

DEAL – Dialogue Management in SCXML for Believable Game Characters

Jenny Brusk
Dept of Game Design, Narrative
and Time-based Media,
Gotland University, Sweden
jenny.brusk@hgo.se

Torbjörn Lager
Dept of Linguistics
Göteborg University, Sweden
lager@ling.gu.se

Anna Hjalmarsson Preben Wik
Centre for Speech Technology
KTH Stockholm, Sweden
{annah, preben}@speech.kth.se

ABSTRACT

In order for game characters to be believable, they must appear to possess qualities such as emotions, the ability to learn and adapt as well as being able to communicate in natural language. With this paper we aim to contribute to the development of believable non-player characters (NPCs) in games, by presenting a method for managing NPC dialogues. We have selected the trade scenario as an example setting since it offers a well-known and limited domain common in games that support ownership, such as role-playing games. We have developed a dialogue manager in State Chart XML, a newly introduced W3C standard¹, as part of DEAL – a research platform for exploring the challenges and potential benefits of combining elements from computer games, dialogue systems and language learning.

Categories and Subject Descriptors

I.2.7 [Artificial Intelligence]: Natural Language Processing – *discourse, language generation, language parsing and understanding, speech recognition and synthesis.*

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence – *coherence and coordination, intelligent agents, languages and structures, multiagent systems.*

General Terms

Algorithms, Design, Languages.

Keywords

Game dialogue, non-player characters, statecharts, serious games, SCXML.

1. INTRODUCTION

As games become more and more sophisticated in terms of graphical and technological capabilities, higher demands are also put on the content provided by them. With content we mean the story, characters, dialogue, gameplay, music, sound etc., i.e. everything that is presented to the player through the interface. We judge the game and the gaming experience in terms of how well the components in the game are composed, how they encourage the player to take certain actions and the extent to which they motivate the player to stay in the game, i.e. the level of immersiveness the game provide. Given that characters that can be encountered in games offer both possibilities of opposition to

players and vehicles for narratives, they are a natural focus for improving the content in games.

Although not primarily focusing on games, a lot of research has been conducted within the field of believable characters, in the sense "characters that provide the illusion of life" [3]. These characters possess qualities such as emotions, personality, the ability to communicate in natural language, adaptive behavior and learning capabilities [7, 8, 9, 12, 19] – qualities usually ascribed to humans. Believable characters tend to appear in animated and/or interactive dramas, or as interactive (conversational) agents, but also in simulations such as the Mission Rehearsal Exercise project [19].

In games, the characters need interactive abilities, such as self-impelled actions, expression of emotions, and self-awareness [16]. It is also important that there is an interplay between the different properties of a character as well as between the character and the game world (including events and actions caused by the player or other characters): For instance, a character that experiences an emotional moment, should be able to express these emotions verbally as well as non-verbally, preferably in accordance with her personality. To avoid repetitious responses, the character needs to have a range of possible responses to choose from, perhaps based on an emotional memory, the interpersonal relationship and the dialogue history. Lankoski & Björk [16] mean that in order for the dialogues to be believable the characters must be able to produce *contextualized conversational responses*, i.e. the character needs to take the current game state into account when formulating a response. A similar view is presented in [5] saying that designers need to consider dialogue flow a part of the game flow, treat dialogue actions on par with other game actions, and think of the current game state as comprising also the state(s) of the dialogue(s) taking place at the current point in time. We believe that the most successful natural language enabled games will treat natural language dialogue as an integral part of the game, rather than something added on as an afterthought.

With these views in consideration, we wish to contribute to the creation of a more realistic and immersive dialogue with non-player characters (NPCs) in games.

2. DIALOGUE SYSTEMS

Dialogue systems are programs that can communicate with humans in natural language [15]. The system takes free-form text or speech input and transforms it to a representation that can be understood by the system. The system must then produce an appropriate response to the input. How this is done depends on the purpose and the complexity of the system.

¹ <http://www.w3.org/TR/scxml/>

Dialogue systems have mainly been designed to collaborate with the user to solve a particular task, such as managing bank transactions or handle time-table tasks, such as *The Phillips Automatic Train Timetable Information System* [2], i.e. to handle dialogues that Allen et al [1] refer to as *practical*. These systems usually operate in a specific and simple domain, which enables them to perform robustly. The complexity of the system may vary from the simplest finite-state based systems, to the most complex agent-based models. (see e.g. [1] for an excellent overview).

Game dialogue systems, on the other hand, usually refer to the tree-structure that constitutes player (character)-NPC interaction. They appear to the player as a dialogue menu displayed in a static interface, as in *Grim Fandango*² and *Fable*³. In some games the dialogue system is more like an information retrieval system, such as in *The Elder Scrolls III: Morrowind*⁴, where questions are posed by clicking hyperlinked key-words connected to a database. Game dialogue systems thus rarely allow free-form input from the players, instead they present the different available options to the player as a list of phrases or topics.

2.1 Practical Dialogue Systems vs Game Dialogue Systems

We argue that there are some critical differences between the requirements of a practical dialogue system as opposed to a dialogue system designed for games in terms of:

Error handling – Any system must of course be able to handle upcoming errors, such as misunderstanding or non-understandings, but the strategies may differ. When a practical dialogue system aims at reaching a level of understanding in the dialogue that will lead to a successful completion of the task, a game character in dialogue with the player may behave irrationally or emotionally instead. It can pretend to have understood, change subject or perhaps just end the dialogue abruptly.

Correctness and cooperativeness – the information provided by the practical dialogue system should be correct and relevant, whereas the outcome of a game dialogue may depend on the character providing it, the current situation and the interpersonal relationship. We can think of situations where the NPC provide the player with false information or withhold information until a relationship has been established or a certain game state has been reached.

Predictability and reliability – The purpose of interacting with a practical dialogue system is to be provided with a certain service. It is therefore important that the system behaves in a consistent manner and has a predictable outcome. In games, on the other hand, the believability of the character may clash with the predictability of the dialogue system.

Motivation – The main motivation for using a practical dialogue system is to successfully perform a certain task or acquire information from it. In a game, there can be several reasons for the player to engage in a dialogue: To socialize, get game hints or

quests, or just get entertained. A game dialogue system should therefore be designed to handle longer exchanges as well as a variety of relevant topics. In the DEAL-project for instance (presented below), one of the goals is to create a game dialogue system for language learning. In this case, the talking in itself is the most important goal of the interaction, not to complete the task.

3. NATURAL LANGUAGE INTERACTION IN GAMES

Many different approaches to introducing natural language in games have been applied. One of the first ways natural language was used in games was as typed commands meant to serve the same purpose as direct manipulation of a graphical user interface. We have seen examples in the old text-based adventure games, such as the *Zork-series*⁵, where the user could type in text-based commands expressed by words or phrases that could be parsed and understood by the system.

Today, players can control the game using voice in addition to the mouse and keyboard. The new technology use voice-over IP to enable the player to communicate with the game system as well as with other players in multiplayer online games. This way, the player can do more tasks simultaneously, such as fighting a monster while talking to co-players. Hence, these new possibilities also support socialization, coordination and collaboration in multiplayer games.

To summarize, 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), player character (PC) and non-player character (NPC), and stretching the notion of dialogue somewhat, we may distinguish between:

- P in dialogue with G: Games may be ‘voice controlled’. Instead of hitting the pause button in order to pause a game, the player may just say “pause”.
- P in dialogue with PC: Player is directing his player character using dialogue.
- P in dialogue with P: Player talking to player, using (voice-based) chat.
- 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 200X*.
- 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.

Conversations between humans are often of the multimodal kind, and realistic dialogue with NPCs should therefore be too. 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

² Lucas Arts (1998), for PC

³ Lionhead Studios Ltd (2004), Microsoft, for Xbox

⁴ Bethesda Softworks (2002), for PC

⁵ Infocom (1980-1982)

and what it does – and its natural language capabilities – what it says – is not very clearcut. Thus, these things should better be controlled and synchronized using one and the same mechanism.

4. STATECHART XML (SCXML)

SCXML can be described as an attempt to render Harel statecharts [11] in XML. Harel developed his statecharts as a graphical notation for specifying reactive systems in great detail. In its simplest form, a statechart is just a finite state machine, where state transitions are triggered by events appearing in an event queue.

Any statechart can be translated into a document written in the linear XML-based syntax of SCXML (see [5] for examples).

Harel also introduced a number of (at the time) novel extensions to finite-state machines, which are also present in SCXML, including:

- **Hierarchy** – a state may contain another statechart down to an arbitrary depth. Hence reducing the number of states.
- **History** – a memory of which state the superstate was in when it was left for another state.
- **Concurrency** – several 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. Concurrency may for example be useful 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 the game world.
- **Broadcast communication** – a statechart may send an event to the global event queue that will trigger a transition in a concurrent statechart. This way, two characters in the game modelled as concurrent statecharts can communicate with each other.
- **Datamodel** (a.k.a. “extended state variables”) – within a statechart we can use datamodels to store and update (XML) data. In the trade example, explained below (see figure 1), we store the data retrieved from the database search in a datamodel associated with the shopkeeper's statechart, allowing us to access and manipulate the data in any state.

Important to note is that SCXML is not intended to interact with a user directly. Rather, it needs a presentation layer, which may support different types of modalities.

5. DEAL

The work (in progress) presented here is part of DEAL, a project investigating the possibilities to create a language learning system for conversational training that uses gameplay elements to create an immersive learning environment. We are talking about a serious game, i.e. a game that has a purpose other than to solely entertain, such as teaching, training or advertising [13].

DEAL sets the scene of a flea market where a talking animated agent (a non-player 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. The flea market is also a place where it is acceptable to negotiate the price.

DEAL is implemented by using components from the Higgins project (see e.g. [18]): 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).

The game dialogue manager (DM) in DEAL presented in this paper is implemented using SCXML for the following reasons:

- It is coupled with the visual statechart representation (makes the system easy to manage and overview)
- It is finite-state based (Most games are finite-state based)
- It support concurrency and hierarchy
- It is XML-based (many game engines communicate with XML data)
- It can invoke different presentation layers (chat, voice, multimodal) depending on the task.

5.1 Implementation

The SCXML application has been implemented and tested in the SCXML Web Laboratory⁶, a web-based interface for SCXML-applications written in Oz, developed by Torbjörn Lager at the Department of Linguistics at Göteborg University.

The Higgins project aims at developing “a collaborative dialogue system in which error handling can be tested empirically” [6]. Higgins is a module-based system consisting of an interpreter, *Pickering*, that takes the result from the ASR as input and creates a semantic representation of the user's communicative act (CA). The CA is then sent to the discourse modeller, *Galatea*. The output from *Galatea* constitutes the input to an action manager, in this case tailored for the DEAL domain [18, 21].

The SCXML dialogue manager described in this paper has been developed separately, 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 eventdata as formalised in the SCXML Web Lab.

We regard a communicative act produced by the user as an incoming event after which the system sends a response in the form of a new communicative act, that (in this case) is hypothetically passed to the generator in Higgins and then back to *Galatea* again.

6. TRADE

As a tool for describing games and gameplay features, Björk & Holopainen [4] have introduced the concept of gameplay design patterns, defined as “...semiformal interdependent descriptions of

⁶ <<http://www.ling.gu.se/~lager/Labs/SCXML-Lab/>>

commonly reoccurring parts of the design of a game that concern gameplay". The idea is that games that feature a certain gameplay design pattern most likely (or perhaps should) feature other related patterns as well. Trade is here described as "*Players exchange some kind of Resource, be it information, actions, or game elements, between each other or the game system*" and is related to patterns such as *Resources*, *Ownership* and *Social Interaction*.

A shopkeeper in a game is typically represented by an NPC, and a trade dialogue between the shopkeeper and a player may be both practical (to successfully complete a transaction) as well as socially oriented, involving elements such as negotiation. Trade is thus a good candidate for exemplifying how natural language dialogue can be used in a game setting.

There are also other 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 is interesting from a language acquisition point of view
- trade may involve bargaining. The negotiation process is in itself an interesting research area due to its complexity containing both rational and emotional elements.

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.

7. TRADE IN SCXML

A trade can be modelled as consisting of three phases: An opening phase, in which the participants acknowledge each other, a middle phase, when the actual transaction takes place – including an optional bargaining phase, and finally an end phase, in which the deal is closed and the participants exit the interaction. We have used this structure as a starting point when modelling the statechart.

The trade statechart (see figure 1.) consists of a top-level state (`shop_keeper`), which represents the shopkeeper's statechart of the dialogue. The dialogue state then consists of three substates, representing each phase in the interaction: `opening`, `trading` and `ending`.

An example of a trade dialogue can be as follows (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!

When the player utters P1, the shopkeeper's active state is `greeting`. After greeting the shopkeeper, the player immediately expresses a request, triggering a transition to `search_object`. On entering this state, the shopkeeper searches the inventory and finds several matches, causing him to ask for additional information (S2 and S3), thus triggering a transition to the subsequent superstate `get_info`. When the shopkeeper retrieves one unique match in the database, the state `present_object` is activated, in which the shopkeeper presents the item to the player (S4). Having agreed upon the item to trade, a negotiation process may begin, in this case initiated by the player in P5. When this happens, a transition to the state `negotiation` is taking place. After a period of offers and counteroffers, the two participants come to an agreement (S7 followed by P7) causing a transition to `resolve_exchange`. The price is confirmed by the shopkeeper (S8), who then waits until he has received the money from the player.

Compared to how trade is usually conducted in games, the NPC in this example is able to take a more active part in the interaction, allowing the player to take initiative (as in the initial request), but also taking the 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 gives 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 trade is cancelled as a whole.

With statecharts we can easily add transitions for new types of events, as well as add states for new situations that may occur. The complexity of the given example lies in the negotiation process, where a variety of parameters determines the outcome, but also in the way initiative is handled.

Below, we will give a detailed description of the different states, what they do and how they work.

7.1 Opening

Either participant may initiate the dialogue, by for instance a "greeting" event, but the player may also choose to immediately send out a "request", which is an event that will trigger a transition to the trading state. If the player is passive ("no-input"), the system may attempt to receive an input by offering assistance (a transition triggered by a "time-out" event). The system may

also fail to recognize or understand the input, which will result in a "no-match" event.

7.2 Trading

As expected, the actual trading phase is the non-trivial part of the dialogue. The complex state trading contains three substates: `define_ooi`, `negotiation` and `resolve_exchange`, all of which are complex states themselves. A trade can only be conducted if the shopkeeper owns a specific item that the buyer is interested in purchasing and if the buyer has something to trade in return (in this case money).

Define object of interest. Before anything else, the shopkeeper must find out what the buyer wants to purchase. This complex state `define_ooi` includes states for gathering information about the requested object (`get_info`), searching for objects in the database (`search_object`) and a state for presenting an object when there is a match in the database (`present_object`). We have used inspiration from form-based dialogue systems, such as VoiceXML⁷, in collecting the necessary information from the player, i.e. 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`). A more sophisticated system would of course try to ask only relevant questions, i.e. questions of distinctive features. The substate `search_object` does the actual database search based upon the information retrieved in `get_info`.

Negotiation. Negotiation is a complex process involving elements of social psychology and game theory and can also include emotional elements and lies. Our implementation so far handles offers and counteroffers. The next step will be to attempt to add complexity in terms of argumentation-based negotiation.

Negotiation can be viewed as a distributed search through a space of potential agreements [14, p6]. 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 or 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 [14] and the "automated bargaining game" presented in [20]. 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 SCXML code in appendix I). The buyer will also make offers according to those parameters, but the number of rounds will most likely be different, since the number of rounds is randomly selected for each agent. Neither agent has information about the other's time and cost/utility. For each round, the agents will increase their eagerness to conclude the deal, we assume neither of them is interested in reaching a breakdown. 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.

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`, 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.

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.

8. RELATED WORK

Initially, we discussed our work in relation to the research on believable characters and practical dialogue systems. Another area that is highly relevant in the context is the ongoing research around interactive stories, such as Façade⁸, further described in [17] and The Interactive Story Project⁹ (IS). The idea with these projects is to allow the user to influence how the story progresses by intervening the play using text (in Façade) or voice (IS). In Façade the player is playing a role in the drama, whereas in IS, the user is regarded as an "active spectator".

Other research projects have used games as environment, for instance The NICE Fairy-tale Game System [10], where the player manipulates the game world through the helper character Cloddy Hans using voice. Similarly, Zubeck [22] presents two examples of game dialogues; one where the player interacts with a shopkeeper, and one simulating a break-up of a relationship between the player and a virtual character.

9. DISCUSSION

A believable character has been described as being "lifelike", possessing human qualities such as emotions, a visual appearance and the ability to use natural language. One way to look at it is to say that a believable character needs a range of different traits expressed by different means and that these traits and expressions in fact are separate from each other, but yet interacting and combined in a multitude of ways. Statecharts allow us to model these traits independently and the behaviour of the character will then be determined by the combination of the active states representing the traits. Hence, this method can be used to create characters with emergent behaviour.

We have in this paper given an example of how statecharts can be used to model the dialogue manager for an NPC in a game. The example is derived from games designed with the patterns associated with ownership, as described by Björk & Holopainen [4], where shops can be used as an arena for trading objects. The NPC has been given the role of a shopkeeper capable of presenting objects of interest as well as negotiating the price. The dialogue allows mixed-initiative interaction, meaning that either

⁷ <<http://www.w3.org/TR/voicexml20/>>

⁸ <<http://www.interactivestory.net/>>

⁹ < <http://www-scm.tees.ac.uk/users/f.charles/>>

participant can initiate and control the dialogue. This is a work-in-progress, we have not yet used statecharts and SCXML to its full potential. The next step will therefore be to add the ability to use emotional elements in the interaction and add argumentation-based negotiation, which we hope will contribute to a believable and immersive gaming experience.

Important to note, however, we are not claiming that all dialogues should be conducted using natural language or that NPCs in general need to be modelled according to the principles associated with believable characters. We are here presenting one of many possible ways to deal with dialogues to invoke a certain type of gaming experience, where it enhances gameplay, it is not aimed to be a general principle for designing game dialogues.

10. ACKNOWLEDGEMENT

This work has been conducted within The Graduate School of Language Technology. The authors would like to thank Rolf Carlsson, Jens Edlund, Gabriel Skantze, Staffan Björk and Mirjam P. Eladhari.

11. REFERENCES

- [1] Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L. & Amanda Stent. (2001) Towards conversational human-computer interaction. *AI Magazine*, 22(4): 2737.
- [2] Aust, H., Oerder, M., Seide, F. & Steinbiss, V. (1995) The phillips automatic train timetable information system. *Speech Communication*, 17:249_262.
- [3] Bates, J. (1994) The role of emotion in believable agents. *Communications of the ACM*, 37(7): 122125, July.
- [4] Björk, S. & Holopainen, J. (2004) *Patterns in Game Design*. Charles River Media. ISBN1-58450-354-8.
- [5] Brusk, J. & Lager, T. (2007) Developing Natural Language Enabled Games in (Extended) SCXML. *Proceedings from the International Symposium on Intelligence Techniques in Computer Games and Simulations (Pre-GAMEON-ASIA and Pre-ASTEC)*, Shiga, Japan, March 1-3.
- [6] Edlund, J., Skantze, G., & Carlson, R. (2004). Higgins – a spoken dialogue system for investigating error handling techniques. In *Proceedings of ICSLP 2004*, 229-231.
- [7] Egges, A., Zhang, X., Kshirsagar, S. & Magnenat-Thalmann, N. (2003) *Emotional communication with virtual humans*. <<http://www.miralab.unige.ch/papers/161.pdf>>
- [8] Egges, A., Kshirsagar, S., & N. Magnenat-Thalmann (2004) Generic personality and emotion simulation for conversational agents. *Computer Animation and Virtual World*, 15:113, January. <<http://www.miralab.unige.ch/papers/81.pdf>>
- [9] Gratch, J., Rickel, J., André, E., Cassell, J., Petajan, E. & Badler, N. (2002) *Creating interactive virtual humans: Some assembly required*. Technical report, Workshop report, IEEE Intelligent Systems. <<http://www.cis.upenn.edu/badler/papers/x4GEW.pdf>>
- [10] Gustafson, J. Boye, J., Fredriksson, M., Johanneson, L. & Königsmann, J. (2005) Providing computer game characters with conversational abilities. In *Proceedings of Intelligent Virtual Agent (IVA05)*, Kos, Greece.
- [11] Harel, David (1987) Statecharts: A Visual Formalism for Complex Systems, In *Science of Computer Programming 8*, North-Holland.
- [12] Hayes-Roth, B. & Doyle, P. (1998) Animate characters. *Autonomous Agents and Multi-Agent Systems*, 1(2):195230. ISSN 1387-2532. <<http://dx.doi.org/10.1023/A:101001981877>>
- [13] Iuppa, N., & Borst, T. (2007). *Story and simulations for serious games : tales from the trenches*. Focal Press.
- [14] Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C. & Wooldridge, M. (2000) Automated Negotiation: Prospects, Methods and Challenges. GDN2000 Keynote Paper. *Int. Journal of Group Decision and Negotiation*
- [15] Jurafsky, D. & Martin, J. H. (2001) *Speech and Language Processing*. Prentice Hall: Upper Saddle River, New Jersey.
- [16] Lankoski, P. & Björk, S. (2007) Gameplay Design Patterns for Believable Non-Player Characters. In *Situated Play, Proceedings of Digra 2007*, University of Tokyo, Japan.
- [17] Mateas, M. & Stern, A. (2003) *Façade: An Experiment in Building a Fully-Realized Interactive Drama*. Game Developers Conference, Game Design track, March.
- [18] Skantze, G. (2005) GALATEA: A Discourse Modeller Supporting Concept-level Error Handling in Spoken Dialogue Systems. In *Proceedings of SigDial (pp. 178-189)*. Lisbon, Portugal.
- [19] Swartout, W., Gratch, J., Hill, R., Hovy, E., Marsella, S., Rickel, J. & Traum, D. (2004) *Toward virtual humans*. Working notes of the AAAI Fall symposium on Achieving Human Level Intelligence through Integrated Systems and Research.
- [20] Tsang, E. & Gosling, T. (2002) Simple Constrained Bargaining Game. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2002)*, Bologna, Italy, July15-19. <<http://lia.deis.unibo.it/aamas2002>>
- [21] Wik, P., Hjalmarsson, A. & Brusk, J. (2007) DEAL: A Serious Game for CALL Practicing Conversational Skills in the Trade Domain. In *Proceedings of SlaTE – Workshop on Speech and Language Technology in Education*, Pennsylvania, USA.
- [22] Zubek, R. (2005) Hierarchical Parallel Markov Models for Interactive Social Agents. PhD Dissertation, Computer Science Department, Northwestern University. Tech Report NWU-CS-05-10. <<http://robert.zubek.net/publications/dissertation.pdf>>

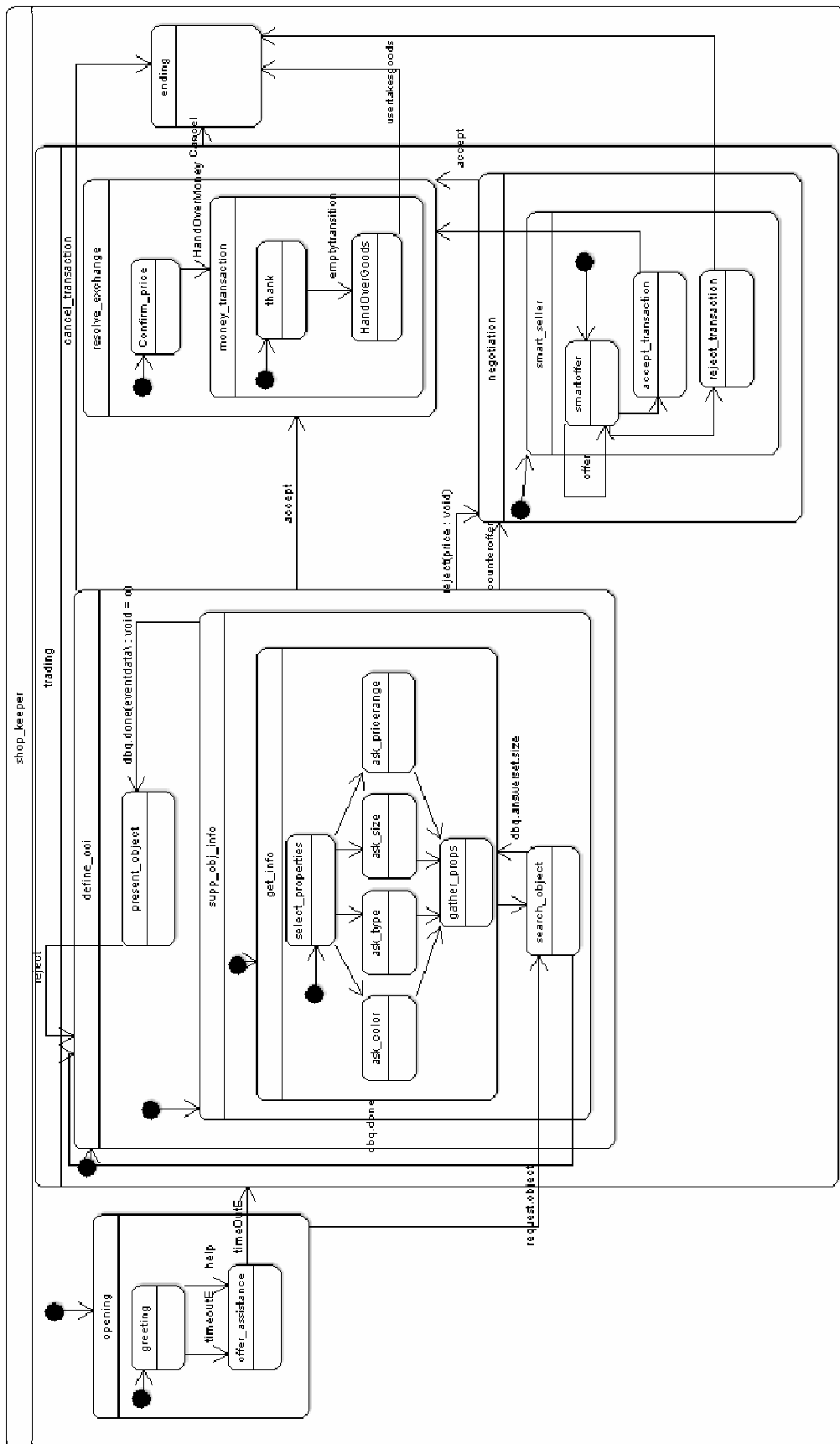


Figure 1. Trade statechart

APPENDIX I – SCXML code example – “smart seller” in negotiation state

```
<state id="smart_seller" target="smartoffer">
  <onentry>
    <assign name="RoundsLeft"
      expr="{Rounds Price}"/>
    <assign name="Interval"
      expr="(Price-Minprice) div RoundsLeft"/>
    <assign name="Price"
      expr="Price-Interval"/>
  </onentry>
  <state id="smartoffer">
    <onentry>
      <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#' kronor'"/>
      <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>=Minprice"
      target="accept_transaction">
      <log label="Eventdata.agent" expr="Eventdata.string#' '#Eventdata.offer#' kronor'"/>
      <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#' kronor'"/>
      <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 Minprice>Eventdata.offer"
      target="reject_transaction">
      <log label="Eventdata.agent" expr="Eventdata.string#' '#PreviousBid#' kronor'"/>
    </transition>
    <transition event="ca"
      cond="Eventdata.ca_type==counteroffer andthen ((Price-Interval)>Eventdata.offer
        or else Minprice>Eventdata.offer)" target="smartoffer">
      <assign name="PreviousBid" expr="Eventdata.offer"/>
      <log label="Eventdata.agent" expr="Eventdata.string#' '#PreviousBid#' kronor'"/>
      <assign name="Price" expr="{NextOffer Price PreviousBid RoundsLeft Interval}"/>
      <assign name="RoundsLeft" expr="RoundsLeft-1"/>
    </transition>
  </state>
</state>
```