There is actually also a fourth possible outcome, “cancel”, a transition that is leaving the top state `Trading` and therefore is implicit in each and every substate of `Trading` as well.

Negotiation

Negotiation is a complex process involving elements of social psychology and game theory and can also include emotional elements (e.g. affect) and lies. Our implementation so far handles offers and counteroffers. A next possible step could be to add complexity in terms of argumentation-based negotiation.

Negotiation can be viewed as a distributed search through a space of potential agreements (Jennings et al., 2001). The aim is to reach a point in the agreement space that will optimise the outcome, which at the same time is accepted by the other agent. The agents present offers and counteroffers based on parameters such as cost/utility and time. The shopkeeper's strategy used in this example has been inspired by the Zeuthen strategy, described in Jennings et al (2001), and the “automated bargaining game” presented in Tsang and Gosling (2002). In effect, the shopkeeper will give an offer based on his last offer (which has become the current price), the cost (minimum acceptable price), and time, corresponding to the number of rounds the shopkeeper has left to conclude the deal (see code sample in section 6.3.3, below). The buyer will also make offers according to those parameters, but the number of rounds will most likely be different, since it

is randomly selected for each agent. Neither agent has information about the other agent's time and cost/utility. It is assumed that neither of the agents is interested in reaching a breakdown and that they therefore will increase their willingness to conclude the deal for each round. The shopkeeper may for instance accept any price that exceeds the cost when running out of rounds. A breakdown may however occur when the shopkeeper has no more than one turn left and the latest bid from the buyer is below the cost or lower than her previous bid.

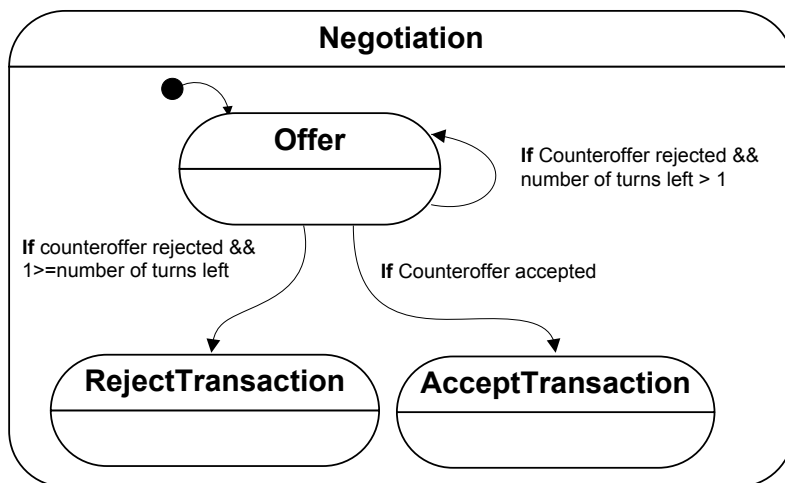


Figure 29: Negotiation

Figure 29 illustrates the negotiation statechart. The states `RejectTransaction` and `AcceptTransaction` correspond to the shopkeeper's response to the buyer's offers, while the buyer's offers are treated as events, where the actual event is a communicative act and the event data contains a type specifying the communicative act as an offer, accept or reject.

Resolve exchange

When the shopkeeper and the buyer have reached an agreement, the next step is to conclude the deal. In the complex state `Resolve_exchange` (Figure 30), the shopkeeper starts by confirming the agreed price (`Confirm_price`), after which the buyer is expected to hand over the money (triggering event). When the shopkeeper has received the requested amount of money, the state `Money_transaction` is entered. The shopkeeper thanks the buyer and hands over the goods and when the buyer takes the goods, the dialogue reaches the end.

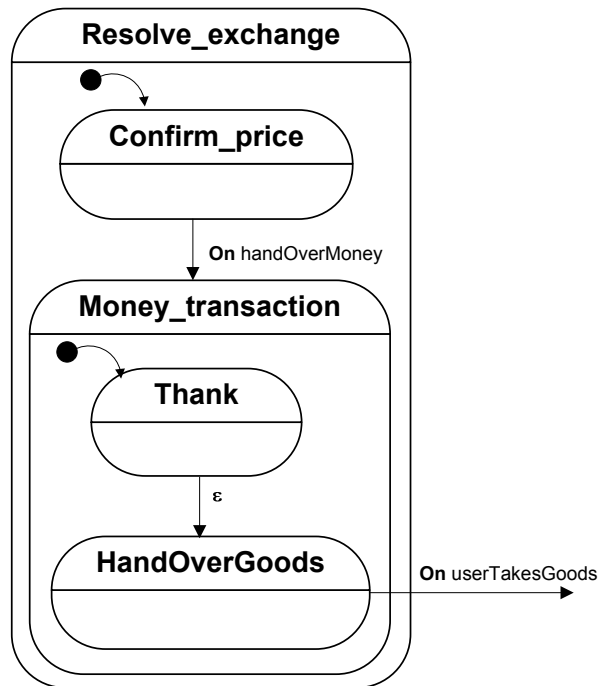


Figure 30: Resolve exchange.

Modification

When the shopkeeper and the buyer have come to an agreement, regardless of whether they have negotiated the price or not, a transition to `Resolve_exchange` is triggered. In the first model, both `Define_ooi` as well as `Negotiation` have a transition to `Resolve_exchange` triggered by the same event – “ca.accept”. This motivates a clustering of these states under a common super state to reduce the number of transitions. Let’s call the new state `ComeToAgreement`, to indicate that the participants have reached an agreement concerning both the item to trade (identified in `Define_ooi`) as well as the price (in either `Define_ooi` or `Negotiation`). By doing so, only one transition to `Resolve_exchange` in case of acceptance is needed. Figure 31, below shows the updated version of `Trading`. This means that when the buyer accepts the deal, the event is caught at the top-level state, i.e. in `ComeToAgreement` rather than in either `Define_ooi` or `Negotiation`.

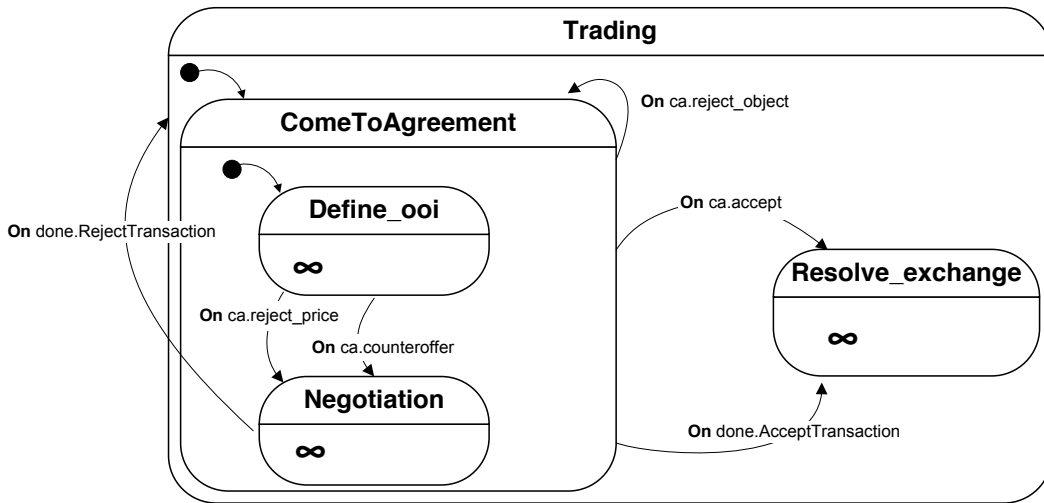


Figure 31: A modified Trading state

Ending

When the transaction is concluded or the trade has been cancelled, the application reaches the end phase, where the agents say goodbye to each other and exit the interaction.

6.3.3 Code Example – Negotiation

```

<state id="negotiation" initial="offer">
  <onentry>
    <assign location="Interval"
      expr="(Price-Minprice) / RoundsLeft"/>
    <assign name="Price" expr="Price-Interval"/>
  </onentry>

  <state id="offer">
    <transition event="ca"
      cond="_event.data.ca_type == 'counteroffer'
      && PreviousBid >=; _event.data.offer"
      target="offer">
      <!-- user offers same or lower bid than previous -->
      <assign location="RoundsLeft" expr="RoundsLeft-1"/>
    </transition>

```

```

<transition event="ca"
  cond="_event.data.ca_type == 'counteroffer' &
    & 1 &ge; RoundsLeft &&
    _event.data.offer > PreviousBid &&
    _event.data.offer &ge; Minprice"
  target="acceptTransaction">
  <assign location="Price"
    expr="_event.data.offer"/>
    <!-- Shopkeeper accepts price -->
</transition>

<transition event="ca"
  cond="_event.data.ca_type == 'counteroffer'
    &&_event.data.offer &ge; Price-Interval"
  target="acceptTransaction">
  <assign name="Price" expr="_event.data.offer"/>
    <!-- Shopkeeper accepts price-->
</transition>

<transition event="ca"
  cond="_event.data.ca_type == 'counteroffer'
    && 1 &ge; RoundsLeft &&
    Minprice > _event.data.offer"
  target="rejectTransaction">
</transition>

<transition event="ca"
  cond="_event.data.ca_type == 'counteroffer' &
    & (Price-Interval) > _event.data.offer"
  target="offer">
  <assign location="PreviousBid"
    expr="_event.data.offer"/>
  <assign location="Price"
    expr="{NextOffer Price PreviousBid
      RoundsLeft Interval}"/>
  <assign location="RoundsLeft" expr="RoundsLeft-1"/>
</transition>

<transition event="ca"
  cond="_event.data.ca_type == 'counteroffer'
    && Minprice > _event.data.offer"
  target="offer">
  <assign name="PreviousBid"
    expr="_event.data.offer"/>
  <assign location="Price"
    expr="{NextOffer Price PreviousBid RoundsLeft
      Interval}"/>
  <assign location="RoundsLeft" expr="RoundsLeft-1"/>
</transition>
</state>
</state>

```

6.4 Discussion

In this chapter we have given an example of how statecharts can be used to model the dialogue manager for a shopkeeper in a game. The example is derived from games designed with the patterns associated with ownership, as described by Björk & Holopainen (2005) where shops can be used as an arena for trading objects. The shopkeeper is capable of presenting objects of interest as well as negotiating the price. The dialogue allows mixed-initiative interaction, meaning that either participant can initiate and control the dialogue.

Compared to how trade usually is conducted in games, the NPC in this example is able to take a more active part in the interaction, both by allowing the player to take initiative (as in the initial request), but also taking initiative when there is a need for it (in this case when more information is required). The NPC can also start to negotiate when the player provides a counteroffer. The outcome of the situation becomes uncertain when we introduce the bargaining process – the price is under negotiation and neither participant knows what the final bid will be. In this particular case the participants came to an agreement leading to a completion of the trade, but situations will occur when the player changes a request, the shopkeeper doesn't supply the requested item or the transaction as a whole is cancelled.

Negotiation adds a challenge to a situation in games that usually are associated with a straightforward operation – to buy something you want or need and that you can afford. Unlike most other challenges found in games, negotiation requires social skills to succeed well. Hence, by introducing negotiation we also offer a socially oriented gameplay experience.

Chapter 7

Casual Conversations

In the previous chapters we have mainly been concerned with practical dialogues and how to make them more socially interesting. The next step now is to explore the possibilities for creating NPCs that have the ability to engage in casual conversations, that is, conversations that are motivated by “interpersonal needs” (Eggins & Slade, 1997), such as small talk and gossip. These types of conversations could be very useful in games, both as an instrument to build and maintain social relationships among game characters, but also to spread news, manipulate, and to create tension between the characters.

This chapter starts by presenting various solutions to how small talk may be integrated in a dialogue model, such as the ones presented earlier. The major part of the chapter is however dedicated to gossip conversations, starting with a review of previous research. The result from conducting two experiments on gossip conversations is then presented, and finally, a statechart model for initiating a gossip conversation is introduced.

7.1 Small Talk

A common way to socialize with other people is through small talk. Small talk is a type of casual conversation that can have different functions depending on when it occurs in the conversation. If it is initiated in the beginning of a conversation, the purpose is usually to build rapport and establish a common ground by revolving around safe topics that the speakers can agree upon (Brown & Levinson, 1987). Small talk can then help to “break the ice” and soften a more face-threatening act, such as a request (Brown & Levinson, 1987; Bickmore & Cassell, 2000, 2001). When it appears in a middle of a conversation, i.e. when other topics already have been dealt with, small talk can be used to fill awkward pauses – i.e. maintain contact if conversation is running slow or if the agents do not know what to talk about or how to act. Small talk can also be used as an “escape strategy” to avoid losing face – e.g. by switching to a “safe” topic when the threat to face is becoming too high. Small talk can to some extent also reveal the agents’ personality, and be used in games by the NPCs to provide general

information about the game world and thereby make less prominent characters in the game more interesting to interact with.

Small talk may have an important role to fill in a practical dialogue system. Bickmore & Cassell (2001), for example, found that small talk could be used to increase the trust between a user and their virtual real-estate agent (Rea). Their experiment however showed that this effect only was noticeable when the user had an extrovert personality type, it had no significant effect for introvert users.

The question is whether the same is true for the relationship between game characters and players? Even if it is so one could, on basis of studies made over player types (Bartle, 1996; Yee, 2005), also assume that other factors may influence how the player perceives a non-playable character. A “socialiser” might for instance prefer small talk while an “achiever” perhaps regards it as a waste of time unless some reward is expected from it.

7.1.1 Integrating Small Talk with the Opening Phase

Let’s say we want our waiter to be able to engage in a dialogue such as the following:

- (1) Waiter: Welcome!
- (2) PC: Hi
- (3) PC: Nice weather today, isn’t it?
- (4) Waiter: Yes, indeed!

Dialogue 12: A simple small talk dialogue between a waiter and a player character

In the conversation the participants acknowledge each other through a greeting followed by small talk as an extension to the greeting. Even if small talk in some situations may be regarded as a social obligation it is typically optional, but some form of acknowledgement is required in order to start the conversation (see e.g. Bunt, 1994). One way to model the dialogue above is therefore to organize the two sequences hierarchically under the umbrella of a superstate `Opening`, as in Figure 32, below.

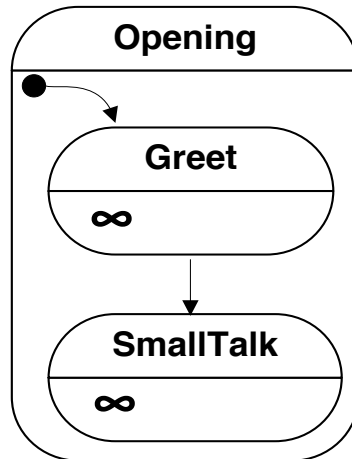


Figure 32 Opening with small talk module

Since a dialogue naturally begins with an exchange of greetings, it follows that `Greet` is the default start state of `Opening` followed by an optional small talk phase. There are several possible options for a transition from `Greet` to `SmallTalk` to take place: it can be an empty transition such that it is taken as soon as the `Greet` state's on-entry and on-exit scripts have been executed; it can be triggered by a timeout event; or be activated as soon as the player has returned the greeting.

The advantage of grouping small talk with greeting is that it is treated as part of the opening phase, which is conceptually reasonable. It doesn't matter whether the agents actually small talk or not. If the purpose for initiating the dialogue is different from just engaging in small talk, for example to solve a particular task, placing the small talk module in this position serves the function of softening the more face-threatening acts to come. It can also serve to help the player getting introduced to interacting with the agent. The disadvantage is that when small talk is hierarchically organized under the `Opening` state, it cannot be accessed once left.

7.1.2 Giving Small Talk the Same Status as the Practical Conversation

Encapsulating small talk together with greeting has the effect that the agents cannot start to small talk at any other point during the conversation. If we want to provide that option, we may take advantage of the modularity that statecharts provide by placing small talk on the same level as the practical conversation. This solution enables switches between the two dialogue types as is shown in Figure 33, below.

DialogueIdle state serves as a transient state (see Horrocks, 1999) that selects the next state (Practical, SmallTalk, or Closing) depending on the value of the communicative act as given in the event data – assuming that the player’s act of saying something, i.e. the locutionary act (Austin, 1962), is treated as a game event. Ending is a final state that terminates the current dialogue.⁴⁴

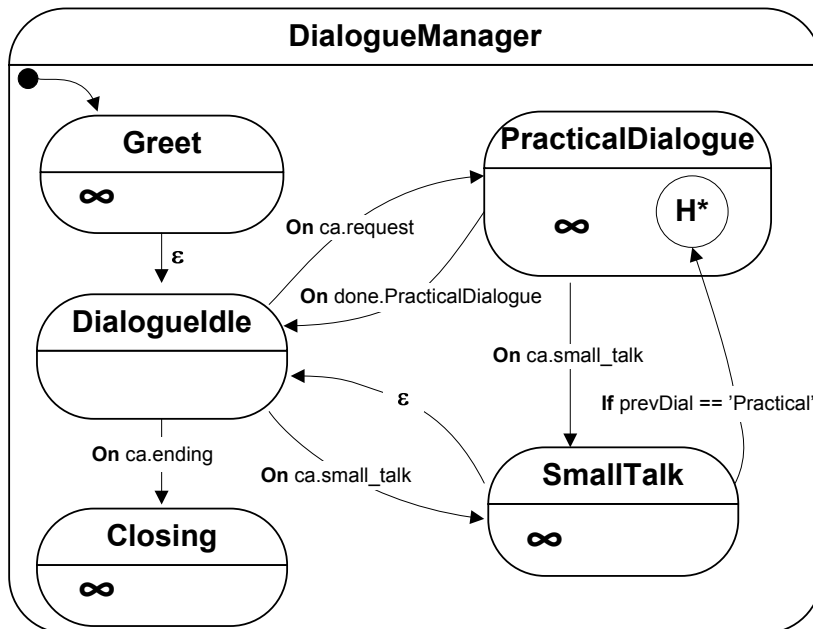


Figure 33: Dialogue manager for switching between small talk and practical dialogue

In the model above the practical conversation can be resumed through the use of a history state (H*) if it was left before completion. If the practical conversation is initiated from DialogueIdle it is assumed that a new session of a practical conversation is starting, thus explaining the transition to the start state of PracticalDialogue. When SmallTalk is terminated, the system needs to return to the state from where the transition to SmallTalk was taken. If the last state visited was PracticalDialogue the target state must be the history state H*, otherwise the practical conversation is irreversibly interrupted and has to start from the beginning again. In order to determine whether to return to PracticalDialogue, we have chosen to save the source state in a variable “prevDial”, which is then used in the conditional expression on the transition to PracticalDialogue. If evaluated to true, a transition to H* in PracticalDialogue is taken otherwise SmallTalk will return to DialogueIdle. Since SCXML code is executed in document order we do not need to evaluate the expression once again; instead, an empty transition is taken

⁴⁴ Transient states are states that may be chosen as a target state if there are several possible target states triggered by the same event. None of the transitions leaving a transient state is triggered by events. Instead, the transitions are taken under certain conditions. Since there are no events associated with the transitions leaving the state, the conditions on the event arrows are evaluated on entry to the state and the next state is entered immediately.

to `DialogueIdle` in case none of the previous transitions are executed. This means that `DialogueIdle` is the default target state of `SmallTalk`. Since `SmallTalk` can be accessed in any order, it can still be initiated directly after the greeting phase to reach the same effect as in the previous version.

Apart from the sequential nature of the practical conversation *per se* (with an opening, task-handling and closing), this model offers a non-sequential treatment of the different topics. A topic switch is allowed at any time and either topic can be selected first, i.e. none of the topics is considered default.

7.1.3 Giving Practical Conversation Precedence

To create NPCs that appear to be capable of keeping up long lasting conversations using natural language is a great challenge, but also a great risk. The greatest risk is to cause player frustration due to lack of understanding or failure to progress the dialogue. If the player has no idea of what to say or talk about the NPC must take initiative and move the dialogue forward. A game dialogue system should also be designed to give the illusion that the NPC is more competent than it really is. Some strategies that come to mind are (see also section 2.5): first, to offer various error handling strategies, e.g. when the system does not understand it can act as if it did not pay enough attention or start talking about something else instead of insisting to reach understanding as practical dialogue systems usually do (by for instance re-prompting). Second, it could appear to have social skills, for instance by acknowledging the status of the interpersonal relationship and act accordingly. Third, instead of encouraging small talk and other types of casual conversations it can merely support them, i.e. handle casual conversations to some degree, but give precedence to practical conversations and use them as default. This way it is more likely that the conversations can reach a completion before the player has discovered the NPC's limitations.

The model presented in Figure 33, above, can be adjusted to support these strategies. By removing the idling state and force transitions to always pass `PracticalDialogue`, but not necessarily the `SmallTalk` state, we naturally give precedence to the practical dialogue. This has the advantage that even if the module is included, it can be ignored by the system under certain circumstances, for example if the system has detected that the player uses a style that is inconsistent with spending time on small talk. An illustration of the alternate model is shown in Figure 34, below.

The advantage with this solution is that the NPC will have as its main goal to complete the practical dialogue, but can easily choose to start small talking when it fails to understand the player. In fact, both the practical conversation as well as small talk can be regarded as error handling strategies, because when the NPC

fails to understand something during the practical dialogue it can find its resort by starting to do small talk, and if the small talk starts to turn into a more complex form of a casual conversation, for instance gossip, that it cannot handle it can naturally return to or initiate the practical dialogue.

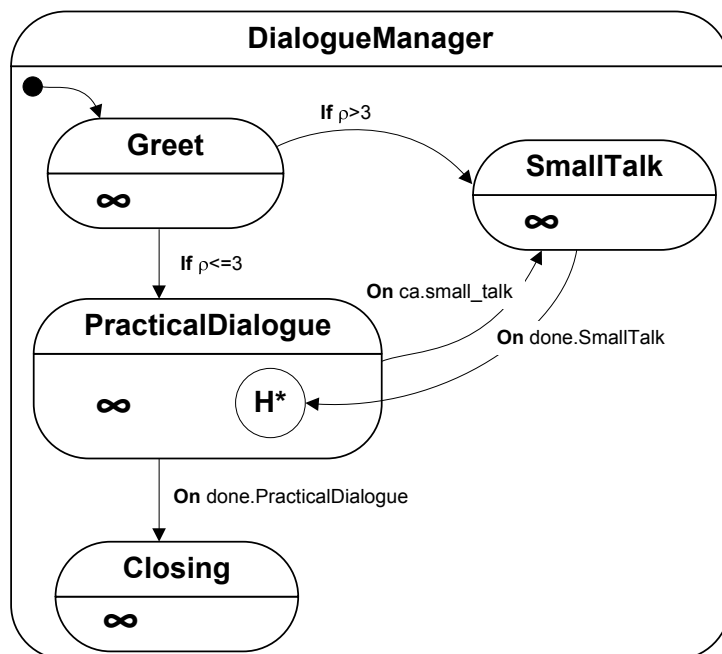


Figure 34: Giving the practical dialogue precedence

7.1.4 Implementation

An example of a practical conversation has already been given; different now is the inclusion of a history state to enable the practical dialogue to be resumed. The main structure of the example depicted in Figure 34 above, would in SCXML notation be as follows:

```

<state id="DialogueManager" initial="Greet">
  <state id="Greet"../>
  <state id="SmallTalk"../>
  <state id="PracticalDialogue">
    ...
    <history id="h"../>
  </state>
  <state id="Closing"../>
</state>
  
```

The previous examples in this thesis have relied on VoiceXML as presentation layer, but VoiceXML may not be the most suitable presentation layer for casual conversations as it is tailored for filling forms in a specific domain. Instead, the

SmallTalk state may invoke a dialogue manager/ASR/TTS combination able to deal with ‘social chat’, such as AIML for example (Wallace, 2005):

```
<state id="SmallTalk">
  <invoke id="c" targettype="AIML" src="chat.aiml"/>
  <transition event="c.done" target="h".../>
</state>
```

Another solution is to control which dialogue manager to invoke by means of a meta-dialogue manager as was discussed in chapter 4.5.

Transition from SmallTalk will always target the history state within PracticalDialogue, since small talk can be initiated from either Greet or PracticalDialogue.

7.2 Gossip

Any sociological attempt to understand gossip must live with the fact that, long before sociology appeared on the scene and made it the subject of scientific investigation, gossip was a social phenomenon of daily life about which actors had their own ideas and made their own judgments. From our everyday experience we know what is meant when a conversation is called a coffee-klatsch. We do not need to resort to a dictionary when a person is called a gossip. (Bergmann, 1993, p. 1)

People seem to have their own intuitive understanding of gossip and there is not yet a clear consensus on how gossip should be defined. Most of the definitions are either too vague or too general to be useful. The Merriam-Webster online dictionary, for example, defines gossip as “rumour or report of an intimate nature” and “chatty talk”, neither of which is specific enough. What we need is a working definition that (a) matches people’s intuitive notion of gossip to the extent possible, given that the notion itself is somewhat vague, and (b) is sufficiently precise to provide a basis for computational implementation.

A number of definitions (e.g. Eder and Enke, 1991; Eggins and Slade, 1997; Hallett et al., 2009) have been derived from analysing transcriptions of real gossip conversations. These definitions have only minor individual differences and can in essence be formulated as “evaluative talk about an absent third person”. This is also the definition that will be used as a starting point here.

7.2.1 Background

Gossip has been described as a mechanism for social control (Gluckman, 1963; Fine & Rosnow, 1978; Bergmann, 1993; Eggins & Slade, 1997) that maintains “the unity, morals and values of social groups” (Gluckman, 1963). It has furthermore been suggested that gossip is a form of “information-management”, primarily to improve one’s self-image and “protect individual interests” (Paine,

1967), but also to influence others (Szwed, 1966; Fine & Rosnow, 1978). Gossip can furthermore be viewed as a form of entertainment (Abrahams, 1970) – “a satisfying diversion from the tedium of routine activities” (Fine & Rosnow, 1978).

Recent studies have used a sociological approach focusing on analysing the structure of gossip conversations (Bergmann, 1993; Eder & Enke, 1991; Eggins & Slade, 1997; Hallett et al., 2009). Rather than observing and interviewing people in a certain community about their gossip behaviour, these studies present analyses of transcripts of naturally occurring gossip conversations. The studies show that gossipers collaborate in creating the gossip, making it a highly interactive genre. They also identified two key elements of gossip:

- Third person focus – the identification of an absent third person that is acquainted with, but emotionally disjoint from the other participants. A person that is emotionally related to either of the participants may be regarded as being “virtually” (Bergmann, 1993) or “symbolically” (Goodwin, 1980) present.
- An evaluation of the person in focus or of his or her behaviour. Eggins & Slade (1997) propose that the evaluation necessarily is pejorative to separate gossip from other types of chat. Instead of claiming that the evaluation necessarily is pejorative, Hallett et al. (2009) suggest that it is “unsanctioned”.

In addition to the two elements described above, Eggins and Slade (1997) propose a third obligatory element:

- Substantiating behaviour – An elaboration of the deviant behaviour that can either be used as a motivation for the negative evaluation, or as a way to introduce gossip in the conversation. Eder & Enke (1991) use a different model, but the substantiating behaviour component corresponds roughly to their optional *Explanation act*.

There seems to be a consensus that gossip conversations have third person focus. The question is whether a gossip conversation necessarily has both a substantiating behaviour component as well as a pejorative evaluation component, and if they do, can they be identified? In the experiments presented later in this chapter, we hope to shed light on whether these components are necessary or not.

7.2.2 The Activity

In terms of a social activity, gossip can be analysed as follows:

Purpose: We have already touched upon and given a number of different reasons why people gossip: To exchange information, to negotiate accepted and

appropriate behaviour, and to establish and maintain group membership. We also engage in gossip simply because we find it entertaining.

Roles: Gossip, just like other types of casual conversations, has an unclear role distribution. The roles are symmetric in the sense that none of the participants are higher ranked than the other and the social distance between the participants is rather low; in fact, gossip requires that ρ is fairly low. The participants must know each other well enough and have some knowledge of or connection to the person in focus and none of the participants should be ranked higher in status than each other. Gossip about a colleague is for example highly inappropriate between a boss and a lower ranked employee. Gossip between equally ranked employees are however acceptable, especially when the target in focus is the boss. Gossip is thus highly regulated by certain social rules that participants must obey in order to succeed with the gossip. For a gossip conversation to take off, the hearer(s) must agree or at least show acceptance. A gossip initiation may fail if any of the hearers disagrees, or even defends the target or the target's behaviour. In that case a gossip conversation may turn into a debate, where the first utterance may be treated as an opinion and the responses as reactions or counterarguments to that opinion (see for example an investigation of the opinion genre in (Horvath & Eggins, 1995)).

Artefacts: Gossip is independent of artefacts; it is a socially oriented conversation that revolves around people rather than objects.

Physical and social environment: The participants must be physically (and to some extent socially) distant from the gossip target. Gossip typically takes place in a "safe" environment among friends.

7.3 Gossip in Fictional Stories

Since a game is placed in a fictional setting, it seems natural to study gossip occurring in other fictional settings, such as in movies or books. As we see it, there are some significant similarities between dialogues in screenplays and game dialogues: (1) They are both tailored to fit a particular scene, which means that they have a natural beginning and end as well as a language use that is consistent with both the role characters as well as the overall theme. (2) They are based on fiction and just like the characters in games present accentuated interpretations of human behaviour; role characters are imaginary figures with accentuated human behaviours. (3) One could say that scripted dialogues reflect "ideal" conversations, i.e. conversations in which all uninteresting and unnecessary parts have been removed; hence they are already distilled (Larsson et al., 2000). One major difference is of course that in a game one of the role characters is played by a human. One must also take into account that scripts are adjusted for an

audience, i.e. the information exchanged in these conversations tends to either be more explicit and extensive than in ordinary conversations, or accompanied with pictures that allow the dialogue to be reduced. So even if they are distilled, other complexities may have been added to these dialogues.⁴⁵ In a way it is the same with games, some information must be explicitly provided to the players in order for them to be properly grounded. In games so called cut scenes, i.e. non-interactive sequences, are often used for this purpose.

There are thus some considerations to account for:

- A screenplay is created for a viewer, which means that some parts are left out and only implicitly implied.
- It is assumed that the dialogue is presented visually; hence non-verbal cues are commonly used (“show not tell”).
- The dialogues have been written to keep the viewer interested and engaged – it does not contain long, irrelevant exchanges.
- While the audience of a movie or TV-show only participate passively, a player of a game takes an active part in progressing the story. This means that one role has to be assigned to the player.
- Humour is important

A player of a game is actively engaged in performing actions that affect how the story is being told, whereas the story in a movie is narrated independently of the audience’s interferences. That is, the narrative content (story) may be the same but the narrative discourse (see e.g. Chatman, 1978) – the way the story is told – may differ. In this sense, interacting with a game character is similar to interacting with a typical ECA. They both serve as an interface to an underlying system. But when the ECA typically is used as a substitute for a human, with whom users communicate using their real identities, a game character has been given a role. And when players interact with the game character they are expected to play their part, i.e. through the projective identity (see 3.2.1).

Since gossip implies and requires many other social aspects of interaction apart from the mere gossip conversation, an NPC that can engage in gossip will also have the ability to use their social skills in other contexts as well. This includes acknowledging the interpersonal relationship and identifying and placing themselves in the social hierarchy or social network and adjusting their behaviour accordingly. These capabilities open up for a range of new types of gameplay options, for instance gameplay based on social relations or dialogue games. Gossip can also serve as a vehicle for story progression as well as support the pattern INFORMATION PASSING (Lankoski & Björk, 2007b). Thus, there are good reasons for introducing a more in depth treatment of the gameplay design pattern

⁴⁵ Thanks to Dr. Anton Leuski at USC/ICT for pointing that out.

Gossip as it was introduced in (Lankoski & Björk, 2007b). Here the focus will be on the conversational structure, while the social consequences beyond direct interpersonal relationship between the participants will be ignored.

7.4 A First Attempt to Model Gossip

The structural analyses of gossip made by Eder and Enke (1991) and Eggins and Slade (1997), show that when the gossip target has been introduced almost any gossip element can follow. This means that the structure is unpredictable and highly complex. Since a gossip conversation in a game or a movie should be limited to a particular scene, we started by investigating whether it would be possible to treat gossip as belonging to the “opinion genre” with which it shares many similarities. The opinion genre has been suggested to be an expression of an attitude towards some person, event or thing (Horvath & Eggins, 1995; Eggins & Slade, 1997). The obligatory elements of opinion are however less than those constituting gossip, and consists solely of an *opinion* followed by a *reaction*, which is a response ranging on a scale from disagree to agree. When a reaction for example involves a “request for evidence”, the structure however becomes more complex. In this case, the conversation may have elements of “evidence” and finally a “resolution” (given that the hearer accepts the evidence). This dialogue from the TV-show *Desperate housewives* (2004) demonstrates the similarities between gossip and opinion:

| Speaker | Utterance | Gossip | Opinion |
|-----------|--|--------------------------|------------------|
| Gabrielle | Can I say something? I'm glad Paul's moving | Third person focus | Opinion |
| Bree | Gaby! | Rejection/Encouragement | Objection/denial |
| Gabrielle | I'm sorry, but he's just always given me the creeps. Haven't you guys noticed? | Substantiating behaviour | Provide evidence |
| Gabrielle | He has this dark thing going on. There's some-thing about him that just feels... | Pejorative evaluation | Provide evidence |
| Lynette | Malignant? | Pejorative evaluation | Agree |
| Gab | Yes | Acknowledgement | Agree |
| Susan | We've all sorta felt it | Agree | Agree |
| Bree | That being said, I do love what he's done with the lawn | Wrap-up | Wrap-up |

Table 10: A comparison between gossip and the opinion genre

From this first analysis, we created a simple dialogue model using statecharts, presented in Figure 35 in below.

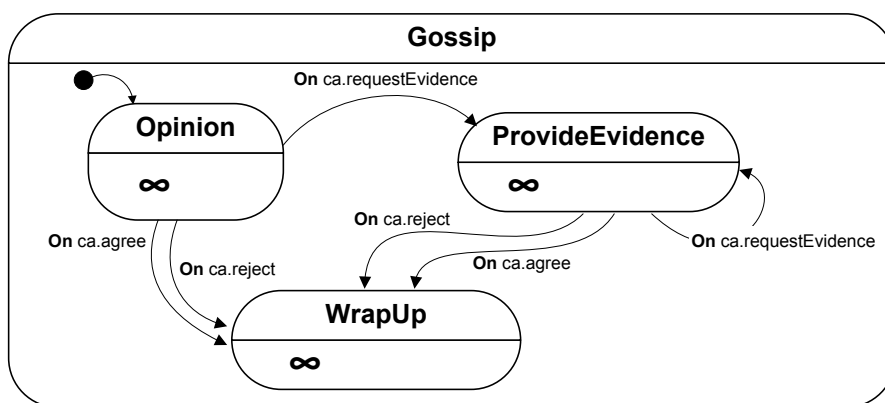


Figure 35: A model of gossip based on the opinion genre

This model is a reflection of a particular dialogue occurring in a specific scene and constructed based on the constituents of the opinion genre. By regarding gossip as a form of opinion, we managed to create a simpler dialogue model while still capturing the whole gossip scene.

Some of the responses to an opinion could be exchanged by another dialogue function, but a specific utterance could also be interpreted in various ways. For instance, even though Bree’s response appears to be an objection (“Gaby!”) (line 2 of table 1), the effect is more or less the same as if she would have requested evidence (e.g. by a probe, such as “why?” or “how so”). That is, Gabrielle responds by providing an explanation for her opinion rather than discarding the gossip. If Bree instead would have said “me too”, in a dialogue between just the two of them, the gossip could be completed immediately and Gabrielle would not have to substantiate her statement (as in line 3), instead the dialogue could be wrapped up. If Bree’s objection had been more convincing, it is likely that Gaby would have chosen to withdraw her statement and thereby discard further gossip. Worth noticing is that the `ProvideEvidence` state can be iterated, meaning that the initiator may have to convince the addressee further.

We implemented a prototype of the model by cutting out the phrases and listing them according to the following schema:

| Function | Iter. | Utterance |
|----------|-------|--|
| Initiate | 1 | I am glad Paul is moving out |
| ProvideE | 1 | He is just always giving me the creeps. Haven't you noticed? |
| ProvideE | 2 | He has this dark thing going on |
| ProvideE | 3 | There is something about him that feels malignant |
| WrapUp: | 1 | That being said, I do love what he has done with the lawn |

Table 11: Schema for canned gossip responses

The NPC can start a gossip conversation by retrieving the first phrase of the “Initiate”-type. If the user then asks for more information or a clarification using typed text, such as “how so?” the NPC will provide evidence (*ProvideE*) to substantiate the opinion or negative evaluation unless the last iteration has been met. When the list of available responses runs out, the NPC can wrap up the conversation. The NPC was implemented as a SitePal-agent⁴⁶ (a talking head) invoked from the SCXML process. The SCXML code is found in Appendix IV. The dialogue manager was implemented in SCXML while VoiceXML was used for the presentation layer. The grammar was limited to a few probes and communicative acts of agreement. Still, the implementation showed that it is possible to create limited gossip conversations as opinion using natural language input to CANNED TEXT RESPONSES (see 3.7), but the implementation does not take into account the interpersonal relationships between the participants and their relation to the target, which are important aspects of gossip. The next step is therefore identify what is required in order to build a complete model of a gossip conversation, but in order to do so we first need a working definition of gossip.

7.5 Two Experiments on Gossip Conversations

In order to investigate how people understand and identify gossip, two experiments on gossip conversations were conducted. The aim of the experiments was to verify to what extent the definition of gossip accords with intuitive recognition of gossip episodes, and secondly whether people could reliably identify constituent elements.

The data was collected using online questionnaires that were distributed through different email-lists mainly targeting researchers and students within game design, language technology, and related fields, located primarily in North America and Europe. The lists were chosen to reach a large number of people at the same time, but it should be noted that most of the participants were academics. The questionnaires had the following structure: The first page consisted of an introduction, including instructions, and each page thereafter had a dialogue excerpt retrieved from a screenplay followed by the question and/or task.

⁴⁶ <<http://www.sitepal.com/>>

7.5.1 Hypotheses

Based on the previous studies presented earlier (in particular Bergmann, 1993; Eder & Enke, 1991; Eggins & Slade, 1997) the following hypotheses were formulated:

- The more gossip elements present in the text, the more likely the conversation will be considered gossip.
- Third person focus is a necessary (but not sufficient) element of gossip.
- Conversations in which the participants (including the target) are intimately related will be less likely to be rated as gossip than conversations in which all participants are emotionally disjoint.

7.6 Experiment I: Identifying Gossip Text

The aim of the first experiment was to investigate how people intuitively understand and interpret gossip conversations.

7.6.1 Material and Procedure

The questionnaire contained 16 different dialogue excerpts retrieved from transcripts of the famous sitcoms *Desperate Housewives*⁴⁷ and *Seinfeld*⁴⁸. The excerpts were selected to cover different combinations of the elements presented in the previous section (third person focus, an evaluation, and a motivation for the evaluation), as in the following dialogue:⁴⁹

- B: Tisha. Tisha. Oh, I can tell by that look on your face you've got something good. Now, come on, don't be selfish.
- T: Well, first off, you're not friends with Maisy Gibbons, are you?
- B: No.
- T: Thank god, because this is too good. Maisy was arrested. While Harold was at work, she was having sex with men in her house for money. Can you imagine?
- B: No, I can't.
- T: And that's not even the best part. Word is, she had a little black book with all her clients' names.
- R: So, uh ... you think that'll get out?
- T: Of course. These things always do. Nancy, wait up. I can't wait to tell you this. Wait, wait.

⁴⁷ © Touchstone Television

⁴⁸ © Castle Rock Entertainment.

⁴⁹ Excerpt retrieved from *Desperate Housewives*, Touchstone Television.

We made a preliminary analysis to determine whether the elements were present or not (see Table 12, below).

| Q | 3 rd p | Pej. Eval | Subst. Behav. | Gossip |
|----|-------------------|-----------|---------------|--------|
| 1 | • | ⊙ | • | • |
| 2 | – | • | – | – |
| 3 | • | • | – | ⊙ |
| 4 | • | • | – | ⊙ |
| 5 | • | – | • | ⊙ |
| 6 | – | – | – | – |
| 7 | • | • | • | • |
| 8 | • | • | • | • |
| 9 | ⊙ | • | • | – |
| 10 | ⊙ | • | • | ⊙ |
| 11 | – | – | – | – |
| 12 | • | • | • | ⊙ |
| 13 | ⊙ | • | • | ⊙ |
| 14 | • | • | • | ⊙ |
| 15 | – | – | – | – |
| 16 | • | • | – | – |

Table 12: A preliminary rating of all excerpts.⁵⁰

The instructions contained no information about the elements and no definition was given. To each excerpt we provided some contextual information, such as the interpersonal relationship between the speakers and other people mentioned in the dialogue, e.g.:

The married couple, Bree (B) and Rex (R) Van de Kamp, are having lunch at the club. Some women laughing at the next table cause the two of them to turn and look. One of their acquaintances, Tisha (T), walks away from that table and heads to another one. Maisy Gibbons is another woman in their neighbourhood, known to be very dominant and judgmental towards the other women.

The subjects were asked to read and rank the excerpts using the following scale:

- Absolutely not gossip

⁵⁰The symbols used in the table should be read as follows: (•) element present, (–) element not present, and (⊙) ambiguously present.

- Could be considered gossip in some contexts
- Would be considered gossip in most contexts
- Absolutely gossip

For the purpose of analysis we converted the above responses to integers from 0 to 3.

7.6.2 Results

A total of 52 participants completed the experiment. Table 13 shows the distribution of ratings for each of the 16 excerpts (the table is sorted by the mean rating).

| ID | Rating distribution | | | | Mean rating |
|----|---------------------|----|----|----|-------------|
| | 0 | 1 | 2 | 3 | |
| 11 | 50 | 1 | 1 | 0 | 0.058 |
| 6 | 46 | 5 | 0 | 1 | 0.154 |
| 15 | 33 | 15 | 4 | 0 | 0.442 |
| 2 | 28 | 20 | 4 | 0 | 0.538 |
| 5 | 30 | 15 | 6 | 1 | 0.577 |
| 10 | 17 | 24 | 10 | 1 | 0.904 |
| 9 | 10 | 26 | 13 | 3 | 1.173 |
| 16 | 11 | 17 | 16 | 8 | 1.404 |
| 4 | 8 | 18 | 18 | 8 | 1.500 |
| 14 | 11 | 13 | 11 | 17 | 1.654 |
| 3 | 6 | 20 | 11 | 15 | 1.673 |
| 1 | 1 | 17 | 25 | 9 | 1.808 |
| 13 | 3 | 18 | 17 | 14 | 1.808 |
| 12 | 5 | 9 | 15 | 23 | 2.077 |
| 8 | 3 | 0 | 11 | 38 | 2.615 |
| 7 | 1 | 2 | 4 | 45 | 2.788 |

Table 13: Gossip ratings of all 16 questions sorted by their mean value

It is apparent from the table that a few excerpts are clearly gossip or clearly not gossip, but there is much disagreement on other excerpts. Inter-rater reliability is $\alpha = 0.437$: well above chance (0), but not particularly high.⁵¹ Only 7 of the 16 excerpts (ID # 2, 5, 6, 7, 8, 11, 15) were clearly rated as gossip or not gossip by more than half of the subjects, and only 5 of those have a mean rating below 0.5 or above 2.5.

Despite the apparently low agreement, the results correspond fairly well with our expectations (c.f. Table 12). The 3 excerpts with a mean value below 0.5 had no gossip elements at all and the other two excerpts with a median value of 0 had only one gossip element. Similarly, the two excerpts rated highest clearly had all gossip elements (7 & 8). The rest of the excerpts, however, either lacked one element or had one element that was un-clear in some regard (see discussion, below). Conversations between family members or partners also caused higher disagreements, which seem to support Bergmann's (1993) remark: "[...] we can ask whether we should call gossip the conversations between spouses [...] alone. This surely is a borderline case for which there is no single answer" (p. 68).

7.6.3 Discussion

Among the nine excerpts with a mean value approximately between 1 and 2 (ID #1, 3, 4, 9, 10, 12, 13, 14, and 16), we made the following observations: 3 excerpts lacked one element; in 2 of them, the gossipers were family members or partners; 3 excerpts had an ambiguous focus, among which one also possibly was perceived as a warning.

By "ambiguous focus" we mean that it is unclear whether the person in focus is the speaker, the addressee or the absent third person. In the first two cases, the absent third person seems to play a sub-ordinate role rather than focused role, for instance as part of a self-disclosure or a confrontation. If the conversation is the least bit confrontational, the addressee tends to go into defence rather than choosing a more typical gossip response, such as support, expansion, or challenge (Eder & Enke, 1991) in order to protect the face. Hence, no "gossip fuel" is added to the conversation.

The result of the excerpt below (corresponds to dialogue #14 in table Table 12) is however more difficult to explain. One possible explanation is that the initiator was unacquainted with the target, but perhaps more likely is that some of the subjects interpreted the conversation as mocking rather than gossip:

⁵¹ The reported value is Krippendorff's α with the interval distance metric (Krippendorff 1980). Interval α is defined as $1 - Do/De$, where Do (observed disagreement) is twice the mean variance of the individual item ratings, and De (expected disagreement) is twice the variance of all the ratings. For the above table, Do = 1.327 and De = 2.585.

- E: Who's that?
D: That's Sam, the new girl in accounting.
W: What's with her arms? They just hang like salamis.
D: She walks like an orangutan.
E: Better call the zoo.

Dialogue 13: Dialogue example from questionnaire

7.7 Experiment II: Identifying Gossip Elements in a Text

While the first experiment aimed to understand people's intuitive notion of gossip, the aim of the second experiment was to investigate whether the subjects could accept and apply a given definition by identifying the three obligatory elements of gossip according to Eggins and Slade (1997) (see also 7.2.1): third person focus, pejorative evaluation, and substantiating behaviour. In addition to the elements, we provided the definition of gossip presented in section 1: "evaluative talk about an absent third person".

The results from the first experiment indicated that conversations in which the person in focus was ambiguous received a lower gossip rating than those having an unambiguous third person focus. So an additional goal was to investigate whether changing the relationship between the participants in these examples and/or excluding information about their relationship status would affect the gossip rating.

7.7.1 Material

We used excerpts from *Seinfeld*⁵², *Desperate Housewives*⁵³, *Legally blonde*⁵⁴, and *Mean girls*⁵⁵. In total we selected 21 excerpts, of which 8 also occurred in the first experiment. Two of the recurring excerpts were used both in their original versions as well as in modified versions, in which we had removed information about the emotional connections between the participants. The purpose of this was to find out whether changing the interpersonal relationship would change the gossip rating.

⁵² Castle Rock Entertainment.

⁵³ Touchstone Television.

⁵⁴ Directed by Robert Luketic. Metro Goldwyn Mayer (2001).

⁵⁵ Directed by Mark Waters. Paramount Pictures (2004).

7.7.2 Procedure

The subjects were instructed to read the excerpts and then identify the gossip elements according to the following description:

- The person being talked about (third person focus) – the “target”, e.g. “Maisy Gibbons was arrested”
- Pejorative evaluation. A judgment of the target him-/herself or of the target’s behaviour. This evaluation is in most cases negative, e.g. “She’s a slut”, “He’s weird”
- The deviant behaviour that motivates the gossip and provides evidence for the judgment (also called the substantiating behaviour stage), e.g. “Maisy Gibbons was arrested”

For each element they found, they were asked to specify the corresponding line reference as given in the text. They were also instructed to say whether they considered the conversation to be gossip or not gossip. If their rating disagreed with the definition, i.e. if they had found all the elements but still rated the conversation as not gossip, or if one or more elements were lacking but the conversation was considered gossip anyway, they were asked to specify why.

7.7.3 Results

We analysed the results from the 19 subjects who completed ratings for all 21 excerpts. This gave a total of 399 yes/no judgments on 4 attributes. Inter-coder reliability is shown in Table 14. The easiest attribute to interpret is third person focus. All but three of the subjects marked either 4 or 5 excerpts as not having third person focus, with the remaining subjects not deviating by much (marking 3, 6, and 7 excerpts). Moreover, the subjects agree on which excerpts have third person focus: only one excerpt gets a substantial number of conflicting ratings (see the analysis given below in section 7.7.4), while the remaining 20 excerpts get consistent ratings from all subjects with only occasional deviation by one or two of the deviant subjects. This accounts for the high observed agreement on this feature (94.9%). Expected agreement is high because the corpus is not balanced (16 of 21 excerpts display third person focus), but even so, chance-corrected agreement is high (85.1%), showing that third person focus is an attribute that participants can readily and reliably identify.⁵⁶

The remaining attributes, including gossip, are less clear. Agreement on all of them is clearly above chance, but is not particularly high, showing that these

⁵⁶ To clarify, since third person focus is present in a majority of the examples the expected agreement of this category is high. In order to make the result comparable across studies, observed agreement has to be adjusted for chance agreement (See for example Artstein & Poesio, 2008)

notions are either not fully defined, or that the excerpts are ambiguous. Gossip itself is identified somewhat more reliably than either substantiating behaviour or pejorative evaluation; this casts doubt about the ability to use the latter two as defining features of gossip.

| | Alpha | Observed agreement | Expected agreement |
|--------------------------|-------|--------------------|--------------------|
| Gossip | 0.466 | 0.744 | 0.520 |
| Third person focus | 0.851 | 0.949 | 0.661 |
| Substantiating behaviour | 0.376 | 0.709 | 0.533 |
| Pejorative evaluation | 0.384 | 0.733 | 0.567 |

Table 14: Inter-coder reliability

To test the relationship between the various features, we looked for co-occurrences among the individual judgments. We have a total of 399 ratings (21 excerpts times 19 judges), each with 4 attributes; these are distributed as shown in Table 15. We can see that third person focus is an almost necessary condition for classifying a screenplay conversation as gossip, though it is by no means sufficient.

| | | 3rd person | | -3rd person | |
|---------|--------|------------|--------|-------------|--------|
| | | Subst | -Subst | Subst | -Subst |
| Gossip | Pejor | 168 | 24 | | 2 |
| | -Pejor | 33 | 14 | | |
| -Gossip | Pejor | 25 | 20 | 17 | 17 |
| | -Pejor | 6 | 23 | 3 | 47 |

Table 15: Relationship between the different elements and gossip.

Table 16-Table 18 show the co-occurrences of individual features to gossip; the association is strongest between gossip and third person focus and weakest between gossip and pejorative evaluation.

| | 3rd person | -3rd person |
|----------------|------------|-------------|
| Gossip | 239 | 2 |
| -Gossip | 74 | 84 |

Table 16: Gossip – third person focus

| | Substantiating behaviour | ¬Substantiating behaviour |
|----------------|-----------------------------|------------------------------|
| Gossip | 201 | 40 |
| ¬Gossip | 51 | 107 |

Table 17: Gossip – substantiating behaviour

| | Pejorative | ¬Pejorative |
|----------------|------------|-------------|
| Gossip | 194 | 47 |
| ¬Gossip | 79 | 79 |

Table 18: Gossip – pejorative evaluation

In addition to the co-occurrences of features on the individual judgments, we can look at these co-occurrences grouped by screenplay. Table 19 shows for each of the 21 excerpts, how many subjects identified each of the four features (the table is sorted by the gossip score). The same co-occurrences are illustrated in the chart below. It is apparent from the tables that all the features are correlated to some extent.

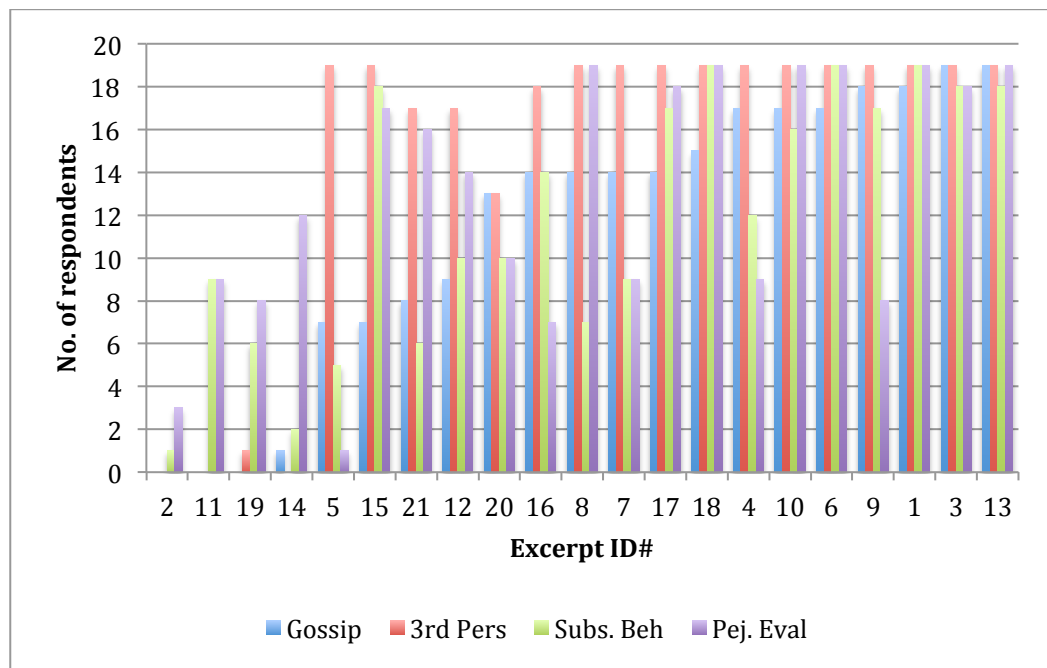


Figure 36: Chart illustrating co-occurrences grouped by excerpts

| ID⁵⁷ | Gossip | Third person | Subst. behaviour | Pejorative evaluation |
|------------------------|---------------|---------------------|-------------------------|------------------------------|
| 2 | 0 | 0 | 1 | 3 |
| 11 | 0 | 0 | 9 | 9 |
| 19 | 0 | 1 | 6 | 8 |
| 14 | 1 | 0 | 2 | 12 |
| 5 | 7 | 19 | 5 | 1 |
| 15 | 7 | 19 | 18 | 17 |
| 21 | 8 | 17 | 6 | 16 |
| 12 | 9 | 17 | 10 | 14 |
| 20 | 13 | 13 | 10 | 10 |
| 16 | 14 | 18 | 14 | 7 |
| 8 | 14 | 19 | 7 | 19 |
| 7 | 14 | 19 | 9 | 9 |
| 17 | 14 | 19 | 17 | 18 |
| 18 | 15 | 19 | 19 | 19 |
| 4 | 17 | 19 | 12 | 9 |
| 10 | 17 | 19 | 16 | 19 |
| 6 | 17 | 19 | 19 | 19 |
| 9 | 18 | 19 | 17 | 8 |
| 1 | 18 | 19 | 19 | 19 |
| 3 | 19 | 19 | 18 | 18 |
| 13 | 19 | 19 | 18 | 19 |

Table 19: Co-occurrences grouped by excerpts

Table 20 shows the correlation between gossip and each of the other three features. The first column calculates correlation based on the individual judgments (399 items, each score is either 0 or 1); the second column calculates correlation based on the rated excerpts (21 items, each score is an integer between 0 and 19, as in Table 19); and the third column groups the judgments by subject (19 items, each score is an integer between 0 and 21, indicating the number of dialogues in which the subject identified the particular feature; the full data are not shown).

⁵⁷ Presentation was ordered by ID, same for all subjects.

| Correlation with gossip | Pearson's r | | |
|--------------------------|-------------|----------|---------|
| | Individual | Excerpt | Subject |
| Third person | 0.622*** | 0.849*** | 0.503* |
| Substantiating behaviour | 0.518*** | 0.765*** | 0.625** |
| Pejorative evaluation | 0.321*** | 0.518* | 0.459* |

* $p < 0.05$ ** $p < 0.01$ *** $p < 0.001$

Table 20: Correlation between gossip and each of the three features

All the correlations are significantly different from 0 at the $p \leq 0.05$ level or greater. The differences between the columns are not significant, except for the difference between the third person correlation by individuals and that by excerpt, which is significant at $p \leq 0.05$. The correlations between the features on the individual judgments show that subjects tend to identify the different features together; this may be partly a reflection of awareness on their part that the features are expected to go together, given the task definition. The correlations between the excerpt scores show that the excerpts themselves differ along the four dimensions, and these differences go hand in hand. Finally, we see that the subjects themselves differ in how often they identify the different features, though the correlations are likely to be just a reflection of the first tendency identified above, to mark the features together.

7.7.4 Discussion

We wanted to find out whether the subjects would accept, understand and be able to apply a given definition. The results from the experiment showed that the subjects accepted the given definition to some extent and managed to apply it. When the subjects disagreed they were asked to say why. One of the subjects, for example, explicitly disagreed with the definition given in the introduction and provided a counter definition: "Gossip is idle talk or rumour, especially about the personal or private affairs of others". Yet another subject was uncertain about which definition to use: "Depends what you mean by gossip. It can either mean malicious, behind the back talk of other people or idle chat. If you mean 'idle chat' with gossip then this is also gossip". A possible explanation could be that the subjects refer to different forms of gossip (see for example Gilmore, 1978) and therefore apply different definitions (such as the lexical definition presented earlier) than the one that was given in the experiment.

Several subjects stated that they judged the conversation as gossip even if they did not identify any pejorative evaluation, and they also questioned whether the evaluation had to be pejorative or even present at all, or as one of the subjects put it: "Although there is no pejorative evaluation (at least not clearly) I believe this

is gossip". These subjects thus explicitly reject Eggins and Slade's (1997) requirement that the evaluation has to be pejorative.

The examples above show that people have variable intuitions of gossip and consequently the concept of gossip is somewhat vague. Even so, the experiment also showed that people to a large degree are in agreement when the examples according to the given definition clearly are gossip or not gossip. Meaning that even though the definition does not capture all types of (potential) gossip conversations, it captures those episodes that most people agree to be gossip, which for our purpose is sufficient.

7.8 Effect of Interpersonal Relations

In some particular cases, the subjects did not choose gossip even if all elements had been found. The results from the first experiment indicated that this deviation either was related to the interpersonal relationship between the gossip participants or that the focus was ambiguous. In order to test whether changing the interpersonal relationship between the participants would change the gossip rating, we compared the results from the conversations we had modified with their original counterparts. In one of the original excerpts, the addressee was romantically involved with the man that the speaker was talking about. The speaker formulated the negative assessment and deviant behaviour in a way that for most people would be interpreted as a warning, which probably explains why only 7 of the 19 subjects rated the original conversation as gossip. The modified version was, on the other hand, rated as gossip by all subjects.

In the second dialogue, the speaker questions the addressee's choice of person to date, and does this by both evaluating the person negatively as well as providing evidence for the evaluation. It turns out, however, that the addressee thinks she is going out for a date with someone else, so a large part of the conversation deals with trying to identify the target:

- A: Hey, by the way, did you ever call that guy from the health club?
B: Oh yeah! Jimmy.
A: Can't believe you're going out with him...
B: Why?
A: I dunno. He's so strange.
B: How so?
A: Did you notice he always refers to himself in the third person. Jimmy can dunk. Jimmy's new in town. Jimmy we'll see you later.
B: No No... That's not him. That's the guy who gave me Jimmy's number.
A: That's Jimmy. That's the way he talks.

B: I'm going to go see Mel Tormé with him?

Dialogue 14: Original excerpt

A: Hey, by the way, did Elaine ever call that guy from the health club?

B: Oh yeah! Jimmy.

A: Can't believe she's going out with him...

B: Why?

A: I dunno. He's so strange.

B: How so?

A: Did you notice he always refers to himself in the third person. Jimmy can dunk. Jimmy's new in town. Jimmy we'll see you later.

B: No No... That's not him. That's the guy who gave her Jimmy's number.

A: That's Jimmy. That's the way he talks.

B: She's going to go see Mel Tormé with him?

Dialogue 15: Modified excerpt

15 of 19 subjects rated the original conversation as gossip, while all subjects rated it as gossip in the modified version. These comparisons indicate that the status of the relationship between the gossipers and the gossip target affects whether the dialogue is considered gossip or not. In the original version of both these examples, the focus was ambiguous, i.e. the focus was as much on the addressee as on the absent third person.

We have shown that third person focus is a key element of gossip. The correlation was also confirmed by the subjects themselves in their comments, where the lack of third person often was listed as a reason for not choosing gossip. In the following example (Dialogue 16) the respondent regarded the conversation as gossip even if it really was an insult directed towards the addressee, but explained it as its “[...] almost like he's forgotten he's talking to the person he's giving this opinion/gossip about”:

E: I can't imagine doing all this and Donovan's internship next year

W: Elle, c'mon, there's no way you'll get the grades to qualify for one of those spots. You're not smart enough... I didn't mean...

E: Am I on glue, or did I not get into the same law school you did, Warner?

W: Well, yeah, but...

E: But what? We took the same LSAT, we take the same classes, ...

W: I just don't want to see you get your hopes up. You know how you get.

E: I'll never be good enough for you, will I?

Dialogue 16: Dialogue excerpt retrieved from the movie *Legally Blonde*.⁵⁸

⁵⁸ (Lutz & Smith, 2001)

The highest disagreement concerning third person focus was found in the following excerpt:

- K: Okay, what is it?
G: Regina says everyone hates you because you're such a slut.
K: She said that?
G: You didn't hear it from me.

Dialogue 17: Example of dialogue excerpt causing high disagreement

The dialogue contains an ambiguous focus in that it both includes a quote as well as a confrontational insult. By using the third person reference, Gretchen avoids taking responsibility for the insult. In some sense both Karen and Regina are in focus, where Karen is the target of the pejorative evaluation and Regina can be interpreted as being the focus of the substantiating behaviour component. How Regina's role is interpreted is determined by the respondents' personal attitude towards gossiping in general (i.e. whether they interpret Gretchen's utterance as containing an implicit evaluation of Regina's behaviour or not), and how they perceive the interpersonal relationship between Karen and Gretchen. Gossip has an inherent contradiction in that it both has a function of negotiating the accepted way to behave while it at the same time often is considered an inappropriate activity that can have serious negative consequences for both the gossipers as well as the gossip target (see e.g. Gilmore, 1978; Bergmann, 1993; Eggins and Slade, 1997; Hallett et al., 2009).

7.8.1 Conclusion

The aim of these studies has been to get a workable definition of gossip that people can agree upon and that is sufficiently precise to provide a basis for computational implementation.

We conducted two experiments to investigate people's intuitive notion of gossip and the results show that (1) conversations in which all elements are present, where no intimate relationships exist between the participants, and in which the person in focus is unambiguous, are very likely to be considered gossip. (2) Conversations that have at most one gossip element are not considered gossip. (3) Inconsistencies are mainly found in conversations that lack one or two elements or have at least one element that is ambiguous, or are taking place between gossipers that have an intimate relationship.

We have suggested that third person focus is a necessary, but not sufficient, element of gossip, but the other elements are less clear even if their co-occurrence in a conversation clearly affects the gossip score. In the second experiment this might be due to the instructions, but it does not explain the unbiased results from the first experiment. So we can clearly see that all three

elements are important for the understanding of gossip, even though the subjects' had trouble in identifying them.

7.9 Towards a Computational Model of Gossip

From the first attempt to model gossip in combination with the experiments conducted on gossip conversation we are now about to propose a tentative model for initiating gossip. The experiments show that the interpersonal relationship between the involved parties has a central role in determining whether a conversation can be regarded as gossip and whether it is appropriate to initiate a gossip conversation at all. Therefore, it seems reasonable to revisit the theory of face management presented in chapter 5.

7.9.1 Determining the Appropriateness for Initiating Gossip

Gossip has been described as containing “morally contaminated information”, which can damage the initiator's reputation (Bergmann, 1993). Because of this, the initiator must make sure that the recipient is willing to gossip (Bergmann, 1993) and that the relationship is sufficiently good to minimize the threat to face. In section 5.2 we introduced ρ for determining the *preconditions* for a certain type of conversation to take place. ρ is individually calculated by each participant and corresponds to the individual's perception of the status of the interpersonal relationship. In this section, ρ will be used by the NPC to determine whether it is appropriate to initiate a gossip conversation at all.

From previous studies and the experiments presented earlier, it is however clear that there is more to a gossip conversation than the perceived relationship between the participants. The following factors will therefore be considered in the model presented later in this section:

- The (perceived) relationship between the NPC and the player character (PC) (ρ)
- The relationship between each of the participants and the potential target
- The news value of the gossip story
- The sensitivity of the story content

In order to qualify as gossip, the story must have a news value (see for example Bergmann, 1993 and Crawford, 2004):

[...] the primary purpose of a Gossip system is to move information through the cast of Actors. Information is interesting only so long as some people know it and other people don't; after everyone knows it, gossip's dramatic value evaporates. (Crawford, 2004, p. 235)

A gossip story is assumed to contain the elements presented earlier: third person focus, a (pejorative) evaluation of the person in focus or of their behaviour, and a "substantiating behaviour" component. It is assumed that revealing the story is more or less sensitive for the gossip target, i.e. the factor "sensitivity" symbolizes a measure of how hurtful it would be for the gossip target if the information were spread. Also, information is typically spread exponentially and so its news value is reduced when the story is revealed.

Gossip is thus here understood as a conversation in which the participants reveal a sensitive gossip story with a certain news value about an absent person who unambiguously is in focus.

7.9.2 Initiating Gossip

Crawford (2004) presents a model of how a character can initiate gossip in a game. The decision to gossip is taken in two steps: In the first step the character chooses one of the characters in the surrounding that could be a potential gossip partner. In the second step, the character evaluates whether it is appropriate to gossip. The first decision is based on factors such as trust and affection, similar to our ρ value, while the other decision relies on the initiating character's loquaciousness, cf. the effects of the NPC's personality type as presented earlier in 5.3.2 and 5.3.3. In deciding what to talk about, Crawford suggests that the character should only reveal information that the other character is unaware of to avoid the pre-sequences typical for gossip initiations (see for example Bergmann, 1993). To get inside the mind of a scripted character is of course possible and his argument for doing so has a point, but in the model we will propose here, we will not consider that option. Instead, we will use a slightly different approach.

First, we evaluate the interpersonal relationship in accordance with the rules for social politeness as introduced in chapter 5. In 5.2.1 we proposed that the social distance (SD) can have one of the following values (with approximate correspondences): 0 for intimate relationships; 1 for friends; 2 for acquaintances; and 3 for strangers. The target is then selected on basis of the following factors assuming that there is an NPC (S) who is talking to the player character (PC) (H) about a third person (T):

- S perceives that the risk of losing face is low in the interaction with H, i.e., the social distance between S and H is (perceived to be) low and there is a (perceived) symmetric power relationship between them ($\rho < 3$).
- S has new, sensitive information about T.

- S knows T and believes that H knows, or is acquainted with, T too, i.e. $SD(S, T) < 3$ and $SD(H, T) < 3$.
- S does not have an intimate relationship with T, and believes that the same holds for H, i.e., $SD(S, T) > 0$ and $SD(H, T) > 0$.
- S believes that T cannot hear the conversation.

Next we need to determine the news value of the gossip story, which in our model is stored as a parameter, `NewsVal`, ranging from the value 0 (“common knowledge”) to 2 (“recently gained information”). The information is assigned a news value when it first is stored in the database, but as soon as it has been revealed, the news value decreases (for that character). We assume that the information is spread exponentially, i.e., that each person in possession of the information spreads it to at least two others up to a certain point in time when the information no longer is relevant or new.

Third, we evaluate how sensitive the information is in itself or to the character it concerns. If it is indifferent for the person in focus that the information is revealed or if the behaviour is generally acceptable within that culture (e.g. within the group, community, or society) it is unlikely that it will be regarded as gossip. In order to account for this, we have added a sensitivity value for the propositional content of the gossip story. Sensitivity is here specified to be an integer between 0 and 3, where 0 indicates a generally acceptable behaviour. The sensitivity is furthermore individually based and can only be an approximation (just like the evaluation of the social distance and the power). This is because a specific event or appraisal means different things for different people and the sensitivity can furthermore vary over time. A woman who is pregnant, for example, might want to keep it a secret until the risk of miscarriage has decreased. The sensitivity level can therefore be high initially, but decrease in time and become completely relaxed closer to giving birth. Hence, it is here assumed that the values of `sensitivity` and `NewsVal` decrease over time.

7.9.3 A Statechart Model for Initiating Gossip

Figure 37, below, presents a statechart specification for initiating gossip.

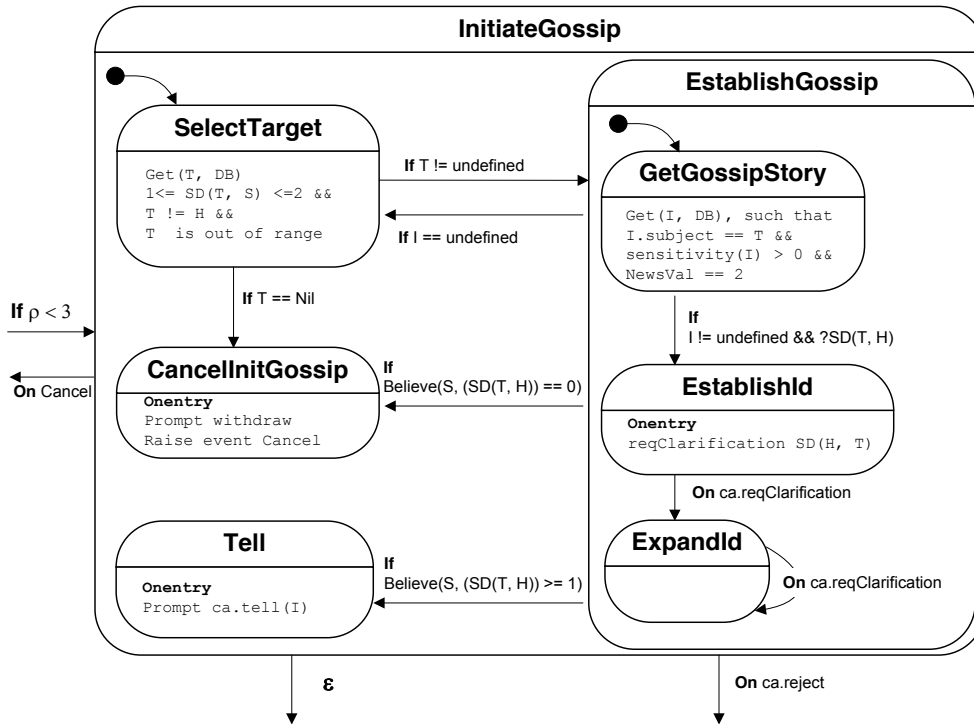


Figure 37 A statechart model for initiating gossip

The model works as follows: S and H are engaged in a conversation. If S and H are acquainted and have equal status, i.e. $\rho < 3$, a transition to the state `InitiateGossip` is triggered. The source state is here unspecified, but we can assume that the participants have greeted each other and perhaps small talked for a while before gossip is initiated.

S starts by searching for a potential gossip target (T) in the database (`Get(T, DB)`) according to the specification presented previously, which is performed on entry of the state `SelectTarget`. The story must not be about S him/herself or about H. If such a target exists in the database (DB), i.e., $T \neq \text{void}$, a transition from `SelectTarget` to `EstablishGossip` is activated. If there is no target that fulfils the initial criteria, the gossip is cancelled (never initiated).

The default start state in `EstablishGossip` is `GetGossipStory`, in which a search for a story about T is conducted. The search has two possible outcomes: there is a story about T such that $\text{NewsVal}=2$ and $\text{Sensitivity} > 0$, or it fails to find such a story.

If a story is found, the next step is to establish H and T's relationship. If S is uncertain of their relationship, a transition is taken to the state `EstablishId`, in which S requests a clarification that will help to establish the social distance between H and T, for instance as a question: "Do you know T?" or "Have you heard about T?". If H responds with a request for clarification of who T is, then S

can provide more information about T, which is handled in `ExpandId`. For example:

S: Do you know Lisa?
H: Lisa, who?
S: Lisa who works in the store with Jay.

Dialogue 18: Example dialogue for clarifying the identity of the target

If S believes that $SD(H, T)=0$, i.e., that they are intimately related, S will choose to back away from the gossip and the gossip is cancelled (which corresponds to a transition to `CancelInitGossip`). Otherwise, S will spread the gossip (which is performed in the state `Tell`). If no story exists that fulfils the criteria, S will attempt to find a new target.

This tentative model attempts to simulate the processes involved when a person intends to initiate a gossip conversation. To actually implement the model may be a tricky task, mainly because of the intricacy of the information that needs to be stored in the database. To have an anecdote of gossip nature may be one thing, but to be able to elaborate on the identity of the target, his or her relationship to the participants (including the speaker) as well as the ability to follow up on the gossip are tasks that require reasoning. This could be done by invoking a Prolog process or by allowing Prolog to be integrated in the SCXML data model as Radomski, Schnelle-Walka, and Radeck-Arneth (2013) suggest. A simpler solution would of course be to pick a gossip story in the database about an absent character, assume the other participants know him or her, and then ignore whether they are emotionally involved or not. This solution would however fail to account for the effect gossip may have on the interpersonal relationship.

7.9.4 Conclusion

One of the most important factors of gossip initiation is the status of the relationship between the gossipers and between them and the target. We therefore suggest that the following factors determine whether the NPC can introduce gossip at all: The (perceived) relationship between the NPC and the PC (ρ); the relationship between each of the participants and the potential target; the news value of the gossip story; and how sensitive the story is (culturally and personally). More specifically this means that the target must not be intimately related to any of the participants and that the participants must be friends or acquaintances. We have no restrictions concerning gossip between closely related participants, even if it is unclear whether it should to be considered gossip (see e.g. Bergmann, 1993). Such a restriction would be unnecessary since it just means that the risk of losing face is very low.

There are many different forms of gossip (see for example Gilmore, 1978) and many ways in which gossip can be initiated. In the model proposed here gossip has been limited to a conversation in which the speaker reveals a gossip story about an absent person who unambiguously is in focus. The gossip story may be more or less sensitive for the target and it has a news value that decreases when the gossip is shared.

The target is selected first (either by being mentioned in the previous discourse or by searching the database on entry of `SelectTarget`), but it could equally well be the story that is chosen first. There are a number of reasons why we chose the former alternative: First, even if it is the behaviour that is being evaluated, it is always a person that (at least) implicitly is being judged and thereby can be damaged by the gossip. Second, the target may already be in focus or mentioned (for instance in a pre-sequence, see Bergmann, 1993), as in the following example, where the actual gossip is initiated when Jerry expresses his opinion in line 3 (we have removed a sequence in which the participants try to establish the identity of the target):

1. Jerry: Hey, by the way, did you ever call that guy from the health club?
2. Elaine: Oh yeah! Jimmy.
- [...]
3. Jerry: Can't believe your going out with him...
4. Elaine: Why?
5. Jerry: I dunno. He's so strange.
- [...]

Dialogue 19: Dialogue excerpt from Seinfeld⁵⁹

Third, if the initiator misinterprets the target's relation to the addressee(s), it is the initiator that is considered to behave inappropriately. Hence, by making a mistake in the selection of the target the initiator faces the risk that the gossip backfires. The NPC will cancel the gossip if he suspects that the relationship between the target and the addressee is close.

7.10 Discussion

ECA's are created to simulate human behaviours and among the things humans do in order to socialize and share information is to gossip. Gossip is multifunctional and one of our most important means for building and maintaining social relationships. Through gossip we establish the accepted way to behave as members of a social group (see for example Eggins & Slade, 1997). In worlds populated with virtual humans, their ability to gossip can be a useful asset for a

⁵⁹ ©Castle Rock Entertainment, 1998.

human user in gaining information and establish relationships that affect future actions. An NPC capable of engaging in gossip conversations will be a resource for different types of useful information that otherwise would be difficult to access, such as cultural cues among members of a social group as well as tensions or conflicts between different group members.

In order to model gossip, we have to understand gossip – both how it is used as well as how it is produced. Gossip seems to work at different levels; the actual gossip conversation, the interpersonal relationship between the gossipers, the relationship each of them have to the target and finally the consequences of the gossip, i.e. the effects of the gossip, for instance changes in the relationships between the parties involved, including the target. Gossip can for instance get back at the gossiper himself; hence, we must determine the interpersonal relationships between those involved in the conversation (including the absent target) and how these relations are affected by the conversation. In order to do so, we need to both identify the dialogue moves typical for gossip as well as how these moves are distributed among the participants. Furthermore, we need to determine when it is appropriate for a character to initiate gossip, with whom, and why. We must furthermore be able to quantify the degree to which the gossip is harmful for the target. In the extension we also need to consider the implications such a dialogue will have for the progression of the game or scenario as a whole.

This chapter has presented a number of arguments for studying people's interpretation of human gossip, i.e. fabricated gossip for instance found in movie scripts, rather than naturally occurring human gossip. First, these conversations are fictional just like conversations in game worlds; second, they are limited to fit a particular scene, meaning that they are limited in time and scope; third, they are distilled, i.e. they do not contain sequences that are irrelevant and uninteresting from a gameplay perspective. It should however be said that this approach requires access to scripts (or transcriptions) in which gossip is likely to occur. In our case it has meant analysing a large number of scripts in search of gossip episodes that could be used in the experiments.

We initially defined gossip as *evaluative talk about an absent third person*, a definition that comprises the lowest common denominator of the definitions that we have found useful. We tested whether this definition was sufficient for creating a computational model of gossip and concluded that the three obligatory elements, third person focus, substantiating behaviour and (pejorative) evaluation, were not sufficient to define gossip, but that they clearly affected the gossip score. Only third person focus was a necessary, but insufficient element. Even though a gossip story contains all elements, it may differ on the account of how sensitive the story is to the participants and whether the information is new to them. In some situations revealing sensitive news might backfire, while spreading “old” stories might make the agent seem uninformed. For that reason

we added two parameters, sensitivity and news value, to the model for initiating gossip.

Previous studies (e.g. Eder & Enke, 1991; Eggins & Slade, 1997) have shown that gossip is a highly interactive genre that does not follow the typical turn-taking structure significant in most goal-directed conversations. These exchanges are usually built around adjacency pairs (Schegloff & Sacks, 1973), naturally assigning each participant either the sender role or the receiver role. Instead, any participant can add fuel to the gossip in terms of for example elaborations, evaluations, rhetorical questions, probes, refusals, agreements and disagreements. The participants can also back off at any time, for example if the gossip gets out of line.

We have presented a model of how an NPC may initiate and tell a gossip story, which means that the NPC is in control over the information that is being spread. To automatically detect gossip is however a much more difficult task. The NPC would for example need the ability to separate “Bill (Clinton) has had a sexual relationship with Monica (Lewinsky)” from “Bill (as in our common friend) has had a sexual relationship with Monica (someone we mutually know and that is not Bill’s wife)”. From a gossip point of view these two utterances differ vastly from each other. In the first example, the gossip is about famous persons neither of the participants (presumably) knows, while the latter example refers to acquaintances, who may be affected by the information exchange in some way or another. To gossip about famous people causes little threat to our own face, while we take social risks by talking about people we mutually know. This is an example of a situation where gossip can backfire. We would like our NPCs to have the ability to make this judgement in most situations, but occasional failures are acceptable, as they would only make the NPC more human-like.

We think that gossip can form an important mechanism for creating games based on social interaction between the player and the NPCs and it is not far-fetched to think that gossip could be used to increase the player’s emotional attachment to specific NPCs. Several gameplay design patterns may also be supported through gossip, e.g. INFORMATION PASSING, ACTIONS HAVE SOCIAL CONSEQUENCES, and RUMORS.

Chapter 8

Conclusions and Future Work

The purpose of this work has been to investigate and present approaches for dealing with socially oriented dialogues and believable behaviour applied to NPCs in games. It builds on theories of social politeness, gossip and activity based communication to explain and suggest ways to manage such dialogues. In statecharts we found a language to express these models, and by use of gameplay design patterns we found a language to express the actual and hypothetical gameplay mechanics associated with dialogues and other forms of PC-NPC interaction that exist in games. Another important aspect has been to present an approach that is accessible and usable to game developers. For this reason, it was an early decision to use standard technologies as they are free, accessible, stable, and provide consistency. Furthermore, it allows developers to use existing frameworks and off-the-shelf products for their application. Of particular interest was SCXML as it explicitly is meant to define the flow of an application, for example a game or a (game) dialogue system.

Much is left to do. The emotional impact is only touched upon here and the work on a gossip module is in its infancy, but we see an interesting opening here for creating socially believable game characters. By equipping them with strategies for how to behave in various situations we can create characters that may provide “the illusion of life” (Bates, 1994). We can design characters that are not cooperative in Grice’s sense as discussed in chapter 2.5.1, characters that can determine whether it is appropriate to gossip, that can adjust their social behaviour according to for example their mental state and the interpersonal relationship, and therefore are able to negotiate the price in a trading situation. We argue that these types of behaviours increase believability and motivate natural language dialogues in games. They can be used to create new types of games, perhaps focusing on social relationships, in which manipulations, gossip and lies are examples of strategies that could be used in order to advance in the game. Social skills are also interesting for virtual humans to make them believable.

In traditional drama, dialogue and language use are important means to present the characters, their personality, intentions, goals and thoughts. This information helps the audience to understand the motivation behind the actions and perhaps even foresee how the character will behave in certain situations. Games allow the audience to become immersed in the story by participating as one of the roles with the ability to affect how the story is presented and the direction it takes. The player is encouraged to interact with the environment, the characters and objects that exist there, and to decide where to go and what to do. This is a major difference between traditional storytelling media such as film and literature, where the reader is merely an observer, who is presented the narrative in a particular order determined by the author. A player is, on the other hand, given the freedom to choose a path and unfold one of possibly several story branches. The same applies to character development – in fixed stories the characters transform according to a predefined pattern, while in a game they may instead transform the story according to the player's actions. This thesis has presented several ways in which a character can act variably depending on the situation and current game state. Initially, an analysis was made to identify the different ways natural language dialogues may be used in games in order to determine the scope of this thesis, which resulted in the following classification: 1) For communicating with the game system; 2) for directing the playable character(s); 3) for letting NPCs communicate with each other; 4) for player-to-player chats; and 5) for letting NPCs engage in socially oriented conversations with the player (character). The last point was then selected as focus of the thesis.

Next, a classification of conversational agents was made in which NPCs were categorised as a type of ECA, but while most ordinary ECAs aim to mimic humans in realistic settings, NPCs may exhibit human-like behaviour in a fictional environment. Just like most ordinary ECAs, some NPCs may engage in practical dialogues and the requirements for building these NPCs are in many respects similar to those used for building cooperative agents. But for socially oriented NPCs, however, some of these requirements may need to be handled differently:

- To be believable, the NPC's dialogue system may need various and sometimes unconventional ways to handle errors. The NPC could for instance act as if it did not hear, pretend to understand, change the subject, or end the dialogue abruptly.
- The dialogue system should take into account the NPC's role, mental state, and goal in the game as well as the current situation including the interpersonal relationship, even if it means acting uncooperatively. This point may also be relevant for virtual humans.
- Socially oriented NPCs should be able to handle longer exchanges with varying topics.

This means that an NPC may talk too much or too little (or ignore the PC altogether), spread gossip and engage in other forms of casual conversations. Even lying and deception could be appropriate for an NPC under certain circumstances.

In order to identify relevant areas for improving game dialogues, an analysis of a number of prototypical games was conducted by means of gameplay design patterns (GPDs). The analysis resulted in a list of existing and hypothetical GDPs that are relevant for designing game characters in general and game dialogues in particular. Several of these patterns were thereafter implemented in the prototypes. It is assumed that all prototypes use FREE TEXT COMMUNICATION and require DIEGETICALLY CONSISTENT DIALOGUES. CHARACTER-SPECIFIC DIALOGUE is also used in all the prototypes – a pattern that is particularly noticeable in asymmetric relations, such as waiter-customer or doctor-patient, since the behaviour is dependent on the roles the characters have. In a gossip conversation this is less obvious due to the fact that gossip is mainly conducted between persons of (more or less) equal status that at least are acquainted with each other; so rather than having dialogue specific for a particular role it may be the gossipers' relationship to each other and to the person in focus that determine what becomes CHARACTER-SPECIFIC DIALOGUE. By means of parallel statecharts it becomes possible to synchronize the NPC's animations with its verbal behaviour and enable GAMEPLAY INTEGRATED DIALOGUES.

Chapter 1 presented a waiter character that interacts with a human user in natural language. In chapter 5, the waiter's dialogue manager was extended with the pattern SOCIAL NORM, i.e. the ability to behave according to a specific norm set for a particular situation (see section 3.7). Based on Brown and Levinson's (1987) algorithm for evaluating the threat to face of a certain action in a specific context, the parameter ρ was introduced as a measure of the individual's evaluation of the interpersonal relationship based on the distribution of power among the members and the social distance between them (corresponding to Brown and Levinson's P and D values, respectively). As such it differs from Brown and Levinson's algorithm as well as Walker and colleagues' (1996) approach since it does not explicitly take into account the evaluation of the imposition of the act (the *I* value). The reason why we chose not to explicitly consider *I* is that we found the parameter too vague to be useful for our purposes: *I* "is a culturally and situationally defined ranking of impositions by the degree to which they are considered to interfere with an agent's wants of self-determination or of approval" (Brown & Levinson, 1987, p. 77) as such it "involves a complex description" (*ibid.*, p. 77), meaning that *I* stands for different things in different contexts and may consist of a combination of different but interrelated values, which suggest that it is difficult to compute. Instead, ρ is used to determine the *precondition* for behaving in a certain way. The value of ρ can for example be

used as a condition for determining which transition to choose given a specific state and be combined with any parameter that suits the given context, similar to the I -value in Brown and Levinson's algorithm. Just like Brown and Levinson's (1987) algorithm, a low ρ indicates that the risk of losing face is low since the social distance between the participants as well as the distribution of power has been evaluated to be low or equal. A high ρ means that either the social distance is high and/or the power is perceived of as asymmetrical (in some situations the asymmetrical power is built-in). In the examples presented in Chapter 5, the parameter was used to determine which communicative act to perform in a particular situation and which linguistic style to use, and in Chapter 7 it was used for determining whether it is appropriate to initiate a gossip conversation in the current situation.

A shopkeeper with the ability to negotiate price was then introduced. As part of the DEAL project, the overarching goal was to create a conversational training partner for a language learner, also reported in (Wik, Hjalmarsson, & Brusk, 2007b; Hjalmarsson, Wik, & Brusk, 2007). In games, TRADE is a common gameplay pattern, allowing players to exchange goods for money in order to improve their possibilities to increase their experience points or in other ways advance in the game. NEGOTIATION is a sub-pattern, but is mostly used when trade is conducted between players. The NPC in the shopkeeper role calculates a counteroffer based on the last bid, the cost, and the number of turns left before the shopkeeper considers the negotiation to have broken down. The shopkeeper can also refuse to sell the item altogether if the customer's bid is considered an insult – for example lower than the cost or lower than the customer's previous bid. The customer is on the other hand expected to perform a similar calculation but instead of the cost, the customer assigns the object a certain value and the number of acceptable turns is most likely different from the shopkeeper's. This is of course information that is not shared between the participants. Instead, the outcome is unpredictable and highly dependent on the player's behaviour. Also, the number of acceptable turns on behalf of the shopkeeper can be randomized to make its behaviour even less predictable. The outcome is thus determined by the participants' ability to reach a point in the agreement space that is accepted by both agents. Sometimes the participants fail to reach an agreement and the negotiation breaks down. Negotiation was here reduced to a mathematical model for reaching this point in the agreement space; an interesting further step would be to introduce argument-based negotiation. Negotiations are also important within diplomacy, and could thus be useful in war-games for example.

In Chapter 7 it was suggested that there are situations in which an NPC should be able to engage in gossip conversations. In particular situations when it is important that the NPC behaves believably like a human agent. This is also applicable for virtual worlds populated with virtual humans. Various proposals of

gossip definitions were presented, but it soon became clear that a workable definition, that captures those dialogues people intuitively identify as gossip, does not exist. In an attempt to formulate such a definition, two experiments on gossip conversations were conducted. Based on previous research on gossip, three elements were proposed to be significant for gossip conversations; third person focus, an evaluation of the person in focus or of his or her behaviour, and some form of explanation or motivation substantiating the judgment. The aim was to investigate people's intuitive notion of gossip and the results showed that (1) conversations in which all gossip elements are present, where no intimate relationships exist between the participants, and in which the person in focus is unambiguous, are very likely to be considered gossip; (2) conversations that have at most one gossip element are not considered gossip; and (3) inconsistencies are mainly found in conversations that lack one or two elements or have at least one element that is ambiguous, or are taking place between gossipers that have an intimate relationship. Hence, it is clear that all three elements are important for the understanding of gossip, even though the subjects had trouble in identifying them. This suggests that we need to further investigate these elements to see how they can be specified more clearly. We have taken a first step toward a computational account of gossip, by empirically verifying the extent to which the given definition can be applied and the components recognized by people. Some of the next steps to further this program would be to include authoring content for believable characters that follow this definition, as well as attempting to automatically recognize these elements.

Several of the previous studies referenced in this thesis presented interesting findings about the nature of gossip. For example, like other casual conversations, gossip lacks a concrete goal; instead the goal is interpersonal in nature, for instance to negotiate the appropriate way to behave. It was also concluded earlier that the participants must make the judgement that the ρ value is low before entering a gossip conversation in order to avoid the gossip backfiring. A low ρ indicates that the participants are closely related and equal in power, which suggest that the roles are symmetrical. One of the participants is always the initiator and may as such have a dominant role in the conversation, at least initially, but this may change during the course of the interaction making the role distribution unclear. Any participant can contribute to the conversation in various ways; Eggins and Slade (1997) mention for example probes, enhancements and extensions concerning substantiating behaviour and pejorative evaluations, as well as rhetorical questions and opinions as possible contributions. Chapter 7 proposed a model for selecting a target and tell a gossip story. The next step is to create a computational model of gossip that also takes into account the other participants' contribution to the conversation. In order to do this it could be worth while to use machine learning techniques since it is unlikely that "the hand-

crafted set of rules does in fact optimally cover all possible aspects of user behaviour” (Schatzmann, Weilhammer, Stuttle, & Young, 2006, s. 103).

When gossip is possible in a virtual world, i.e. a world populated by a large number of other characters, the consequences of spreading the gossip may be considered in the larger context and not only among the participants engaged in the actual conversation. These consequences should also be further investigated.

It has not been the goal of this thesis to argue that natural language dialogue should replace these other forms of dialogue systems in games. Rather, the application areas mentioned here suggest that NL dialogues may be considered when designing specific situations where dialogue has a limited and well-defined purpose, social or functional. This is also similar to how typical ECAs operate. A well-defined domain limits the reasonable scope for the dialogue and ensures that the dialogue, even though apparently unlimited, becomes manageable.

8.1.1 SCXML

This thesis has presented a large number of reasons for why statecharts and SCXML should be a good fit for the gaming industry. Statecharts allow us to solve small problems in isolation that combined can solve bigger problems. We would like to summarize our arguments as follows: At its core, SCXML has FSMs – and it has been shown, over and over again, that FSMs are useful for describing the flow of a game, i.e. the pace and sequence of its states and events, and the range of choices in its progression. As a result of its Harel Statechart heritage, SCXML also supports hierarchy and concurrency, and thereby avoids the most pressing problem with ordinary FSMs – the notorious state explosion problem. The presence of hierarchy furthermore allows the developer to describe game flow at different levels of granularity, and to apply the methodological principles of top-down refinement and bottom-up clustering. In addition, the fact that SCXML is closely aligned to statechart theory and UML2 will help those using model driven development methodologies. It is possible to use a top-down approach (so called refinement) as well as bottom-up approach (clustering) in the design process and tasks may be delegated to different team members. Moreover, SCXML is at the final stage of becoming a web standard, which means that it is free to use, accessible, stable, distributable, and provides consistency. It is XML-based, and most game engines already handle XML data. The fact that SCXML is endorsed by the W3C may translate to better support in tooling, number of implementations and various runtime environments.

In particular SCXML and VoiceXML forms a powerful combination, where SCXML is used for specifying and implementing the flow aspect of a dialogue system, and VoiceXML supplies the voice widgets required. This enables an approach to the development of natural language enabled games where natural

language dialogue flow is seen as just an aspect of the overall game flow, and where SCXML is used for specifying and implementing (the major parts) of both kinds of flow. By moving dialogue control to SCXML we have been able to design different strategies for concluding a deal as well as for managing face. In fact, specifying dialogue flow by using statecharts turned out to be so intuitive that we decided to move all control to SCXML – an approach that is in line with how the DFP is supposed to work.

References

Abrahams, R. D. (1970). A Performance-Centered Approach to Gossip. *Man*, 5 (2), 290-301.

Adams, E. (2010). *Fundamentals of Game Design* (2. ed. ed.). Berkeley, Calif., USA: New Riders.

Adams, E., & Rollings, A. (2007). *Fundamentals of Game Design*. Upper Saddle River, New Jersey, USA: Pearson Prentice Hall.

Alewine, N., Ruback, H., & Deligne, S. (2004). Pervasive Speech Recognition. *IEEE Pervasive Computing*, 2 (4), 78-81.

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. New York, USA: Oxford University Press.

Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. (2001). Towards conversational human-computer interaction. *AI Magazine*, 22 (4), 27-37.

Allen, J. F., Guinn, C. I., & Horvitz, E. (1999). Mixed Initiative Interaction. *IEEE Intelligent Systems* , 14 (5), 14-23.

Allwood, J. (1995). *An Activity Based Approach to Pragmatics*. University of Gothenburg, Dept. of Linguistics. Gothenburg: Gothenburg Papers in Theoretical Linguistics 76.

Allwood, J. (1993). Language, Communication and Social Activity - Towards and Analysis of The Linguistic Communicative Aspects of Social Activities. In K. Junefelt (Ed.), *Proceedings of the XIVth Scandinavian Conference of Linguistics and the VIIIth Conference of Nordic and General Linguistics: Special Session on Activity Theory*. Gothenburg: University of Gothenburg.

Allwood, J. (1976). *Linguistic Communication as Action and Cooperation*. University of Gothenburg, Dept. of Linguistics. Gothenburg: Gothenburg Monographs in Linguistics 2.

Allwood, J., Traum, D., & Jokinen, K. (2000). Cooperation, dialogue and ethics. *International Journal Human-Computer Studies*, 53, 871-914.

Artstein, R., & Poesio, M. (2008). Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* , 34 (4), 555--596.

Aust, H., Oerder, M., Seide, F., & Steinbiss, V. (1995). The Philips automatic train timetable information system. *Speech Communication*, 17 (3-4), 249-262.

- Austin, J. L. (1962). *How to Do Things With Words* (2nd edition ed.). Cambridge, Massachusetts, USA: Harvard University Press.
- Barnett, J., Akolkar, R., Auburn, R., Bodell, M., Burnett, D. C., Carter, J., et al. (2012, February 16). *State Chart XML (SCXML): State Machine Notation for Control Abstraction*. From W3C Working Draft: <http://www.w3.org/TR/scxml/>
- Bartle, R. A. (2003). *Designing Virtual Worlds*. Indianapolis New Riders: New Riders.
- Bartle, R. A. (1996). Players Who Suit MUDs. *Journal of MUD research*, 1 (1).
- Bateman, C. (2007). Dialogue Engines. In C. Bateman (Ed.), *Game Writing: Narrative Skills for Videogames* (pp. 267-292). Boston, MA, USA: Charles River Media.
- Bates, J. (1994). The Role of Emotion in Believable Agents. *Communications of the ACM*, 37 (7), 122-125.
- Bergmann, J. R. (1993). *Discreet Indiscretions: The Social Organization of Gossip*. (J. J. Bednartz, Trans.) New York: Aldine de Gruyter.
- Bernsen, N. O., & Dybkjær, L. (2005). Meet Hans Christian Andersen. *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*. Lisbon, Portugal: ISCA Archive.
- Bernsen, N. O., Dybkjær, H., & Dybkjær, L. (1996). Co-operativity in Human-Machine and Human-Human Spoken Dialogue. *Discourse Processes*, 21 (2), 213-236.
- Bethesda Game Studios. (2002). *The Elder Scrolls III: Morrowind* [Computer game]. Ubisoft.
- Bethesda Game Studios. (2006). *The Elder Scrolls IV: Oblivion* [Computer game]. Bethesda Softworks LLC .
- Bethesda Game Studios. (2011). *The Elder Scrolls V: Skyrim* [Computer game]. Bethesda Softworks LLC.
- Bethke, E. (2003). *Game Development and Production*. Plano, Texas, USA: Wordware Publishing.
- Bickmore, T., & Cassell, J. (2000). "How about this weather?" Social Dialogue with Embodied Conversational Agents. *Proceedings of the American Association for Artificial Intelligence (AAAI) Fall Symposium on Narrative Intelligence*, (pp. 4-8). Cape Cod, MA.
- Bickmore, T., & Cassell, J. (2001). Relational Agents: A Model and Implementation of Building User Trust. *Proceedings of the SIBRUGCHI conference on Human factors in computing systems* (pp. 396 - 403). New York, USA: ACM.

- Bioware. (2011). *Dragon Age II* [Computer game]. Electronic Arts.
- BioWare. (2010). *Mass Effect 2* [Computer game]. Electronic Arts, Inc.
- BioWare. (2008). *Mass Effect* [Computer game]. Microsoft Game Studios.
- BioWare. (2012). *Mass Effect 3* [Computer game]. Electronic Arts.
- Björk, S., & Holopainen, J. (2003a). Describing Games: An Interaction-Centric Structural Framework. In M. Copier, & J. Raessens (Ed.), *Level Up Conference Proceedings: Proceedings of the 2003 Digital Games Research Association Conference*. Utrecht: University of Utrecht.
- Björk, S., & Holopainen, J. (2005). *Patterns in Game Design*. Hingham, Massachusetts: Charles River Media.
- Björk, S., Lundgren, S., & Holopainen, J. (2003b). Game Design Patterns. In M. Copier, & J. Raessens (Ed.), *Level Up Conference Proceedings: Proceedings of the 2003 Digital Games Research Association Conference*. Utrecht: University of Utrecht.
- Black Isle Studios. (1997-). *Fallout* [Computer game]. Interplay Productions Inc.
- Black Isle Studios. (1999). *Planescape: Torment* [Computer game]. Interplay Entertainment Corp.
- Blizzard Entertainment. (2005). *World of Warcraft* [Computer game]. Blizzard Entertainment.
- Boffo Games, Inc. (1997). *The Space Bar* [Computer game]. Bomico Entertainment Software GmbH.
- Bos, J., Klein, E., Lemon, O., & Oka, T. (2003). DIPPER: Description and Formalisation of an Information-State Update Dialogue System Architecture. *4th SIGdial Workshop on Discourse and Dialogue* (pp. 115–124). Sapporo, Japan: ACL.
- Boye, J., & Gustafson, J. (2005). How to do dialogue in a fairy-tale world. *Proceedings of SigDial: 6th Workshop on Discourse and Dialogue*, (pp. 242-246). Lisbon.
- Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. Cambridge, Massachusetts, USA: Harvard University Press.
- Brown, P., & Levinson, S. C. (1987). *Politeness: Some universals in language usage*. Cambridge: Cambridge University Press.
- Brusk, J. (2010). A Computational Model for Gossip Initiation. In P. Lupkowski, & M. Purver (Ed.), *Aspects of Semantics and Pragmatics of Dialogue. SemDial 2010, 14th Workshop on the Semantics and Pragmatics of Dialogue* (pp. 139-142). Poznan: Polish Society for Cognitive Science.

- Brusk, J. (2008). Dialogue Management for Social Game Characters Using Statecharts. *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, (pp. 219-222). Yokohama, Japan.
- Brusk, J. (2009). Using Screenplays as Corpus for Modeling Gossip in Game Dialogues. In J. Edlund, J. Gustafson, A. Hjalmarsson, & G. Skantze (Ed.), *Proceedings of DialHolmia: 2009 Workshop on the Semantics and Pragmatics of Dialogue* (pp. 139-140). Stockholm: Royal Institute of Technology.
- Brusk, J., & Björk, S. (2009). Gameplay Design Patterns for Game Dialogues. *Proceedings of DiGRA 2009: Breaking New Ground: Innovation in Games, Play Practice and Theory*. West London, UK: Brunel University.
- Brusk, J., & Eladhari, M. (2006). Playing the Character. *Seminar on Playing Roles*. Tampere, Finland: Tampere University.
- Brusk, J., & Lager, T. (2008). Developing Natural Language Enabled Games in SCXML. *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, 12 (2), 156-163.
- Brusk, J., Artstein, R., & Traum, D. (2010). Don't tell anyone! Two Experiments on Gossip Conversations. *Proceedings of the SIGDIAL 2010 Conference* (pp. 193-200). Association for Computational Linguistics.
- Brusk, J., Lager, T., Hjalmarsson, A., & Wik, P. (2007). DEAL – Dialogue Management in SCXML for Believable Game Characters. *Proceedings of ACM Future Play* (pp. 137-144). Toronto, Canada: ACM.
- Bunt, H. (1994). Context and Dialogue Control. *Think*, 3:19-33 (1994). *Think Quarterly*, 3, 19-33.
- Burbeck, S. (1992). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). *Urbana-Champaign (UIUC) Smalltalk Archive*, 2nd. University of Illinois.
- Carlsson, R., & Granström, B. (1996). The Waxholm Spoken Dialogue System. (Z. Palková, Ed.) *Phonetica Pragensia IX. Charisteria viro doctissimo Premysl Janota oblata, I*, 39-52.
- Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H., et al. (1999). Embodiment in Conversational Interfaces. *Proceedings of the CHI'99 Conference*, (pp. 520-527). Pittsburgh, PA.
- Cassell, J., Bickmore, T., Campbell, L., Vilhjálmsson, H., & Yan, H. (2000a). Human Conversation as a System Framework. In J. Cassell, J. Sullivan, S. Prevost, & E. Churchill (Eds.), *Embodied Conversational Agents* (pp. 29-63). Cambridge, MA: MIT Press.
- Cassell, J., Sullivan, J., Prevost, S., & Churchill, E. (Eds.). (2000b). *Embodied Conversational Agents*. MIT Press.

Cavazza, M., & Charles, F. (2005). Dialogue Generation in Character-based Interactive Storytelling. *Proceedings of AAAI First Annual Artificial Intelligence and Interactive Digital Entertainment Conference*. Marina del Rey, California, USA.

Cavazza, M., Martin, O., Charles, F., Mead, S. J., & Marichal, X. (2003). Users Acting in Mixed Reality Interactive Storytelling. *Proceedings of the 2nd International Conference on Virtual Storytelling*, (pp. 189-197).

Cermak-Sassenrath, D. (2012). Experiences with design patterns for oldschool action games. In *Proceedings of The 8th Australasian Conference on Interactive Entertainment: Playing the System (IE '12)* (p. Article no. 14). New York, NY, USA: ACM.

Chatman, S. (1978). *Story and Discourse: Narrative Structure in Fiction and Film*. Ithaca and London: Cornell University Press.

Chu-Carroll, J. (2000). MIMIC: An Adaptive Mixed Initiative Spoken Dialogue System for Information Queries. *Proceedings of the sixth Conference on Applied Natural Language Processing* (pp. 97-104). Morristown, USA: ACM.

Church, D. (1999). *Formal Abstract Design Tools*. Retrieved 06 11, 2010 from Gamasutra:
http://www.gamasutra.com/view/feature/3357/formal_abstract_design_tools.php

Clark, A., Fox, C., & Lappin, S. (Eds.). (2010). *The Handbook of Computational Linguistics and Natural Language Processing*. West Sussex, UK: Wiley-Blackwell.

Clark, H. H. (1996). *Using Language*. Cambridge: Cambridge University Press.

Clark, H. H., & Brennan, S. E. (1991). Grounding in Communication. In L. B. Resnick, J. M. Levine, & S. D. Teasley (Eds.), *Perspectives on Socially Shared Cognition*. Washington D.C.: American Psychological Association.

Cooper, R., & Larsson, S. (1998). Dialogue Moves and Information States. *Gothenburg Papers in Computational Linguistics*. Gothenburg, Sweden: University of Gothenburg.

Corkrey, R., & Parkinson, L. (2002). Interview voice response: Review of studies 1989-2000. *Behavior Research Methods, Instruments, & Computers*, 34 (3), 342-353.

Crawford, C. (2004). *On Interactive Storytelling*. Indianapolis, Ind.: New Riders.

Cristiano, G. (2008). *The Storyboard Design Course: The Ultimate Guide for Artists, Directors, Producers and Scriptwriters*. Thames & Hudson Ltd.

Cyan, Inc. (1993). *Myst* [Computer game]. Brøderbund Software, Inc.

EA Sports. (1993-2012). *FIFA* [Computer game]. Electronic Arts.

- Eder, D., & Enke, J. L. (1991). The Structure of Gossip: Opportunities and Constraints on Collective Expression among Adolescents. *American Sociological Review* , 56 (4), 494-508.
- Edlund, J., Skantze, G., & Carlson, R. (2004). Higgins - a Spoken Dialogue System for Investigating Error Handling Techniques. *Proceedings of INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing*, (pp. 229-231). Jeju Island, Korea: ISCA Archive.
- Egges, A., Zhang, X., Kshirsagar, S., & Magnenat-Thalmann, N. (2003). Emotional Communication with Virtual Humans. *Proceedings of the 9th International Conference on Multi-Media Modeling* .
- Eggin, S., & Slade, D. (1997). *Analysing Casual Conversation*. London, UK: Equinox Publishing Ltd.
- Eladhari, M. P. (2009). *Characterising action potential in virtual game worlds applied with the mind module*. Gotland University, Department of Game Design, Narrative and Time-based Media. Middlesbrough: University of Teesside.
- Electronic Arts. (1998-). The Sims(tm). Electronic Arts.
- Fey, T. (Writer), & Waters, M. (Director). (2004). *Mean Girls* [Motion Picture].
- Field, S. (2005). *Screenplay : The Foundations of Screenwriting*. Westminster, MD, USA: Dell Publishing.
- Fine, G. A., & Rosnow, R. L. (1978). Gossip, Gossipers, Gossiping. *Personality and Social Psychology Bulletin* , 4 (1), 161-168.
- Fu, D., & Houlette, R. (2002). Putting AI in entertainment: an AI authoring tool for simulation and games. *Intelligent Systems, IEEE* , 17 (4), 81-84.
- Fu, D., & Houlette, R. (2004). The Ultimate Guide to FSMs in Games. In S. Rabin (Ed.), *AI Game Programming Wisdom 2*. Hingham, Massachusetts, USA: Charles River Media, Inc.
- Fullerton, T. (2008). *Game Design Workshop* (2nd ed.). Burlington, MA, USA: Elsevier Inc.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series.
- Gandhe, S., DeVault, D., Roque, A., Martinovski, B., Artstein, R., Leuski, A., et al. (2008). From Domain Specification to Virtual Humans: An Integrated Approach to Authoring Tactical Questioning Characters. *Proceedings of INTERSPEECH 2008 - 9th Annual Conference of the International Speech Communication Association* (pp. 2486-2489). Brisbane, Australia: ISCA Archive.

- Gebhard, P., Kipp, M., Klesen, M., & Rist, T. (2003). Authoring Scenes for Adaptive, Interactive Performances. *AAMAS'03: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 725-732). New York, USA: ACM.
- Gee, J. P. (2003). *What Video Games Have to Teach us About Learning and Literacy*. New York: Palgrave MacMillan .
- Gilmore, D. (1978). Varieties of Gossip in a Spanish Rural Community. *Ethnology* , 17 (1), 89-99.
- Ginzburg, J. (1996). Dynamics and the Semantics of Dialogue. In J. Seligman (Ed.), *Language, Logic and Computation* (Vol. 1). Stanford: CSLI Lecture Notes, CSLI.
- Ginzburg, J., & Fernández, R. (2010). Computational Models of Dialogue. In A. Clark, C. Fox, & S. Lappin (Eds.), *The Handbook of Computational Linguistics and Natural Language Processing*. West Sussex, UK: Wiley-Blackwell.
- Gluckman, M. (1963). Papers in Honor of Melville J. Herskovits: Gossip and Scandal. *Current Anthropology* , 4 (3), 307-316.
- Goffman, E. (1967). On Face-Work: An Analysis of Ritual Elements in Social Interaction. In E. Goffman, *Essays on Face-to-Face Behavior*. New York, USA: Pantheon Books.
- Goodwin, M. H. (1980). He-Said-She-Said: Formal Cultural Procedures for the Construction of a Gossip Dispute Activity. *American Ethnologist* , 7 (4), 674-695.
- Grice, H. P. (1975). Logic and Conversation. (P. Cole, & J. L. Morgan, Eds.) *Syntax and Semantics: Speech Acts* , 3, 41-58.
- Griol, D., Callejas, Z., & López-Cózar, R. (2010). Statistical Dialog Management Methodologies for Real Applications. *Proceedings of the SIGDIAL 2010 Conference* (pp. 269-272). Association for Computational Linguistics.
- Gustafson, J., Boye, J., Fredriksson, M., Johanneson, L., & Königsmann, J. (2005). Providing computer game characters with conversational abilities. *Proceedings of Intelligent Virtual Agents (IVA05)* (pp. 37-51). Kos, Greece: Lecture Notes in Computer Science, Springer Berlin / Heidelberg.
- Gustafson, J., Lindberg, N., & Lundeberg, M. (1999). The August Spoken Dialogue System. *Proceedings of Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99)*. Budapest, Hungary: ISCA Archive.
- Hallett, T., Harget, B., & Eder, D. (2009). Gossip at Work: Unsanctioned Evaluative Talk in Formal School Meetings. *Journal of Contemporary Ethnography* , 38 (5), 584-618.

- Halliday, M. A. (1978). *Language as a Social Semiotics: The Social Interpretation of Language and Meaning*. Baltimore: University Park Press.
- Hammersley, T. (2009, June 24). *Planning For Fun In Game Programming - Part 2*. Retrieved 06 15, 2011 from Gamasutra: http://www.gamasutra.com/view/feature/4060/planning_for_fun_in_game_.php
- Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems . *Science of Computer Programming* , 8 (3), 231-274.
- Hayes-Roth, B., & Doyle, P. (1998). Animate Characters. *Autonomous Agents and Multi-Agent Systems* , 1 (2), 195-230.
- Hjalmarsson, A., Wik, P., & Bruski, J. (2007). Dealing with DEAL: A Dialogue System for Conversation Training. In S. Keizer, H. Bunt, & T. Paek (Ed.), *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*. Antwerp, Belgium: Association for Computational Linguistics.
- Horrocks, I. (1999). *Constructing the User Interface with Statecharts*. Boston, MA, USA: Addison Wesley Longman Publishing Co., Inc. .
- Horvath, B. M., & Eggins, S. (1995). Opinion Texts in Conversation. In P. H. Fries, & M. Gregory (Eds.), *Discourse in Society: Systemic Functional Perspectives* (pp. 29-46). Norwood, NJ, USA: Ablex.
- Hullett, K., & Whitehead, J. (2010). Design patterns in FPS levels. *FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games* (pp. 78-85). New York, NY, USA: ACM.
- Hunicke, R., LeBlanc, M., & Zubek, R. (2004). MDA: A Formal Approach to Game Design and Game Research. *Challenges in Game AI Workshop, AAAI-04*. AAAI Technical Report WS-04-04.
- Infocom, Inc. (1984). *The Hitchhiker's Guide to the Galaxy* [Computer game]. Infocom, Inc.
- Infocom, Inc. (1980). *Zork I: The Great Underground Empire* [Computer game]. Infocom, Inc.
- Isbister, K. (2006). *Better game characters by design: A psychological approach*. San Francisco: Morgan Kaufmann.
- Ishiguro, H., & Nishio, S. (2007). Building artificial humans for understanding humans. *Journal of Artificial Organs* , 10 (3), 133-142.
- Ippa, N. V., & Borst, T. (2007). *Story and Simulations for Serious Games: Tales from the Trenches*. Burlington, MA, USA: Focal Press.
- Järvinen, A. (2008). *Games without Frontiers: Theories and Methods for Game Studies and Design* . Tampere: Tampere University Press.

Jan, D., & Traum, D. (2005). Dialog Simulation for Background Characters. *Intelligent Virtual Agents. 3661/2005*, pp. 65-74. Lecture Notes in Computer Science.

Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C., & Wooldridge, M. (2001). Automated Negotiation: Prospects, Methods and Challenges. *Int. Journal of Group Decision and Negotiation* , 10 (2), 199-215.

Jokinen, K. (2009). *Constructive Dialogue Modelling*. Chichester, West Sussex, UK: John Wiley & Sons, Ltd.

Jokinen, K., & Hurtig, T. (2006). User Expectations and Real Experience on a Multimodal Interactive System. *Proceedings of Interspeech-2006*. Pittsburgh, US.

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, New Jersey, USA: Prentice-Hall, Inc. .

Juul, J. (2005). *Half-real: Video Games between Real Rules and Fictional Worlds*. Cambridge, Massachusetts, USA: The MIT Press.

Kempen, G., & Hoenkamp, E. (1987). An Incremental Procedural Grammar for Sentence Formulation. *Cognitive Science* , 11, 201-258.

Kempen, G., & Hoenkamp, E. (1982). Incremental sentence generation: Implications for the structure of a syntactic processor. In J. Horeck (Ed.), *Proceedings of the Ninth International Conference on Computational Linguistics (Coling '82)*. Prague: North-Holland Publishing Company.

Kenny, P., Parsons, T., Gratch, J., Rizzo, A., & Leuski, A. (2007). Virtual Patients for Clinical Therapist Skills Training. *7th International Conference on Intelligent Virtual Agents* . Paris, France.

Kreimeier, B. (2003). *Game Design Methods: A 2003 Survey*. Retrieved 06 11, 2010 from Gamasutra:
http://www.gamasutra.com/features/20030303/kreimeier_01.shtml

Kreimeier, B. (2002, Mar 13). *The Case for Game Design Patterns*. Retrieved 02 15, 2011 from Gamasutra:
http://www.gamasutra.com/view/feature/4261/the_case_for_game_design_patterns.php

Krippendorff, K. (1980). *Content Analysis: An Introduction to Its Methodology*. Beverly Hills, CA: Sage.

Kronlid, F. (2008). *Steps towards Multi-Party Dialogue Management*. PhD Thesis, University of Gothenburg, Department of linguistics.

- Kronlid, F., & Lager, T. (2007). Implementing the Information-State Update Approach to Dialogue Management in a Slightly Extended SCXML. In R. Artstein, & L. Vieu (Ed.), *Proceedings of the 11th International Workshop on the Semantics and Pragmatics of Dialogue (DECALOG)*, (pp. 99-106). Trento, Italy.
- Kölzer, A. (1999). Universal Dialogue Specification for Conversational Systems. *Linköping Electronic Articles in Computer and Information Science*, 4 (028).
- Lankoski, P. (2010). *Character-Driven Game Design: A Design Approach and Its Foundations in Character Engagement*. PhD Thesis, School of Art and Design, Aalto University.
- Lankoski, P., & Björk, S. (2007a). Gameplay Design Patterns for Non-Player Characters. In A. Baba (Ed.), *Situated Play: Proceedings of the 2007 Digital Games Research Association Conference* (pp. 416-423). Tokyo, Japan: The University of Tokyo.
- Lankoski, P., & Björk, S. (2007b). Gameplay Design Patterns for Social Networks and Conflicts. *Proceedings of Game Design and Technology Workshop*. Liverpool: John Moores University.
- Larsson, S. (2005). Dialogue Systems: Simulations or Interfaces? In C. Gardent, & B. Gaiffe (Ed.), *Proceedings of DIALOR'05: The Ninth Workshop on the Semantics and Pragmatics of Dialogue (SemDial)*. Nancy, France.
- Larsson, S., & Traum, D. (2000). Information State and Dialogue Management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6 (3 & 4), 323-340.
- Larsson, S., Jönsson, A., & Santamarta, L. (2000). Using the Process of Distilling Dialogues to Understand Dialogue Systems. *Proceedings of ICSLP 2000*, (pp. 374-377). Beijing, China.
- Lee, C., Jung, S., Kim, K., Lee, D., & Lee, G. G. (2010). Recent Approaches to Dialog Management for Spoken Dialog Systems. *Journal of Computing Science and Engineering*, 4 (1), 1-22.
- Levinson, S. C. (1983). *Pragmatics*. Cambridge: Cambridge University Press.
- Linden Lab. (2003). (Linden Lab) Retrieved 09 03, 2012 from Second Life: <http://secondlife.com>
- Lionhead Studios. (2001). Black & White [Computer game]. EA Games.
- Lionhead Studios. (2004). Fable [Computer game]. Microsoft Game Studios.
- LucasArts. (1998a). Grim Fandango [Computer game]. LucasArts.
- LucasArts. (1998b). Grim Fandango: Manual. LucasArts.
- Lucasfilm Games. (1987). Maniac Mansion [Computer game]. Lucasfilm Games.

- Lucasfilm Games. (1990). *The Secret of Monkey Island* [Computer game]. Lucasfilm Games.
- Lundgren, S., & Björk, S. (2003). Game Mechanics: Describing Computer-Augmented Games in Terms of Interaction . *Proceeding of Technologies for Interactive Digital Storytelling and Entertainment (TIDSE)*. . Darmstadt, Germany .
- Lutz, K. M., Smith, K. (Writers), & Luketic, R. (Director). (2001). *Legally Blonde* [Motion Picture].
- Marsella, S., Gratch, J., & Petta, P. (2010). Computational Models of Emotion. In K. R. Scherer, T. Bänziger, & E. Roesch (Eds.), *A Blueprint for Affective Computing: A sourcebook*. Oxford, UK: Oxford University Press.
- Mateas, M. (2001). A Preliminary Poetics for Interactive Drama and Games . *Digital Creativity* , 12 (3), 140-152.
- Mateas, M., & Stern, A. (2004). A Behavior Language: Joint Action and Behavioral Idioms. In H. Prendinger, & M. Ishizuka (Eds.), *Life-like Characters. Tools, Affective Functions and Applications*. Springer.
- Mateas, M., & Stern, A. (2005). Façade. <http://www.interactivestory.net/>.
- Mateas, M., & Stern, A. (2003). Façade: An experiment in building a fully-realized interactive drama. *Game Developers Conference: Game Design Track*.
- McCrae, R. R., & Costa, P. T. (1999). A Five-factor Theory of Personality. In L. A. Pervin, & O. P. John (Eds.), *Handbook of Personality* (2nd Edition ed., pp. 139-153). New York, USA: The Guilford Press.
- McGlashan, S., Burnett, D. C., Akolkar, R., Auburn, R., Baggia , P., Barnett, J., et al. (Eds.). (2010, December 16). *Voice Extensible Markup Language (VoiceXML) 3.0 (Working draft)*, 3.0. Retrieved 08 10, 2011 from <http://www.w3.org/TR/voicexml30/>
- McGlashan, S., Burnett, D. C., Carter, J., Danielsen, P., Ferrans, J., Hunt, A., et al. (Eds.). (2004, March 16). *Voice Extensible Markup Language (VoiceXML) Version 2.0. W3C Recommendation*, 2.0. Retrieved 03 08, 2011 from <http://www.w3.org/TR/voicexml20/>
- McKee, R. (1997). *Story: Substance, Structure, Style and The Principles of Screenwriting*. New York, NY, USA: HarperCollins.
- McTear, M. F. (2004). *Spoken Dialogue Technology*. London: Springer-Verlag.
- McTear, M. F. (2002). Spoken Dialogue Technology: Enabling the Conversational Interface. *ACM Computing Surveys* , 34 (1), 90-169.

- Mead, S., Cavazza, M., & Charles, F. (2003). Influential Words: Natural Language in Interactive Storytelling. *Proceedings of the 10th International Conference on Human-Computer Interaction*. Crete, Greece.
- Milam, D., & Seif El Nasr, M. (2010). Analysis of Level Design 'Push & Pull' within 21 games. *Foundations of Digital Games*. Monterey, California, USA.
- Mori, M. (1970). The Uncanny Valley. *Energy* (4), 33-35.
- Murray, J. H. (1997). *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. Cambridge, MA: MIT Press.
- Nintendo. (2001). Animal Crossing. Nintendo Co., Ltd.
- Orkin, J. (2006). Three States and a Plan: The A.I. of F.E.A.R. *Proceedings of GDC*.
- Paek, T., & Pieraccini, R. (2008). Automating Spoken Dialogue Management Design Using Machine Learning: An Industry Perspective. *Speech Communication* , 50, 716-729.
- Paine, R. (1967). What is gossip about? An Alternative Hypothesis. *Man* , 2 (2), 278-285.
- Pieraccini, R., & Huerta, J. (2005). Where do we go from here? research and commercial spoken dialog systems. *6th SIGdial Workshop on Discourse and Dialogue*, (pp. 1-10). Lisbon.
- Pieraccini, R., Suendermann, D., Dayanidhi, K., & Liscombe, J. (2009). Are We There Yet? Research in Commercial Spoken Dialog Systems. In V. Matoušek, & P. Mautner (Eds.), *Text, Speech and Dialogue* (Vol. 5729, pp. 3-13). Berlin, Heidelberg: Springer.
- Prendinger, H., & Ishizuka, M. (2001a). Let's talk! Socially Intelligent Agents for Language Conversation Training. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans. Special Issue on Intelligent Agents – The Human in the Loop.* , 31 (5), 465-471.
- Prendinger, H., & Ishizuka, M. (2001b). Simulating Affective Communication with Animated Agents. *Proceedings of 8th IFIP TC.13 Conference on Human-Computer Interaction (INTERACT'01)*, (pp. 182-189). Tokyo.
- Pressman, R. S. (2010). *Software Engineering: A Practitioner's Approach* (7th Edition ed.). New York, NY, USA: McGraw-Hill.
- Radomski, S., Schnelle-Walka, D., & Radeck-Arneth, S. (2013). A Prolog Datamodel for State Chart XML. *Proceedings of the SIGDIAL 2013 Conference* (pp. 127-131). Metz, France: Association for Computational Linguistics.

- Reeves, B., & Nass, C. (1996). *The Media Equation: How People Treat Computers, Television, and New Media Like Real People and Places*. New York: Cambridge University Press.
- Reilly, W. S., & Bates, J. (1995). *Natural Negotiation for Believable Agents*. Technical Report CMU-CS-95-164, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, USA.
- Remedy Entertainment. (2001). Max Payne [Computer game]. Gathering.
- Rockstar Games. (1997-2010). Grand Theft Auto (series) [Computer game]. Rockstar Games.
- Rockstar Games. (2011). L.A. Noire [Computer game]. Rockstar Games.
- Rollings, A., & Morris, D. (2000). *Game Architecture and design*. Scottsdale, Ariz.: Coriolis.
- Ryan, M. (2001). Beyond Myth and Metaphor – The Case of Narrative in Digital Media. *Game Studies: The International Journal of Computer Game Research* , 1 (1).
- Sacks, H., Schegloff, E., & Jefferson, G. (1974). A simplest systematics for the organization of turn-taking for conversation. , *Language* , 50 (4), 696-735.
- Salen, K., & Zimmerman, E. (2004). *Rules of Play: Game Design Fundamentals*. Massachusetts: MIT Press.
- Sali, S., Wardrip-Fruin, N., Dow, S., Mateas, M., Kurniawan, S., Reed, A. A., et al. (2010). Playing with Words: From intuition to evaluation of game dialogue interfaces. *Foundation of Digital Games (FDG)* (pp. 179-186). Monterey, CA, USA.: ACM.
- Samek, M. (2002). *Practical Statecharts in C/C++* , , 2002. San Fransisco, CA, USA: CMP-Books.
- Saygin, A. P., Cicekli, I., & Akman, V. (2000). Turing Test: 50 Years Later. *Minds and Machines* , 10 (4), 463-518.
- Schatzmann, J., Weilhammer, K., Stuttle, M., & Young, S. (2006). A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *The Knowledge Engineering Review* , 21 (2), 97-126.
- Schegloff, E. A., & Sacks, H. (1973). Opening up Closings. *Semiotica* , 7 (4), 289-327.
- Searle, J. R. (1969). *Speech Acts: An Essay in the Philosophy of Language*. New York, USA: Cambridge University Press.
- Seinfeld, J., & David, L. (Writers). (1989-1998). *Seinfeld* [Motion Picture].

- Seneff, S., Glass, J., Goddeau, D., Goodine, D., Hirschman, L., Leung, H., et al. (1991). Development and preliminary evaluation of the MIT ATIS System. *Proceedings of the DARPA Speech and Natural Language Workshop*, (pp. 88-93).
- Sheldon, L. (2004). *Character Development and Storytelling for Games*. Boston, MA, USA: Course Technology.
- Sicart, M. (2008). Defining Game Mechanics. *Game Studies: the international journal of computer game research* , 8 (2).
- Skantze, G. (2007). *Error Handling in Spoken Dialogue Systems: Managing Uncertainty, Grounding and Miscommunication*. PhD Thesis, Royal Institute of Technology, Dept of Speech, Music and Hearing, Stockholm.
- Skantze, G. (2003). Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems. *Proceedings of ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*, (pp. 71-76). Chateau-d'Oex-Vaud, Switzerland.
- Skantze, G. (2005). GALATEA: A Discourse Modeller Supporting Concept-level Error Handling in Spoken Dialogue Systems. In L. Dybkjær, & W. Minker (Ed.), *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, (pp. 178-189). Lisbon, Portugal.
- Skantze, G., & Hjalmarsson, A. (2010). Towards Incremental Speech Generation in Dialogue Systems. *Proceedings of the SIGDIAL 2010 Conference* (pp. 1-8). Tokyo, Japan: Association for Computational Linguistics.
- Skantze, G., & Schlangen, D. (2009). Incremental Dialogue Processing in a Micro-Domain. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 745-753). Morristown, NJ, USA: Association for Computational Linguistics.
- Smith, R. W., & Hipp, D. R. (1995). *Spoken Natural Language Dialog Systems: A Practical Approach*. New York, NY, USA: Oxford University Press.
- Sony Computer Entertainment. (2010). Heavy Rain [Computer game]. Sony Computer Entertainment.
- Swain, C. (2008). The Augmented Conversation Engine - A System for Achieving Believable Conversation in Games and Interactive Stories. *Proceedings of ACM SIGCHI International Conference on Advances in Computer Entertainment Technology (ACE 2008)* (pp. 213-218). ACM.
- Swartout, W., Traum, D., Artstein, R., Noren, D., Debevec, P., Bronnenkant, K., et al. (2010). Ada and Grace: Toward Realistic and Engaging Virtual Museum Guides. In J. Allbeck, N. Badler, T. Bickmore, C. Pelachaud, & A. Safonova

(Eds.), *Lecture Notes in Computer Science: Intelligent Virtual Agents* (Vol. 6356, pp. 286-300). Springer Berlin / Heidelberg.

Szwed, J. F. (1966). Gossip, Drinking, and Social Control: Consensus and Communication in a Newfoundland Parish. *Ethnology*, 5 (4), 434-441.

Taylor, M. J., Gresty, D., & Baskett, M. (2006). Computer game-flow design. *Comput. Entertain.*, 4 (1).

Traum, D. (1999). Speech Acts for Dialogue Agents. In M. J. Wooldridge, & A. Rao (Eds.), *Foundations for Rational Agency* (pp. 169-201). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Traum, D., & Larsson, S. (2003). The Information State Approach to Dialogue Management. In J. van Kuppevelt, & R. W. Smith (Eds.), *Current and New Directions in Discourse and Dialogue* (pp. 325-353). Kluwer Academic Publishers.

Traum, D., & Rickel, J. (2002). Embodied agents for multi-party dialogue in immersive virtual worlds. *first International Joint Conference on Autonomous Agents and Multiagents Systems (AAMAS '02)*, (pp. 766-773). Bologna, Italy.

Traum, D., Swartout, W., Gratch, J., & Marsella, S. (2008). A Virtual Human Dialogue Model for Non-Team Interaction. In L. Dybkjær, & W. Minker (Eds.), *Recent Trends in Discourse and Dialogue*.

Tsang, E., & Gosling, T. (2002). Simple Constrained Bargaining Game. In *Distributed Constraint Satisfaction Workshop, First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2002)*. Bologna, Italy.

Ubisoft. (2008). Tom Clancy's EndWar [Computer Game]. Ubisoft.

Walker, M. A., Cahn, J. E., & Whittaker, S. J. (1997). Improvising Linguistic Style: Social and Affective Bases for Agent Personality. *Proceedings of the first international conference on Autonomous agents, Feb 5-8*, pp. 96-105. Marina del Rey, USA.

Walker, M. A., Cahn, J. E., & Whittaker, S. J. (1996). Linguistic Style Improvisation for Lifelike Computer Characters. In P. C. Hiroaki Kitano (Ed.), *Entertainment and AI/A-Life: Papers from 1996 AAAI Workshop*.

Walker, M., Kamm, C., & Litman, D. (2000). Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6 (3&4), 363-377.

Wallace, R. (2005). *Artificial Intelligence Markup Language (AIML) Version 1.0.1. Working Draft*. From <http://docs.aitools.org/aiml/spec/WD-aiml>

- Weizenbaum, J. (1966). Eliza - a Computer Program for the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, 9 (1), 36-45.
- Wertheimer, M. (1938). Gestalt theory. In W. D. Ellis (Ed.), *A source book of Gestalt psychology*. London: Routledge & Kegan Paul.
- Wik, P. (2011). *The Virtual Language Teacher: Models and applications for language learning using embodied conversational agents*. PhD Thesis, KTH School of Computer Science and Communication.
- Wik, P., Hjalmarsson, A., & Brusk, J. (2007a). Computer Assisted Conversation Training for Second Language Learners. *Proceedings of Fonetik. 50*, pp. 57-60. Stockholm, Sweden: Speech, Music and Hearing, KTH, TMH-QPSR.
- Wik, P., Hjalmarsson, A., & Brusk, J. (2007b). DEAL A Serious Game For CALL Practicing Conversational Skills In The Trade Domain. *Proceedings of the SLaTE Workshop on Speech and Language Technology in Education*. Farmington, Pennsylvania USA: Carnegie Mellon University and International Speech Communication Association (ISCA).
- Wilhelmsson, U. (2001). *Enacting the Point of Being: Computer Games, Interaction and Film Theory*. PhD Thesis, University of Copenhagen (University of Skövde), Dept of Film and Media Studies (Dept of Humanities).
- Visser, T., Traum, D., DeVault, D., & op den Akker, R. (2012). Toward a Model for Incremental Grounding in Spoken Dialogue Systems. *Workshop on Real-Time Conversations with Virtual Agents (RCVA 2012)*. Santa Cruz, California.
- Wittgenstein, L. (1953). *Philosophische Untersuchungen (Philosophical investigations)* (3. ed., repr ed.). (G. E. Anscombe, Trans.) Oxford: Blackwell Publishing.
- Yee, N. (2005). Motivations of Play in MMORPGs. In S. de Castell, & J. Jenson (Ed.), *Proceedings of DiGRA 2005: Changing Views: Worlds in Play*. Vancouver: University of Vancouver.
- Young, S., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., et al. (2010). The Hidden Information State Model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24 (2), 150-174.
- Yue, B., & de-Byl, P. (2006). The State of the Art in Game AI Standardisation. *Proceedings of the 2006 international conference on Game research and development* (pp. 41-46). Murdoch University, Australia, Australia: Murdoch University.
- Zagal, J. P., Mateas, M., Fernández-Vara, C., Hochhalter, B., & Lichti, N. (2005). Towards an Ontological Language for Game Analysis. In S. de Castell, & J.

Jenson (Ed.), *Changing Views: Worlds in Play (DiGRA 2005)*. Vancouver: University of Vancouver.

Zubek, R. (2005). *Hierarchical Parallel Markov Models for Interactive Social Agents*. PhD Thesis, Northwestern University, Computer Science Department.

Zubek, R., & Horswill, I. D. (2005). Hierarchical Parallel Markov Models of Interaction. *Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2005) Conference*, (pp. 141-146). Marina del Ray, CA.

Zynga Inc. (2009). *Café World* [Computer Game].

Appendix I: Waiter DM

```
<scxml initial="Environment" scriptlanguage="javascript"
id="top">
  <parallel id="Environment">
    <datamodel>
      <data ID="customers">
        <customers>
          <customer>
            <name>Lisa</name>
            <sd>1</sd>
            <p>1</p>
            <regular>true</regular>
            <relation>8</relation>
            <regorder>latte</regorder>
          </customer>
          <customer>
            <name>Pelle</name>
            <sd>2</sd>
            <p>3</p>
            <regular>true</regular>
            <relation>2</relation>
            <regorder>espresso</regorder>
          </customer>
          <customer>
            <name>Kalle</name>
            <sd>2</sd>
            <p>1</p>
            <regular>false</regular>
            <relation>5</relation>
          </customer>
        </customers>
      </data>
      <data ID="greet_phrases">
        <phrases>
          <rude>Yeah?</rude>
          <rude>What's up?</rude>
          <rude>Huh?</rude>
          <rude>Ok?</rude>
          <polite>Welcome to our coffee shop!</polite>
          <polite>Good day!</polite>
          <polite>Please feel welcome!</polite>
          <polite>Nice day so far?</polite>
          <neutral>Welcome</neutral>
          <neutral>Can I help you?</neutral>
          <neutral>Hello</neutral>
          <neutral>Hi there!</neutral>
          <flirt>Ciao bella!</flirt>
          <flirt>Hello there beauty!</flirt>
        </phrases>
      </data>
    </parallel>
  </scxml>
```

```

<data ID="take_order_phrases">
  <phrases>
    <rude>What do you want?</rude>
    <rude>So, what's it gonna be?</rude>
    <rude>I don't have all day!</rude>
    <rude>Wanna leave?</rude>
    <polite>May I take your order?</polite>
    <polite>Can we offer you something?</polite>
    <polite>We have espresso, latte, regular coffee
    </polite>
    <polite>Do you find anything in your taste?
    </polite>
    <neutral>What are you having?</neutral>
    <neutral>Have you decided yet?</neutral>
    <neutral>What can I serve you?</neutral>
    <neutral>Latte, espresso or regular coffee?
    </neutral>
    <flirt>What can I get you, sweetie?</flirt>
    <flirt>Have you anything in mind, apart from
      me?</flirt>
  </phrases>
</data>
<data ID="reject_phrases">
  <phrases>
    <rude>Do you see any alcohol here?</rude>
    <rude>Does this look like a bar to you?!</rude>
    <rude>Didn't I just tell you we don't serve
      alcohol?</rude>
    <rude>We don't have that!</rude>
    <polite>I am sorry, but we only serve coffee
      here</polite>
    <polite>I am truly sorry, but this is a
      coffeeshop</polite>
    <polite>I am afraid you can only buy coffee
      here</polite>
    <polite>I am sorry to say this, but we only serve
      coffee here</polite>
    <neutral>Sorry, we only have coffee</neutral>
    <neutral>You can only buy coffee here</neutral>
    <neutral>I can only offer you coffee</neutral>
    <neutral>We only serve coffee here</neutral>
    <flirt>Sorry dear, this is a coffeeshop</flirt>
    <flirt>Me too, is that a date?</flirt>
  </phrases>
</data>
<data ID="accept_phrases">
  <phrases>
    <rude>Two dollars</rude>
    <rude>Ok, Two dollars</rude>
    <rude>Two dollars even</rude>
    <rude>Two dollars excluding tips</rude>
    <polite>Very well, that will be two dollars
      please</polite>
    <polite>Thank you. Two dollars please</polite>
  </phrases>
</data>

```

```

    <polite>That will be two dollars, thank
        you</polite>
    <polite>Thank you for your order. That\'ll be two
        dollars</polite>
    <neutral>OK, Two dollars please</neutral>
    <neutral>OK, that\'ll be two dollars</neutral>
    <neutral>Two dollars please</neutral>
    <neutral>Now that sums up to two
        dollars</neutral>
</phrases>
</data>
<data ID="resp_req_day_off_phrases">
    <phrases>
        <polite>Good afternoon Boss, Would it be okay for
            me to take the day off tomorrow?</polite>
        <neutral>Excuse me Boss, may I take the day off
            tomorrow?</neutral>
    </phrases>
</data>
<data ID="viol_req_day_off_phrases">
    <phrases>
        <polite>Good Afternoon Boss, I need to take the
            day off tomorrow</polite>
        <neutral>Hey Boss, I will take the day off
            tomorrow</neutral>
        <rude>Boss! I will not be here tomorrow, just for
            the record</rude>
    </phrases>
</data>
<data ID="reject_day_off_phrases">
    <phrases>
        <rude>No way! We are busy as hell tomorrow so I
            will need you here!</rude>
        <neutral>Actually, I need you tomorrow. Thank
            you.</neutral>
        <polite>It will be a busy day. So I kindly ask
            you to come.</polite>
    </phrases>
</data>
<data ID="resp_response_reject_phrases">
    <phrases>
        <polite>Okay, Boss, I will be here</polite>
        <neutral>Alright, I</neutral>
        <rude>We're not slaves you know!</rude>
    </phrases>
</data>
<data ID="viol_response_reject_phrases">
    <phrases>
        <polite>You don't have to thank me because I will
            not be here tomorrow!</polite>
        <neutral>Well, I will not be here tomorrow so you
            better find a replacer</neutral>
        <rude>To my knowledge we do not hold people as
            slaves in this country so don\'t expect to

```

```

        see me here tomorrow</rude>
    </phrases>
</data>
<data ID="propose_order">
    <offer>
        <drink name="'espresso'">
            <text>Can I offer you an espresso today?</text>
            <type>coffee</type>
        </drink>
        <drink name="'latte'">
            <text>Latte?</text>
            <type>coffee</type>
        </drink>
        <drink name="'regular'">
            <text>Would you like a regular coffee?</text>
            <type>coffee</type>
        </drink>
    </offer>
</data>
</datamodel>
<parallel id="Waiter1">
    <datamodel>
        <data ID="coworkers">
            <coworkers>
                <coworker>
                    <name>waiter2</name>
                    <threat>2</threat>
                    <resp_conv_pract>>false</resp_conv_pract>
                </coworker>
                <coworker>
                    <name>friendly_manager</name>
                    <threat>5</threat>
                    <resp_conv_pract>>false</resp_conv_pract>
                </coworker>
            </coworkers>
        </data>
        <data ID="respect_social_conv"/>
        <data ID="current_customer"/>
        <data ID="newC" expr="0" eval="true"/>
        <data ID="addressee"/>
        <data ID="manner"/>
    </datamodel>
    <state id="Actions">
        <initial>
            <transition target="Idle"/>
        </initial>
        <transition event="approach"
            cond="event.data.data.role=='coworker'">
            <script>
addressee=event.data.data.name.text()
resp=coworkers.coworker.(name==addressee).resp_conv_pract.*

coworkerThreat=parseInt(coworkers.coworker.(name==addressee)
).threat.*)

```

```

        </script>
    </transition>
    <history id="H_acts"/>
    <state id="Idle">
        <onentry>
            <log expr="'in idle'"/>
        </onentry>
        <transition event="approach"
                    cond="event.data.data.role=='customer'"
                    target="Service">
            <script>
current_customer=event.data.data.name.text()
respect_social_conv='true'
threat=parseInt(customers.customer.(name==current_customer)
.sd.*) +
parseInt(customers.customer.(name==current_customer).p.*)
newC=1
            </script>
        </transition>
    </state>
    <state id="Service">
        <onentry>
            <log expr="'in service'"/>
            <log expr="current_customer"/>
            <log expr="threat"/>
        </onentry>
        <transition event="exit_customer_dialogue"
                    target="Idle">
            <script> newC=0 </script>
        </transition>
    </state>
</state>
<state id="Dialogue_acts">
    <initial>
        <transition target="Init_Dialogue"/>
    </initial>
    <state id="Init_Dialogue">
        <onentry>
            <log expr="'in initdial'"/>
        </onentry>
        <transition event="approach"
                    cond="event.data.data.role=='customer'"
                    target="Service_conv"/>
        <transition event="approach"
                    cond="event.data.data.role=='coworker'"
                    target="Internal_conv"/>
    </state>
    <state id="Service_conv">
        <datamodel>
            <data ID="greet"/>
            <data ID="take_order" expr="0" eval="true"/>
            <data ID="phraseNo" expr="0" eval="true"/>
            <data ID="order" expr="'nil'"/>
        </datamodel>
    </state>

```

```

<initial>
  <transition target="Select_DA"/>
</initial>
<history id="H" type="deep">
  <transition target="Select_DA"/>
</history>
<transition event="approach"
  cond="event.data.data.role=='coworker'"
  target="Internal_conv"/>
<state id="Select_DA">
  <onentry>
    <raise sendid="a" event="timeout"
      delay="500ms"/>
    <raise sendid="orderTO"
      event="takeorder_timeout" delay="1s"/>
  </onentry>
  <transition cond="order=='done'"
    target="Init_Dialogue">
    <raise sendid="done"
      event="exit_customer_dialogue"
      delay="100ms"/>
  </transition>
  <transition event="timeout"
    cond="take_order==0 &&&
      In('Anger') &&& 3>threat"
    target="Take_order">
    <script>manner='neutral'
      greet='true'</script>
  </transition>
  <transition event="timeout"
    cond="greet!='true'"
    target="Greet">
    <script>
      if(threat>2){manner='polite'}
      else{manner='neutral';}
    </script>
  </transition>
  <transition event="cGreet"
    cond="greet!='true' &&&
      (event.data=='polite' || threat>2)"
    target="Greet">
    <script>manner='polite'</script>
  </transition>
  <transition event="cGreet"
    cond="event.data=='rude' &&&
      3>threat"
    target="Take_order">
    <script>manner='neutral'
      greet='true'</script>
  </transition>
  <transition event="cGreet"
    cond="greet!='true'"
    target="Greet">
    <script>manner='neutral'</script>

```

```

</transition>
<transition event="timeout"
            cond="greet=='true' &&&
                order=='nil' &&&
                take_order==0"
            target="Take_order"/>
<transition event="takeorder_timeout"
            target="Take_order"/>
<transition event="order"
            target="Handle_order">
    <script>
        order=event.data.order
    </script>
</transition>
<onexit>
    <cancel sendid="a"/>
    <cancel sendid="orderTO"/>
    <cancel sendid="req"/>
</onexit>
</state>
<state id="Greet">
    <onentry>
        <raise sendid="b" event="wGreet"
            delay="200ms"/>
        <script> textp='greet_phrases.' + manner +
            '[' + phraseNo + '].text()';
            if(phraseNo>2) {phraseNo=1;} else
            {phraseNo+=1;}
            greet='true'
        </script>
        <log expr="eval(textp)"/>
    </onentry>
    <transition target="Select_DA"/>
    <onexit>
        <cancel sendid="b"/>
    </onexit>
</state>
<state id="Take_order">
    <onentry>
        <script>
            textp='take_order_phrases.' + manner + '[' +
                take_order + '].text()'
            if(take_order>2) {take_order=1;} else
            {take_order+=1;}
        </script>
        <log label="'waiter'" expr="eval(textp)"/>
        <raise sendid="req" event="req_order"
            delay="100ms"/>
    </onentry>
    <transition target="Select_DA"/>
</state>
<state id="Handle_order">
    <initial>
        <transition target="Select_response"/>

```

```

</initial>
<transition event="serve_response"
            target="Select_DA"/>
<state id="Select_response">
  <onentry>
    <log label="'current order'" expr="order"/>
  </onentry>
  <transition cond="order=='beer' ||
                order='wine'"
            target="Refuse_serve"/>
  <transition cond="order=='latte' ||
                order=='espresso' ||
                order=='regular' ||
                order=='coffee'"
            target="Serve"/>
</state>
<state id="Refuse_serve">
  <onentry>
    <script>textp='reject_phrases.' + manner +
      '[' + phraseNo + '].text()'
      if(phraseNo>2) {phraseNo=1;} else
      {phraseNo+=1;}
      order='nil'</script>
    <log label="'waiter'" expr="eval(textp)"/>
    <raise event="serve_response"
          expr="'reject'" delay="200ms"/>
  </onentry>
</state>
<state id="Serve">
  <onentry>
    <script>textp='accept_phrases.' + manner +
      '[' + phraseNo + '].text()'
      if(phraseNo>2) {phraseNo=1;} else
      {phraseNo+=1;}
      order='done'; </script>
    <log label="'waiter'" expr="eval(textp)"/>
    <raise event="serve_response"
          expr="'accept'" delay="200ms"/>
  </onentry>
</state>
</state>
</state>
<state id="Internal_conv">
  <datamodel>
    <data ID="reqdayoff"/>
  </datamodel>
  <initial>
    <transition target="Select_act"/>
  </initial>
  <state id="Select_act">
    <transition event="exit_internal_conv"
              target="H"
              cond="In('Cust1_dialogue')">
      <log expr="'going to H'"/>
    </transition>
  </state>
</state>

```



```

</transition>
<transition event="exit_internal_conv"
            target="Init_Dialogue">
    <log expr="'going to init'"/>
</transition>
<transition cond="addressee!='waiter2'
                &&& reqdayoff!='true'
                &&& In('Anger')"
            target="Request_dayoff">
    <script>
        if(coworkerThreat>4){manner='neutral'}
        else{manner='rude';}
        reqdayoff='true';
    </script>
</transition>
<transition cond="addressee!='waiter2'
                &&& reqdayoff!='true'"
            target="Request_dayoff">
    <script>
        if(coworkerThreat>4){manner='polite'}
        else{manner='neutral';}
        reqdayoff='true';
    </script>
</transition>
<transition event="manager_response"
            cond="event.data=='reject'"
            target="Wrap_up">
    <script> response='reject'; </script>
    <raise event="change_emotion" expr="'anger'"
            delay="50ms"/>
</transition>
<transition event="manager_response"
            cond="event.data=='accept'"
            target="Wrap_up">
    <script> response='accept';</script>
    <raise event="change_emotion" expr="'joy'"
            delay="50ms"/>
</transition>
</state>
<state id="Request_dayoff">
    <onentry>
        <raise event="req_dayoff" delay="200ms"/>
        <script> texttp='viol_req_dayoff_phrases.' +
            manner + '[0].text()' </script>
        <log expr="eval(texttp)"/>
    </onentry>
    <transition target="Select_act"/>
</state>
<state id="Wrap_up">
    <transition cond="In('Anger') &&&
                addressee=='friendly_manager'
                &&& response=='reject'"
            target="Select_act">
        <log label="'waiter'"

```

```

    expr="viol_response_reject_phrases.neutral.text()"/>
        <raise event="exit_internal_conv"
            delay="100ms"/>
    </transition>
    <transition cond="addressee=='friendly_manager'
        && response=='reject'"
        target="Select_act">
        <log label="'waiter'"
    expr="viol_response_reject_phrases.polite.text()"/>
        <raise event="exit_internal_conv"
            delay="100ms"/>
    </transition>
    <onexit>
        <cancel sendid="wuto"/>
    </onexit>
</state>
</state>
</state>
</state>
<state id="Emotion">
    <transition event="change_emotion"
        cond="event.data=='joy'"
        target="Joy"/>
    <transition event="change_emotion"
        cond="event.data=='anger'"
        target="Anger"/>
    <transition event="change_emotion"
        cond="event.data=='neutral'"
        target="Neutral"/>
    <initial>
        <transition target="Neutral"/>
    </initial>
    <state id="Joy">
        <onentry>
            <log expr="':)'" />
        </onentry>
    </state>
    <state id="Neutral">
        <onentry>
            <log expr="':|'" />
        </onentry>
    </state>
    <state id="Anger">
        <onentry>
            <log expr="':('"/>
        </onentry>
    </state>
</state>
</parallel>
<parallel id="Friendly_Manager">
    <state id="Friendly_Manager_DAs">
        <transition event="req_dayoff"
            target="Friendly_Manager_DAs">
            <raise event="manager_response" expr="'reject'"
                delay="500ms"/>

```

```

        <log label="'manager'"
            expr="reject_dayoff_phrases.neutral.text()"/>
    </transition>
</state>
<state id="Friendly_Manager_Acts">
    <datamodel>
        <data ID="friendlyManagerData">
            <data>
                <role>coworker</role>
                <name>friendly_manager</name>
            </data>
        </data>
    </datamodel>
    <initial>
        <transition target="Friendly_Manager_Act_idle"/>
    </initial>
    <state id="Friendly_Manager_Act_idle">
        <onentry>
            <raise sendid="m" event="timeoutact"
                delay="1s"/>
        </onentry>
        <transition event="timeoutact"
            target="Friendly_Manager_Approach_counter"/>
    </state>
<state id="Friendly_Manager_Approach_counter">
    <onentry>
        <raise event="approach"
            expr="friendlyManagerData"
            delay="300ms"/>
        <log expr="'manager approaching'"/>
    </onentry>
</state>
</state>
</parallel>
<state id="Customers">
    <initial>
        <transition target="Customer1"/>
    </initial>
    <parallel id="Customer1">
        <datamodel>
            <data ID="manner" expr="'polite'"/>
            <data ID="thinkingdone"/>
            <data ID="cust1data">
                <data>
                    <role>customer</role>
                    <name>Lisa</name>
                </data>
            </data>
            <data ID="orderprio">
                <orders>
                    <order>latte</order>
                    <order>beer</order>
                    <order>regular</order>
                </orders>
            </data>
        </datamodel>
    </parallel>
</state>

```

```

    </data>
</datamodel>
<script> orderNo=0;
    line_length=0; </script>
<state id="Cust1_actions">
  <initial>
    <transition target="Stand_in_line">
      <script> line_length+=1
        linepos=line_length-1;</script>
    </transition>
  </initial>
  <state id="Stand_in_line">
    <transition event="next" cond="linepos>0"
      target="Stand_in_line">
      <script>
        linepos-=1;
      </script>
    </transition>
    <transition cond="linepos==0"
      target="First_in_line">
      <raise event="approach" expr="cust1data"
        delay="200ms"/>
    </transition>
  </state>
  <state id="First_in_line">
    <onentry>
      <log expr="'first in line'"/>
    </onentry>
    <transition event="exit_customer_dialogue"
      target="Leave_line">
      <raise event="next" delay="200ms"/>
    </transition>
  </state>
  <final id="Leave_line"/>
</state>
<state id="Cust1_dialogue">
  <initial>
    <transition target="Cust1_greet"/>
  </initial>
  <state id="Cust1_greet">
    <onentry>
      <raise event="timeout" delay="700ms"/>
    </onentry>
    <transition event="timeout"
      cond="In('First_in_line')"
      target="Cust1_order">
      <log label="cust1data.name"
        expr="'Good day'"/>
      <raise event="cGreet" expr="manner"
        delay="200ms"/>
    </transition>
    <transition event="wGreet"
      cond="In('First_in_line')"
      target="Cust1_order">

```

```

        <log label="custldata.name"
            expr="'Good day'"/>
        <raise event="cGreet" expr="manner"
            delay="200ms"/>
    </transition>
</state>
<state id="Cust1_order">
    <onentry>
        <log expr="'in custlorder'"/>
        <raise event="thinking" delay="800ms"/>
    </onentry>
    <transition event="thinking">
        <script>thinkingdone='true'; </script>
    </transition>
    <transition event="req_order"
        cond="3>orderNo & & thinkingdone=='true'"
        target="Cust1_order">
        <raise event="order" delay="200ms"
            expr="orderprio.order[orderNo]"/>
        <script> textp='I would like a ' +
            orderprio.order[orderNo].text() + ',
            please';</script>
        <log label="custldata.name" expr="textp"/>
        <script> orderNo+=1; </script>
    </transition>
    <transition event="serve_response"
        cond="event.data=='accept'"
        target="Cust1_end_dialogue">
        <log label="custldata.name"
            expr="'That looks great! Thank you!'/>
    </transition>
    <transition event="serve_response"
        cond="event.data=='reject'"
        target="Cust1_order">
        <log label="custldata.name"
            expr="'Okay, Hmmm'"/>
    </transition>
</state>
<state id="Cust1_end_dialogue">
    <transition event="exit_customer_dialogue"
        target="leave">
        <log label="custldata.name"
            expr="'Goodbye!'/>
    </transition>
</state>
<final id="leave"/>
</state>
</parallel>
</state>
</parallel>
</scxml>

```

Appendix II: Waiter DM run – log

```
09.44.30  :|
09.44.30  in initdial
09.44.30  in idle
09.44.30  Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_greet,Cust1_actions,Stand_in_line,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Init_Dialogue,Actions,Idle}
09.44.30  first in line
09.44.30  Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_greet,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Init_Dialogue,Actions,Idle,First_in_line}
09.44.31  Ext. Event      <event> <name>approach</name>
                                <data> <data>
                                <role>customer</role>
                                <name>Lisa</name> </data>
                                </data> </event>
09.44.31  in service
09.44.31  Lisa
09.44.31  2
09.44.31  Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_greet,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Select_DA,Service}
```

09.44.31 Ext. Event <event> <name>timeoutE</name>
<data/> </event>

09.44.31 Lisa Good day

09.44.31 in cust1 order

09.44.31 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Select_DA,Service,Cust1_order}

09.44.31 Ext. Event <event> <name>timeout</name>
<data/> </event>

09.44.31 Welcome

09.44.31 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Service,Cust1_order,Greet}

09.44.31 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_Act_idle,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Service,Cust1_order,Select_DA}

09.44.31 Ext. Event <event> <name>cGreet</name>
<data>polite</data> </event>

09.44.31 Ext. Event <event> <name>timeoutact</name>
<data/> </event>

09.44.31 manager approaching

09.44.31 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Service,Cust1_order,Select_DA,Friendly_Manager_Ap

```
proach_counter}
```

- 09.44.32 Ext. Event <event> <name>timeout</name>
<data/> </event>
- 09.44.32 waiter1 What are you having?
- 09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Service,Cust1_order,Friendly_Manager_Approach_counter,Take_order}
- 09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service_conv,Service,Cust1_order,Friendly_Manager_Approach_counter,Select_DA}
- 09.44.32 Ext. Event <event> <name>approach</name>
<data> <data>
<role>coworker</role>
<name>friendly_manager</name>
</data> </data> </event>
- 09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_Approach_counter,Internal_conv,Select_act}
- 09.44.32 Good Afternoon Boss, I need to take the day off tomorrow

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_Approach_counter,Internal_conv,Request_dayoff}

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_Approach_counter,Internal_conv,Select_act}

09.44.32 Ext. Event <event> <name>thinking</name><data/> </event>

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Friendly_Manager_Acts,Friendly_Manager_DAs,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_Approach_counter,Internal_conv,Select_act}

09.44.32 Ext. Event <event> <name>req_dayoff</name><data/> </event>

09.44.32 manager Actually, I need you tomorrow. Thank you.

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Internal_conv,Select_act,Friendly_Manager_DAs}

09.44.32 Ext. Event <event><name>manager_response</name><data>reject</data> </event>

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Internal_conv,Friendly_Manager_DAs,Wrap_up}

09.44.32 waiter1 You don't have to thank me because I will not be here tomorrow!

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Internal_conv,Friendly_Manager_DAs,Select_act}

09.44.32 Ext. Event <event>
<name>exit_internal_conv</name>
<data/> </event>

09.44.32 going to H

09.44.32 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_DAs,Service_conv,Select_DA}

09.44.33 Ext. Event <event> <name>timeout</name>
<data/> </event>

09.44.33 Ext. Event <event>
<name>takeorder_timeout</name>
<data/> </event>

09.44.33 waiter Can we offer you something?

09.44.33 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_DAs,Service_conv,Take_order}

09.44.33 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Cust1_order,Friendly_Manager_DAs,Service_conv,Select_DA}

09.44.34 Ext. Event <event> <name>req_order</name>
<data/> </event>

09.44.34 Lisa I would like a latte, please

09.44.34 in custlorder

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Friendly_Manager_DAs,Service_conv,Select_DA,Cust1_order}

09.44.34 Ext. Event <event> <name>order</name>
<data> <order>latte</order>
</data> </event>

09.44.34 current order latte

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Friendly_Manager_DAs,Service_conv,Cust1_order,Handle_order,Select_response}

09.44.34 waiter Thank you. Two dollars please

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Friendly_Manager_DAs,Service_conv,Cust1_order,Handle_order,Serve}

09.44.34 Ext. Event <event>
<name>serve_response</name>
<data>accept</data> </event>

09.44.34 Lisa That looks great! Thank you!

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Friendly_Manager_DAs,Service_conv,Cust1_end_dialogue,Select_DA}

09.44.34 in initdial

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,First_in_line,Service,Friendly_Manager_DAs,Cust1_end_dialogue,Init_Dialogue}

09.44.34 Ext. Event <event>
<name>exit_customer_dialogue</name> <data/> </event>

09.44.34 Lisa Goodbye!

09.44.34 in idle

09.44.34 Configuration {top,Environment,Customers,Customer1,Cust1_dialogue,Cust1_actions,Waiter2,Friendly_Manager,Waiter1,Emotion,Neutral,Dialogue_acts,Actions,Friendly_Manager_DAs,Init_Dialogue,leave,Leave_line,Idle}

09.44.34 Int. Event Cust1_dialogue.Done

09.44.34 Int. Event Cust1_actions.Done

09.44.34 Int. Event Customer1.Done

09.44.34 Ext. Event <event> <name>next</name>
<data/> </event>

09.44.34 Ext. Event <event> <name>thinking</name>
<data/> </event>

Appendix III: DEAL

```
<?xml version="1.0"?>
<!-- synergy version 1.0-->
<scxml initialstate="start">
  <datamodel>
    <data name="Nil"/>
    <data name="OUD"/>
    <data name="Color"/>
    <data name="Price"/>
    <data name="Cost"/>
    <data name="Type"/>
    <data name="Size"/>
    <data name="Pricerange"/>
    <data name="Text" expr="'no object is selected yet'"/>
    <data name="RoundsLeft"/>
    <data name="PreviousBid" expr="0" eval="true"/>
    <data name="Interval"/>
    <data name="Rounds" eval="true">
      fun {$ Price}
        if 10>=Price then ({OS.rand()} mod 5) + 1
        else ({OS.rand()} mod 5) + 5 end
      end
    </data>
    <data name="NextOffer" eval="true">
      fun {$ P B R I}
        if R==0 then R = 1 end
        if I>=(P-B) div R then P-((P-B) div R)
        else P-I
        end
      end
    </data>
  </datamodel>
  <parallel id="start">
    <state id="system">
      <initial>
        <transition target="opening"/>
      </initial>
    <state id="opening">
      <initial>
        <transition target="greeting"/>
      </initial>
      <transition event="ca"
        cond="Eventdata.ca_type==request_object"
        target="search_object">
        <assign name="Color"
          expr="{CondSelect Eventdata color Color}"/>
        <assign name="Type"
          expr="{CondSelect Eventdata type Type}"/>
        <assign name="Size"
          expr="{CondSelect Eventdata size Size}"/>
      </transition>
    </state>
  </parallel>
</scxml>
```

```

    <assign name="Pricerange"
           expr="{CondSelect Eventdata pricerange
                  Pricerange}"/>
    <log label="user" expr="Eventdata.string"/>
    <log label="system" expr="Eventdata.grounding"/>
</transition>
<transition event="noinput" target="trading"/>
<state id="greeting">
  <transition event="ca"
              cond="Eventdata.ca_type==greeting">
    <log label="user" expr="Eventdata.string"/>
    <send target="Self" event="greet"
         expr="o(agent:system string:'Welcome')"/>
  </transition>
  <transition event="help"
              target="offerAssistance"/>
</state> <!--greeting-->
<state id="offerAssistance">
  <onentry>
    <log label="system" expr="'How can I help
                              you?'"/>
    <send target="Self" event="assist"
         expr="o(agent:system string:'How can I
                              help you?')"/>
  </onentry>
</state> <!--offerassistance-->
</state> <!--opening-->
<state id="trading">
  <initial>
    <transition target="define_ooi"/>
  </initial>
  <transition event="ca"
              cond="Eventdata.ca_type==cancel_transaction"
              target="ending">
    <log label="Eventdata.agent"
         expr="Eventdata.string"/>
  </transition>
<state id="define_ooi">
  <initial>
    <transition target="supp_obj_info"/>
  </initial>
  <transition event="ca"
              cond="Eventdata.ca_type==request_object"
              target="search_object">
    <assign name="Type"
           expr="{CondSelect Eventdata type Type}"/>
    <assign name="Color"
           expr="{CondSelect Eventdata color Color}"/>
    <assign name="Size"
           expr="{CondSelect Eventdata size Size}"/>
    <assign name="Pricerange"
           expr="{CondSelect Eventdata pricerange
                  Pricerange}"/>
    <log label="user" expr="Eventdata.string"/>

```

```

    <log label="system"
        expr="Eventdata.grounding"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==accept
        andthen OUD\=undefined"
    target="resolve_exchange">
    <log expr="Eventdata.string"/>
</transition>
<state id="present_selected_object">
    <onentry>
        <log label="system"
            expr="'How about '#Text#'?'" />
        <send target="Self" event="present_object"
            expr="o(agent:system type:Type
                color:Color size:Size string:Text)"/>
    </onentry>
    <transition event="ca"
        cond="Eventdata.ca_type==request_price">
        <log label="user" expr="Eventdata.string"/>
        <log label="system"
            expr="'It costs '#Price'"/>
        <send target="Self" event="present_price"
            expr="o(agent:system price:Price)"/>
    </transition>
    <transition event="ca"
        cond="Eventdata.ca_type==reject_object"
        target="define_ooi">
        <log label="user" expr="Eventdata.string"/>
        <assign name="OUD" expr="Nil"/>
        <assign name="Color" expr="Nil"/>
        <assign name="Type" expr="Nil"/>
        <assign name="Size" expr="Nil"/>
        <assign name="Pricerange" expr="Nil"/>
        <assign name="Price" expr="Nil"/>
        <assign name="RoundsLeft" expr="Nil"/>
    </transition>
    <transition event="ca"
        cond="Eventdata.ca_type==reject_price"
        target="negotiation"/>
    <transition event="ca"
        cond="Eventdata.ca_type==counteroffer"
        target="negotiation">
        <assign name="PreviousBid"
            expr="{CondSelect Eventdata offer
                PreviousBid}"/>
        <log label="Eventdata.agent"
            expr="Eventdata.string#' '#PreviousBid"/>
    </transition>
    <transition event="ca"
        cond="Eventdata.ca_type==reject_type"
        target="supp_obj_info">
        <assign name="OUD" expr="Nil"/>
        <assign name="Color" expr="Nil"/>

```

```

    <assign name="Type" expr="Nil"/>
    <assign name="Size" expr="Nil"/>
    <assign name="Pricerange" expr="Nil"/>
    <assign name="Price" expr="Nil"/>
    <assign name="RoundsLeft" expr="Nil"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==reject_color"
    target="supp_obj_info">
    <assign name="Color" expr="Nil"/>
    <assign name="OUD" expr="Nil"/>
    <assign name="Price" expr="Nil"/>
    <assign name="RoundsLeft" expr="Nil"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==reject_pricerange"
    target="supp_obj_info">
    <assign name="OUD" expr="Nil"/>
    <assign name="Pricerange" expr="Nil"/>
    <assign name="Price" expr="Nil"/>
    <assign name="RoundsLeft" expr="Nil"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==reject_size"
    target="supp_obj_info">
    <assign name="Size" expr="Nil"/>
    <assign name="OUD" expr="Nil"/>
    <assign name="Price" expr="Nil"/>
    <assign name="RoundsLeft" expr="Nil"/>
</transition>
</state> <!--present_selected_object-->
<state id="supp_obj_info" >
    <initial>
        <transition target="get_info"/>
    </initial>
    <state id="get_info">
        <initial>
            <transition target="select_properties"/>
        </initial>
        <state id="select_properties">
            <transition target="ask_type"
                cond="Type==undefined"/>
            <transition target="ask_color"
                cond="Color==undefined"/>
            <transition target="ask_size"
                cond="Size==undefined"/>
            <transition target="ask_pricerange"
                cond="Pricerange==undefined"/>
        </state> <!--select_properties-->
        <state id="ask_type">
            <onentry>
                <log expr="'Are you looking for something
                    in particular? We have clocks, swords
                    and other things'"/>
            </onentry>
        </state>
    </state>
</state>

```



```

        <send target="Self" event="asktype"/>
    </onentry>
    <transition target="gather_prop_info"/>
</state> <!--ask_type-->
<state id="ask_color">
    <onentry>
        <log label="system"
            expr="'What color would you prefer?'" />
        <send target="Self" event="askcolor"/>
    </onentry>
    <transition target="gather_prop_info"/>
</state> <!--ask_color-->
<state id="ask_size">
    <onentry>
        <log label="system"
            expr="'What size do you prefer?'" />
        <send target="Self" event="asksize"/>
    </onentry>
    <transition target="gather_prop_info"/>
</state> <!--ask_size-->
<state id="ask_pricerange">
    <onentry>
        <log label="system"
            expr="'And what pricerange did you
                have in mind?'" />
        <send target="Self"
            event="askpricerange"/>
    </onentry>
    <transition target="gather_prop_info"/>
</state> <!--ask_pricerange-->
<state id="gather_prop_info">
    <transition event="ca"
        cond="Eventdata.ca_type==request_object"
        target="search_object">
        <assign name="Type"
            expr="{CondSelect Eventdata
                type Type}" />
        <assign name="Color"
            expr="{CondSelect Eventdata
                color Color}" />
        <assign name="Size"
            expr="{CondSelect Eventdata
                size Size}" />
        <assign name="Pricerange"
            expr="{CondSelect Eventdata
                pricerange Pricerange}" />
        <log label="Eventdata.agent"
            expr="Eventdata.string" />
        <log label="system"
            expr="Eventdata.grounding" />
    </transition>
    <transition event="ca"
        cond="Eventdata.ca_type==no_match"
        target="select_properties">

```

```

        <log expr="'I am not sure of what you
            mean now!'" />
    </transition>
</state> <!--gather_prop_info-->
</state> <!--get_info-->
<parallel id="search_object">
    <state id="DB">
        <invoke id="dbq" targettype="x-dbQuery">
            <content>
                <dbQuery mode="unique">
                    <entry>
                        <entityid>2</entityid>
                        <text>'a cheap red and medium
                            size clock'</text>
                        <size>2</size>
                        <color>red</color>
                        <type>clock</type>
                        <price>10</price>
                        <cost>7</cost>
                        <pricerange>1</pricerange>
                        <num>3</num>
                    </entry>
                    <entry>
                        <entityid>1</entityid>
                        <text>'a cheap small black
                            clock'</text>
                        <size>1</size>
                        <color>red</color>
                        <type>clock</type>
                        <price>1</price>
                        <cost>1</cost>
                        <pricerange>1</pricerange>
                        <num> 2</num>
                    </entry>
                    <entry>
                        <entityid>3</entityid>
                        <text>'a large green
                            clock'</text>
                        <size>3</size>
                        <color>green</color>
                        <type>clock</type>
                        <price>100</price>
                        <cost>40</cost>
                        <pricerange>2</pricerange>
                        <num>4</num>
                    </entry>
                    <entry>
                        <entityid>4</entityid>
                        <text>'a medium size blue
                            clock'</text>
                        <size>2</size>
                        <color>blue</color>
                        <type>clock</type>
                        <price>1000</price>

```

```

        <cost>500</cost>
        <pricerange>2</pricerange>
        <num>1</num>
    </entry>
    <entry>
        <entityid>5</entityid>
        <text>'a small knife'</text>
        <size>1</size>
        <color>black</color>
        <type>dagger</type>
        <price>10000</price>
        <cost>700</cost>
        <pricerange>3</pricerange>
        <material>steal</material>
        <num>2</num>
    </entry>
    <entry>
        <entityid>6</entityid>
        <text>'a large sword'</text>
        <size>3</size>
        <color>silver</color>
        <type>sword</type>
        <price>100000</price>
        <cost>90000</cost>
        <pricerange>3</pricerange>
        <material>steal</material>
        <num>1</num>
    </entry>
</dbQuery>
</content>
</invoke>
</state> <!-- DB -->
<state id="DB_search">
    <datamodel>
        <data name="EventNr"
            expr="1" eval="true" />
        <data name="Searches"
            expr="0" eval="true" />
    </datamodel>
    <onentry>
        <send target="dbq"
            event="extendQuery"
            expr="fun {$ 0} 0.num>=1 end"
            delay="100ms"/>
        <if cond="Type\=undefined" >
            <send target="dbq"
                event="extendQuery"
                expr="fun {$ 0} 0.type==Type end"
                delay="300ms"/>
            <assign name="Searches"
                expr="Searches+1"/>
        </if>
        <if cond="Color\=undefined">
            <send target="dbq"

```

```

        event="extendQuery"
        expr="fun {$ O} O.color==Color end"
        delay="500ms"/>
    <assign name="Searches"
        expr="Searches+1"/>
</if>
<if cond="Size\=undefined">
    <send target="dbq"
        event="extendQuery"
        expr="fun {$ O} O.size==Size end"
        delay="700ms"/>
    <assign name="Searches"
        expr="Searches+1"/>
</if>
<if cond="Pricerange\=undefined">
    <send target="dbq"
        event="extendQuery"
        expr="fun {$ O}
            O.pricerange==Pricerange end"
        delay="900ms"/>
    <assign name="Searches"
        expr="Searches+1"/>
</if>
</onentry>
<onexit>
    <send target="dbq" event="Cancel"/>
</onexit>
<transition event="dbq.answerset.size"
    cond="EventNr==Searches"
    target="select_properties">
    <log expr="'We have '#Eventdata#'
        different '#Type#'s'"/>
</transition>
<transition event="dbq.answerset.size"
    cond="EventNr\=Searches">
    <assign name="EventNr"
        expr="EventNr+1"/>
</transition>
<transition event="dbq.Done"
    cond="Eventdata\=o"
    target="present_selected_object">
    <assign name="Type"
        expr="Eventdata.type"/>
    <assign name="Color"
        expr="Eventdata.color"/>
    <assign name="OUD"
        expr="Eventdata.entityid"/>
    <assign name="Price"
        expr="Eventdata.price"/>
    <assign name="Size"
        expr="Eventdata.size"/>
    <assign name="Pricerange"
        expr="Eventdata.pricerange"/>
    <assign name="Cost"

```

```

        expr="Eventdata.cost"/>
    <assign name="Text"
        expr="Eventdata.text"/>
</transition>
<transition event="dbq.Done"
        cond="Eventdata==o">
    <log expr="'I am sorry, we just sold
        the last one.'"/>
    <assign name="OUD" expr="Nil"/>
    <assign name="Type" expr="Nil"/>
    <assign name="Color" expr="Nil"/>
    <assign name="Price" expr="Nil"/>
    <assign name="Pricerange" expr="Nil"/>
    <assign name="Size" expr="Nil"/>
    <assign name="RoundsLeft" expr="Nil"/>
</transition>
</state>      <!--DBsearch-->
</parallel> <!--search_object-->
</state>      <!--supp_obj_info-->
</state>      <!--define_ooi-->
<state id="negotiation">
    <initial>
        <transition target="smart_seller"/>
    </initial>
    <transition event="ca"
        cond="Eventdata.ca_type==accept"
        target="resolve_exchange">
        <log expr="Eventdata.string"/>
    </transition>
    <state id="smart_seller">
        <initial>
            <transition target="smartoffer"/>
        </initial>
        <onentry>
            <assign name="RoundsLeft"
                expr="{Rounds Price}"/>
            <assign name="Interval"
                expr="(Price-Cost) div RoundsLeft"/>
            <assign name="Price"
                expr="Price-Interval"/>
        </onentry>
    </state id="smartoffer">
        <onentry>
            <log label="system" expr="'Smart Seller
                offers the object of price '#Price'"/>
            <send target="Self" event="sysoffer"
                expr="o(agent:system price:Price)"/>
        </onentry>
        <transition event="ca"
            cond="Eventdata.ca_type==counteroffer
                andthen PreviousBid >= Eventdata.offer"
            target="smartoffer">
            <log label="Eventdata.agent"
                expr="Eventdata.string#'

```

```

        '#PreviousBid"/>
    <log expr="'user offers same or lower bid
        than previous'"/>
    <assign name="RoundsLeft"
        expr="RoundsLeft-1"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==counteroffer
    andthen 1>=RoundsLeft andthen
    Eventdata.offer>PreviousBid andthen
    Eventdata.offer>=Cost"
    target="accept_transaction">
    <log label="Eventdata.agent"
        expr="Eventdata.string#'
        '#Eventdata.offer"/>
    <assign name="Price"
        expr="Eventdata.offer"/>
    <log label="system" expr="'System accepts
        price: '#Price"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==counteroffer
    andthen Eventdata.offer>=Price-Interval"
    target="accept_transaction">
    <log label="Eventdata.agent"
        expr="Eventdata.string#'
        '#Eventdata.offer"/>
    <assign name="Price"
        expr="Eventdata.offer"/>
    <log label="system" expr="'System accepts
        price: '#Price"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==counteroffer
    andthen 1>=RoundsLeft andthen
        Cost>Eventdata.offer"
    target="reject_transaction">
    <log label="Eventdata.agent"
        expr="Eventdata.string#'
        '#PreviousBid"/>
</transition>
<transition event="ca"
    cond="Eventdata.ca_type==counteroffer
    andthen ((Price-Interval)>Eventdata.offer
    or else Cost>Eventdata.offer) "
    target="smartoffer">
    <assign name="PreviousBid"
        expr="Eventdata.offer"/>
    <log label="Eventdata.agent"
        expr="Eventdata.string#'
        '#PreviousBid"/>
    <assign name="Price" expr="{NextOffer
        Price PreviousBid RoundsLeft
        Interval}"/>

```

```

        <assign name="RoundsLeft"
              expr="RoundsLeft-1"/>
    </transition>
</state> <!-- smart_offer -->
<state id="accept_transaction">
    <onentry>
        <log label="system" expr="'Okey, we have a
            deal!'" />
    </onentry>
    <transition target="resolve_exchange"/>
</state> <!--accept_transaction-->
<state id="reject_transaction">
    <onentry>
        <log expr="'Deal is off!'" />
    </onentry>
    <transition target="ending"/>
</state> <!--reject_transaction-->
</state> <!-- smart_seller-->
</state>
<state id="resolve_exchange">
    <initial>
        <transition target="confirm_price"/>
    </initial>
    <state id="confirm_price">
        <onentry>
            <log expr="'That will be '#Price# ',
                please '" />
            <send target="Self" event="confirmprice"
                expr="o(agent:system price:Price)"/>
        </onentry>
        <transition event="handovermoney"
            target="moneytransaction"/>
    </state> <!--confirm_price-->
    <state id="moneytransaction">
        <initial>
            <transition target="thank"/>
        </initial>
        <state id="thank">
            <onentry>
                <log expr="'Thank you very much!'" />
            </onentry>
            <transition target="handovergoods"/>
        </state> <!--thank-->
        <state id="handovergoods">
            <onentry>
                <log expr="'And your goods..
                    (shopkeeper hands over the goods)'" />
                <send target="Self"
                    event="handovergoods"/>
            </onentry>
            <transition event="usertakesgoods"
                target="ending"/>
        </state> <!--handovergoods-->
    </state> <!--moneytransaction-->

```

```

    </state> <!--resolvexchange-->
</state> <!--trading-->
<state id="ending">
  <onentry>
    <log expr="'Welcome back!'" />
  </onentry>
  <transition target="exit" />
</state> <!--ending-->
</state> <!--system-->
<state id="user">
  <datamodel>
    <data name="ReqType" />
    <data name="ReqColor" />
    <data name="ReqSize" />
    <data name="ReqPricerange" />
    <data name="ReqPrice" />
    <data name="Money" expr="500" />
    <data name="Bid" />
    <data name="Raise" />
    <data name="UserRoundsLeft" />
    <data name="UserRounds" eval="true">
      fun {$ P}
        if 15>=P then ({OS.rand()} mod 5) + 1
        else ({OS.rand()} mod 10) + 1 end
      end
    </data>
  </datamodel>
  <onentry>
    <send target="Self" event="ca"
      expr="o(ca_type:greeting string:'hi')"
      delay="200ms" />
  </onentry>
  <transition event="greet" >
    <send target="Self" event="ca"
      expr="o(agent:user ca_type:request_object
        type:clock string:'I would like to
        buy a clock' grounding:'a clock')"/>
    <assign name="ReqType" expr="clock" />
    <log expr="Eventdata.string" />
  </transition>
  <transition event="asktype">
    <send target="Self" event="ca"
      expr="o(agent:user ca_type:request_object
        type:clock string:'a clock, please'
        grounding:'a clock')"/>
    <assign name="ReqType" expr="clock" />
  </transition>
  <transition event="askcolor">
    <send target="Self" event="ca"
      expr="o(agent:user ca_type:request_object
        color:green string:'Do you have any
        green one?' grounding:'green')"/>
    <assign name="ReqColor" expr="green" />
  </transition>

```



```

<transition event="asksize">
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:request_object
      size:1 string:'a small one'
      grounding:'small')"/>
  <assign name="ReqSize" expr="1"/>
</transition>
<transition event="askpricerange">
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:request_object
      pricerange:1 string:'the cheapest'
      grounding:'a cheap one')"/>
  <assign name="ReqPricerange" expr="clock"/>
</transition>
<transition event="present_object">
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:request_price
      string:'How much is it?')"/>
</transition>
<transition event="present_price">
  <assign name="Bid" expr="Eventdata.price div 2"/>
  <assign name="UserRoundsLeft" expr="{UserRounds
    Eventdata.price}"/>
  <assign name="Raise" expr="(Eventdata.price-Bid)
    div UserRoundsLeft"/>
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:counteroffer
      offer:Bid string:'No, that is too
      expensive! I can pay you ')/>
</transition>
<transition event="sysoffer"
  cond="1>=UserRoundsLeft andthen
    Money>Eventdata.price">
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:accept
      offer:Eventdata.price string:'I will take
      it')"/>
  <log expr="'Okey, I take it, I do not have time
    to argue about the price any more...'">
</transition>
<transition event="sysoffer"
  cond="(Bid+Raise)>=Eventdata.price
    andthen Money>=Eventdata.price" >
  <send target="Self" event="ca"
    expr="o(agent:user ca_type:accept
      offer:Eventdata.price string:'I take it!')"/>
  <log expr="That is a fair price, I will take
    it!'">
</transition>
<transition event="sysoffer" cond="1>=UserRoundsLeft
  andthen Eventdata.price>=Money">
  <send target="Self" event="ca"
    expr="o(ca_type:cancel_transaction offer:Bid
      agent:user string:'Thanks, but no thanks, I

```

```

        think I will move on...')"/>
</transition>
<transition event="sysoffer"
            cond="Eventdata.price>=(Bid+Raise)">
  <assign name="Bid" expr="Bid+Raise"/>
  <assign name="UserRoundsLeft"
            expr="UserRoundsLeft-1"/>
  <send target="Self" event="ca"
            expr="o(ca_type:counteroffer agent:user
                    offer:Bid string:'I can pay you ')">
</transition>
<transition event="confirmprice">
  <log expr="'Okey, here you are'"/>
  <send target="Self" event="handovermoney"/>
</transition>
<transition event="handovergoods">
  <send target="Self" event="usertakesgoods"/>
</transition>
</state>
</parallel>
<final id="exit"/>
</scxml>

```

Appendix IV: Movie-gossip

```
<?xml version="1.0"?>
<!DOCTYPE scxml SYSTEM "assets/dtd/scxml.dtd">
<scxml initial="init"
xmlns="http://www.w3.org/2005/07/scxml">
  <invoke targettype="x-sitepal-agent" ID="agent">
    <param name="agent" expr="1608122"/>
    <param name="height" expr="600"/>
    <param name="width" expr="500"/>
  </invoke>
  <invoke ID="parser" targettype="x-srsgs-parser"/>
  <state id="init">
    <transition event="ready" target="npc"/>
  </state>
  <datamodel>
    <data ID="numofE"/>
    <data ID="iterations" expr="0"/>
    <data ID="utterance"/>
    <data ID="threat" expr="3"/>
    <data ID="dialogue" expr="0"/>
    <data ID="persons">
      [{name:'Paul', gen:'m'}, {name:'Susan', gen:'f'},
       {name:'Jimmy', gen:'m'}, {name:'Elaine', gen:'f'}]
    </data>
    <data ID="topics">
      [{init:'I am glad Paul is moving out',
        provideE:["he is just always giving me the creeps.
        Haven't you noticed?", "He has this dark thing
        going on", "There is something about him that
        feels malignant"],
        wrapUp:'that being said, I do love what he has
        done with the lawn'},
       {init:'I hate Susan Mayer. Everytime I see those
        big doe eyes of hers. I swear to God, I just wanna
        go out and shoot a deer',
        provideE:['She is throwing herself at Mike Delfino.
        Again.', 'She has been lusting after him ever
        since he moved in'],
        wrapUp:'Got to go. Bye!'}]
    </data>
  </datamodel>
  <parallel id="npc">
    <state id="presentationLayer">
      <invoke ID="conversation"
        targettype="x-vxml-interpreter">
        <content>
          <vxml version="2.0"
            xmlns="http://www.w3.org/2001/vxml">
            <form id="gossip">
              <grammar version="1.0" root="start">
```

```

<rule id="start">
  <one-of>
    <item><ruleref uri="#yn"/></item>
    <item> <ruleref uri="#seekE"/>
    </item>
  </one-of>
  <tag>out.ca=ca</tag>
</rule>
<rule id="yn">
  <one-of>
    <item> yes <tag>
      ca='agree'</tag></item>
    <item> yeah <tag>
      ca='agree'</tag></item>
    <item> I totally agree <tag>
      ca='agree'</tag></item>
    <item> me too <tag>
      ca='agree'</tag></item>
    <item> absolutely <tag>
      ca='agree'</tag></item>
    <item> no <tag>
      ca='disAgree'</tag></item>
    <item> I do not agree <tag>
      ca='disAgree'</tag></item>
    <item> I do not think so <tag>
      ca='disAgree'</tag></item>
  </one-of>
</rule>
<rule id="seekE">
  <one-of>
    <item>ok<tag>
      ca='reqEvidence'</tag></item>
    <item>really<tag>
      ca='reqEvidence'</tag></item>
    <item> why <tag>
      ca='reqEvidence' </tag></item>
    <item> why do you say that <tag>
      ca='reqEvidence' </tag></item>
    <item> how so <tag>
      ca='reqEvidence'</tag></item>
    <item> how come <tag>
      ca='reqEvidence'</tag></item>
    <item> tell me more <tag>
      ca='reqEvidence'</tag></item>
  </one-of>
</rule>
</grammar>
<initial name="topics"/>
<field name="ca"/>
<field name="garbage"/>
<nomatch count="1">
  <prompt> Sorry? </prompt>
</nomatch>
<nomatch count="2">

```

```

        <prompt> What did you say?</prompt>
    </nomatch>
    <nomatch count="3">
        <prompt> Well well</prompt>
    </nomatch>
    <nomatch count="3">
        <prompt> ok </prompt>
    </nomatch>
    </form>
</vxml>
</content>
</invoke>
<transition event="utterance">
    <forward target="'conversation'"/>
</transition>
<transition event="recognize">
    <forward target="'parser'"/>
</transition>
<transition event="recResult">
    <forward target="'conversation'"/>
</transition>
<transition event="prompt">
    <send target="'agent'" event="sayText">
        {"txt":dm.event.data,"voice":1,"lang":1,"engine":2}
    </send>
</transition>
<transition event="filled">
    <log expr="dm.event.data.result"/>
</transition>
</state>
<state id="dialogueManager">
    <initial>
        <transition target="Idle"/>
    </initial>
    <state id="Idle">
        <transition target="Initiate"/>
    </state>
    <state id="Initiate">
        <initial>
            <transition target="Opinion"/>
        </initial>
        <state id="Opinion">
            <onentry>
                <assign name="dm.utterance"
                    expr="eval('dm.topics[' +
                        dm.dialogue + '].init')"/>
                <send target="'agent'"
                    event="sayText">
                    {"txt":dm.utterance,"voice":1,"lang":1,"engine":2}
                </send>
            </onentry>
            <transition event="filled"
                cond="dm.event.data.result.ca=='disAgree'
                    && dm.threat>4" target="WrapUp"/>
        </state>
    </state>
</state>

```

```

<transition event="filled"
  cond="dm.event.data.result.ca=='agree'"
  target="WrapUp"/>
<transition event="filled"
  cond="dm.event.data.result.ca=='reqEvidence'
  || dm.event.data.result.ca=='disAgree'"
  target="ProvideEvidence">
  <assign name="dm.numofE"
    expr="eval('dm.topics[' + dm.dialogue +
      '].provideE.length')"/>
</transition>
</state>
<state id="ProvideEvidence" timeout="'5s'">
  <onentry>
    <if cond="dm.iterations <= dm.numofE">
      <assign name="dm.utterance"
        expr="eval('dm.topics[' + dm.dialogue +
          '].provideE[' + dm.iterations + ']')"/>
      <send target="'agent'"
        event="sayText">
{"txt":dm.utterance,"voice":1,"lang":1,"engine":2}
      </send>
    </if>
  </onentry>
  <transition cond="dm.iterations > dm.numofE"
    target="WrapUp"/>
  <transition event="filled"
    cond="dm.event.data.result.ca=='reqEvidence' ||
    dm.event.data.result.ca=='disAgree'"
    target="ProvideEvidence"/>
  <transition event="filled"
    cond="dm.event.data.result.ca=='agree'"
    target="WrapUp"/>
  <transition event="ProvideEvidence.timeout"
    target="ProvideEvidence"/>
  <onexit>
    <assign name="dm.iterations"
      expr="dm.iterations + 1"/>
  </onexit>
</state>
<state id="WrapUp">
  <onentry>
    <assign name="dm.utterance"
      expr="eval('dm.topics[' +
        dm.dialogue + '].wrapUp')"/>
    <send target="'agent'" event="sayText">
{"txt":dm.utterance,"voice":1,"lang":1,"engine":2}
    </send>
  </onentry>
  <transition event="promptEnded" target="Idle"/>
  <onexit>
    <if cond="dm.dialogue > dm.topics.length">
      <assign name="dm.dialogue" expr="0"/>
    <else/>

```

```
        <assign name="dm.dialogue"  
              expr="dm.dialogue + 1"/>  
        <assign name="dm.iterations"  
              expr="0"/>  
      </if>  
    </onexit>  
  </state>  
</state>  
</parallel>  
</scxml>
```