Multidimensional Dialogue Management

Simon Keizer and Harry Bunt
Department of Language and Information Science
Faculty of Arts, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
{s.keizer,harry.bunt}@uvt.nl

Abstract

In this paper we present an approach to dialogue management that supports the generation of multifunctional utterances. It is based on the multidimensional dialogue act taxonomy and associated context model as developed in Dynamic Interpretation Theory (DIT). The multidimensional organisation of the taxonomy reflects that there are various aspects that dialogue participants have to deal with simultaneously during a dialogue. Besides performing some underlying task, a participant also has to pay attention to various aspects of the communication process itself, including social conventions.

Therefore, a multi-agent approach is proposed, in which for each of the dimensions in the taxonomy a specialised dialogue act agent is designed, dedicated to the generation of dialogue acts from that particular dimension. These dialogue act agents operate in parallel on the information state of the system. For a simplified version of the taxonomy, a dialogue manager has been implemented and integrated into an interactive QA system.

1 Introduction

During (task-oriented) dialogues, the participants have to deal with many different aspects of communication simultaneously. Besides some underlying task that may be performed through the dialogue, there are also various aspects of managing the communicative process itself, including dealing with social obligations. Therefore, speakers often use utterances that are multifunctional.

We will present an approach to dialogue management that accounts for the generation of multifunctional utterances. The approach is based on a dialogue theory involving a multidimensional dialogue act taxonomy and associated context model. In this theory, called Dynamic Interpretation Theory (DIT) (Bunt, 1996; Bunt, 2000a), a dialogue is modelled as a sequence of (sets of) *dialogue acts* operating on the *Information State* of each of the participants. The dialogue acts are organised in a taxonomy that is *multidimensional*, i.e., each utterance may involve dialogue acts of at most one type from each dimension. The taxonomy has dimensions for aspects like feedback, interaction-management, social obligations management and managing the underlying task.

In a dialogue system developed according to the principles of DIT, the information state is represented through a context model, containing all information considered relevant for interpreting user utterances an generating system utterances in terms of dialogue acts. Hence, given the multidimensionality of the taxonomy, the input interpretation components of the system result in several dialogue acts for each utterance, at most one from each of the dimensions. Using these recognised user dialogue acts, the context model is updated.

On the other hand, the ultimate task for a dialogue manager component of a dialogue system is deciding which dialogue acts to generate. So, again with the multidimensional organisation of the taxonomy in mind, we argue for a multi-agent approach, in which the dialogue act generation task is divided over several agents that operate in parallel on the context model, each agent being dedicated to the generation of dialogue acts from one particular dimension in the taxonomy. This leads to the design of a number of so-called *Di-*

alogue Act Agents, including e.g. a task-oriented agent, two feedback agents and an agent dealing with social obligations management.

The multi-agent approach to dialogue management itself is not new: JASPIS (Turunen and Hakulinen, 2000; Salonen et al., 2004) is a multi-agent framework for dialogue systems which allows for implementations of several agents for the same tasks, varying from input interpretation and output presentation to dialogue management. Depending on the situation, the agent that is most appropriate for a given task is selected in a process involving several so-called 'evaluators'. In JASPIS the multi-agent approach is aimed at flexibility and adaptiveness, while our approach focuses more on supporting multidimensionality in communication.

In a very general sense, our dialogue management approach follows an information state update approach similar to the dialogue managers that are developed within the TRINDI framework (Larsson and Traum, 2000). For example, Matheson et al. (2000) describe the implementation of a dialogue management system focusing in the concepts of grounding and discourse obligations.

An approach to dialogue management which identifies several simultaneous processes in the generation of system utterances, is described in (Stent, 2002). In this approach, which is implemented in the TRIPS dialogue system, dialogue contributions are generated through three core components operating independently and concurrently, using a system of conversation acts organised in several levels (Traum and Hinkelman, 1992).

Although there are apparent similarities between our approach and that of the TRINDI based dialogue managers and the TRIPS system, there are clear differences as well, which for an important part stem from the system of dialogue acts used and the way the information state is organised. More particularly, the way in which mechanisms for generating dialogue acts along multiple dimensions are modelled and implemented by means of multiple agents, differs from existing approaches.

This paper is organised as follows. First we explain the closely connected DIT notions of dialogue act and information state, and the multidimensional dialogue act taxonomy and context model (Sections 2 and 3). We then introduce the multi-agent approach to dialogue management (Section 4) and illustrate it by a description of the current implementation (Section 4.1). This implementation is carried out in the PARADIME project (PARallel Agent-based DIalogue Management Engine), which is part of the multiproject IMIX (Interactive Multimodal Information Extraction). The PARADIME dialogue manager is integrated into an interactive question-answering system that is developed in a collaboration between several projects participating in IMIX. The paper ends with conclusions and directions for future research (Section 5).

2 The DIT dialogue act taxonomy

Based on studies of a variety of dialogues from several dialogue corpora, a dialogue act taxonomy was developed consisting of a number of *dimensions*, reflecting the idea that during a dialogue, several aspects of the communication need to be attended to by the dialogue participants (Bunt, 2006). Even within single utterances, several aspects are dealt with at the same time, i.e., in general, utterances are *multifunctional*. The multidimensional organisation of the taxonomy supports this multifunctionality in that it allows several dialogue acts to be performed in each utterance, at most one from each dimension. The 11 dimensions of the taxonomy are listed below, with brief descriptions and/or specific dialogue act types in that dimension. For convenience, the dimensions are further grouped into so-called *layers*. At the top level are two layers: one for dialogue control acts and one coinciding with the task-domain dimension. Dialogue control is further divided into 3 layers: Feedback (2 dimensions), Interaction Management (7 dimensions), and a layer coinciding with the Social Obligations Management dimension.

- **Dialogue Control**
  - **Feedback**
    1. ***Auto-Feedback***: acts dealing with the speaker's processing of the addressee's utterances; contains positive and negative feedback acts on the levels of perception, interpretation, evaluation, and execution;
    2. ***Allo-Feedback***: acts dealing with the addressee's processing of the speaker's previous utterances (as viewed by the

speaker); contains positive and negative feedback-giving acts and feedback elicitation acts, both on the levels of perception, interpretation, evaluation, and execution;

- **Interaction management**
   3. ***Turn Management***: turn accepting, giving, grabbing, keeping;
   4. ***Time Management***: stalling, pausing;
   5. ***Dialogue Structuring***: opening, preclosing, closing, dialogue act announcement;
   6. ***Partner Processing Management***: completion, correct-misspeaking;
   7. ***Own Processing Management***: error signalling, retraction, self-correction;
   8. ***Contact Management***: contact check, contact indication;
   9. ***Topic Management***: topic introduction, closing, shift, shift announcement;
   10. ***Social Obligations Management***: salutation, self-introduction, gratitude, apology, valediction;

11. ***Task/domain***: acts that concern the specific underlying task and/or domain.

Formally, a dialogue act in DIT consists of a *Semantic Content* and a *Communicative Function*, the latter specifying how the information state of the addressee is to be updated with the former. A dialogue act in a particular dimension may have either a dimension-specific communicative function, or a *General-Purpose* communicative function with a *content type* (type of semantic content) in that dimension. The general-purpose communicative functions are hierarchically organised into the branches of Information Transfer and Action Discussion functions, Information Transfer consisting of information-seeking (e.g., WH-QUESTION, YN-QUESTION, CHECK) and information-providing functions (e.g., INFORM, WH-ANSWER, YN-ANSWER, CONFIRM), and Action Discussion consisting of commissives (e.g., OFFER, PROMISE, ACCEPT-REQUEST) and directives (e.g., INSTRUCT, REQUEST, DECLINE-OFFER).

The taxonomy is currently being evaluated in annotation experiments, involving several annotators and several dialogue corpora. Measuring inter-annotator agreement will give an indication of the usability of the taxonomy and annotation scheme. A first analysis has resulted in promising scores (Geertzen and Bunt, 2006).

3 The DIT context model

The Information State according to DIT is represented by a Context Model, containing all information considered relevant for interpreting user utterances (in terms of dialogue acts) and generating system dialogue acts (leading to system utterances). The contents of the context model are therefore very closely related to the dialogue act taxonomy; in (Bunt and Keizer, 2005) it is argued that the context model serves as a formal semantics for dialogue annotation, such an annotation being a kind of underspecified semantic representation. In combination with additional general conceptual considerations, the context model has evolved into a five component structure:

1. *Linguistic Context*: linguistic information about the utterances produced in the dialogue so far (a kind of 'extended dialogue history'); information about planned system dialogue acts (a 'dialogue future');

2. *Semantic Context*: contains current information about the task/domain, including assumptions about the dialogue partner's information;

3. *Cognitive Context*: the current processing states of both participants (on the levels of perception, interpretation, evaluation, and task execution), as viewed by the speaker;

4. *Physical and Perceptual Context*: the perceptible aspects of the communication process and the task/domain;

5. *Social Context*: current communicative pressures.

In Figure 1, a feature structure representation of the context model is given, in which the five components have been specified in further detail. This specification forms the basis for the dialogue manager being implemented in the PARADIME project.

The Linguistic Context contains features for storing dialogue acts performed in the dialogue so far: *user_utts* and *system_utts*, having lists of dialogue act representations as values. It also has features for information about topics and conversational structure: *topic_struct* and *conv_state* respectively. Finally, there are two features that

$$\begin{bmatrix}
\mathit{LingContext}: \begin{bmatrix} \mathit{user\_utts}: \langle \mathit{last\_user\_dial\_act} = \mathit{uda}_0, \mathit{uda}_{-1}, \mathit{uda}_{-2}, \ldots \rangle \\ \mathit{system\_utts}: \langle \mathit{last\_system\_dial\_act} = \mathit{sda}_0, \mathit{sda}_{-1}, \mathit{sda}_{-2}, \ldots \rangle \\ \mathit{topic\_struct}: \langle \mathit{referents} \rangle \\ \mathit{conv\_state}: \mathit{opening} | \mathit{body} | \mathit{closing} \\ \mathit{candidate\_dial\_acts}: \ldots \\ \mathit{dial\_acts\_pres}: \ldots \end{bmatrix} \\
\mathit{SemContext}: \begin{bmatrix} \mathit{task\_progress}: \mathit{comp\_quest} | \mathit{quest\_qa} | \mathit{answ\_eval} | \mathit{user\_sat} \\ \mathit{user\_info\_needs}: \langle \ldots, \begin{bmatrix} \mathit{question}: \ldots \\ \mathit{satisfied}: + | - \end{bmatrix}, \ldots \rangle \\ \mathit{qa\_answers}: \langle \ldots \rangle \end{bmatrix} \\
\mathit{CogContext}: \begin{bmatrix} \mathit{own\_proc\_state}: [\mathit{proc\_problem}: \mathit{perc} | \mathit{int} | \mathit{eval} | \mathit{exec} | \mathit{none}] \\ \mathit{partner\_proc\_state}: [\mathit{proc\_problem}: \mathit{perc} | \mathit{int} | \mathit{eval} | \mathit{exec} | \mathit{none}] \end{bmatrix} \\
\mathit{PhysPercContext}: [\ ] \\
\mathit{SocContext}: \begin{bmatrix} \mathit{reactive\_pressures}: \mathit{none} | \mathit{grt} | \mathit{apo} | \mathit{thk} | \mathit{valed} \\ \mathit{interactive\_pressures}: \mathit{none} | \mathit{grt} | \mathit{apo} | \mathit{thk} | \mathit{valed} \end{bmatrix}
\end{bmatrix}$$

Figure 1: Feature structure representation of the PARADIME context model.

are related to the actual generation of system dialogue acts: *candidate_dial_acts* stores the dialogue acts generated by the dialogue act agents, and *dial_acts_pres* stores combined dialogue acts for presentation as system output; in Section 4, this will be discussed in more detail.

The specification of the Semantic Context is determined by the character of the task-domain. In Section 4.1, the task-domain of interactive question-answering on encyclopedic medical information will be discussed and from that, the specification of the Semantic Context for this purpose.

The Cognitive Context is specified by means of two features, representing the processing states of the system (*own_proc_state*) and the user (*partner_proc_state*). Both features indicate whether or not a processing problem was encountered, and if so, on which level of processing this happened.

The Physical and Perceptual Context is considered not to be relevant for the current system functionality.

The Social Context is specified in terms of reactive and interactive pressures; the corresponding features indicate whether or not a pressure exists and if so, for which social obligations management act it is a pressure (e.g., *reactive_pressures: grt* indicates a pressure for the system to respond to a greeting).

4 Dialogue Act Agents

Having discussed the dialogue act taxonomy and context model in DIT, we can now move on to the dialogue management approach that is also closely connected to these concepts. Having 11 dimensions of dialogue acts that each attend to a different aspect of communication, the generation of (system) dialogue acts should also happen along those 11 dimensions. As a dialogue act in a dimension can be selected independent of the other dimensions, we propose to divide the generation process over 11 *Dialogue Act Agents* operating in parallel on the information state of the system, each agent dedicated to generating dialogue acts from one particular dimension.

All of the dialogue act agents continuously monitor the context model and, if appropriate, try to generate candidate dialogue acts from their associated dimension. This process of monitoring and act generation is modelled through a triggering mechanism: if the information state satisfies the agent's *triggering conditions*, i.e., if there is a motivation for generating a dialogue act from a particular dimension, the corresponding agent gets triggered and tries to generate such a dialogue act. For example, the Auto-Feedback Agent gets triggered if a processing problem is recorded in the Own Processing State of the Cognitive Context. The agent then tries to generate a negative auto-feedback act in order to solve the processing problem (e.g., “Could you repeat that please?” or “Did you say ’five’?”). The Auto-Feedback Agent may also be triggered if it has reason to believe that the user is not certain that the system has understood a previous utterance, or simply if it has not given any explicit positive feedback for some time. In these cases of triggering, the agent tries to generate a positive auto-feedback act.

Hence the dialogue management process involves 11 dialogue act agents that operate in parallel on the context model. The dialogue acts generated by these agents are kept in the linguistic context as candidates. The selection of dialogue acts from different dimensions may happen independently, but for their order of performance and their combination, the relative importance of the dimensions at the given point in the dialogue has to be taken into account.

An additional *Evaluation Agent* monitors the list of candidates and decides which of them can be combined into a multifunctional system utterance for generation, and when. Some of the dialogue act candidates may have higher priority and should be generated at once, some may be stored for possible generation in later system turns, and some will already be implicitly performed through the performance of other candidate acts.

4.1 A dialogue manager for interactive QA

The current implementation of the PARADIME dialogue manager is integrated in an interactive question-answering (QA) system, as developed the IMIX multiproject. The task-domain at hand concerns encyclopedic information in the medical domain, in particular RSI (Repetitive Strain Injury). The system consists of several input analysis modules (ASR, syntactic analysis in terms of dependency trees, and shallow semantic tagging), three different QA modules that take self-contained domain questions and return answers retrieved from several electronic documents with text data in the medical domain, and a presentation module that takes the output from the dialogue manager, possibly combining any QA-answers to be presented, into a multimodal system utterance.

The dialogue management module provides support for more interactive, coherent dialogues, in which problems can be solved about both communication and question-answering processes. In interaction with the user, the system should play the role of an Information Search Assistant (ISA). This HCI metaphor posits that the dialogue system is not an expert on the domain, but merely assists the user in formulating questions about the domain that will lead to QA answers from the QA modules satisfying the user’s information need (Akker et al., 2005).

In the context model for this dialogue manager, as represented by the feature structure in Figure 1, the Semantic Context has been further specified according to this underlying task. It contains a state variable for keeping track of the question-answering process (the feature *task_progress* with values to distinguish between the states of composing a self-contained question to send to the QA modules, waiting for the QA results in case a QA-question has been sent, evaluating the QA results, and discussing the results with the user). Also, the Semantic Context keeps a record of user’s information need, by means of a list *user_info_needs* of ’information need’ specifications in terms of semantic descriptions of domain *questions* and whether or not these info-needs have been *satisfied*.

For the first version of the dialogue manager we have defined a limited system functionality, and following from that a simplified version of the dialogue act taxonomy. This simplification means for example that Social Obligations Management (SOM) and the various dimensions in the Interaction Management (IM) layer have been merged into one dimension, following the observation that utterances with a SOM function very often also have a function in the IM layer, especially in human-computer dialogue; see (Bunt, 2000b). Also several general-purpose communicative functions have been clustered into single types. Table 1 lists the dialogue acts that the dialogue act recogniser is able to identify from user utterances.

GP	AUF	IM-SOM
YN-Question WH-Question H-Question Request Instruct	PosAutoFb NegAutoFb-Int NegAutoFb-Eval	Init-Open Init-Close

Table 1: Dialogue act types for interpreting user utterances.

Table 2 lists the dialogue acts that can be generated by the dialogue manager. Task-domain acts, generally answers to questions about the domain, consist of a general-purpose function (either a WH-ANSWER or UNC-WH-ANSWER; the latter reflecting that the speaker is uncertain about the information provided) with a semantic content containing the answers obtained from QA.

AUF	ALF	IM-SOM
NegAutoFb-Int	Fb-Elicit	React-Open
NegAutoFb-Exe		React-Close

Table 2: Dialogue act types for generating system responses.

The above considerations have resulted in a dialogue manager containing 4 dialogue act agents that operate on a slightly simplified version of the context model as specified in Figure 1: a *Task-Oriented (TO) Agent*, an *Auto-Feedback (AUF) Agent*, an *Allo-Feedback (AUF) Agent*, and an *Interaction Management and Social Obligations Management (IMSOM) Agent*. In addition, a (currently very simple) Evaluation Agent takes care of merging candidate dialogue acts for output presentation.

In Appendices A.1 and A.2, two example dialogues with the IMIX demonstrator system are given, showing system responses based on candidate dialogue acts from several dialogue act agents. The ISA metaphor is reflected in the system behaviour especially in the way in which QA results are presented to the user. In system utterances S2 and S3 in Appendix A.1, for example, the answer derived from the retrieved QA results is isolated from the first part of the system utterance, showing that the system has a neutral attitude concerning that answer.

4.1.1 The Task-Oriented Agent

The TO-Agent is dedicated to the generation of task-specific dialogue acts, which in practice involves ANSWER dialogue acts intended to satisfy the user's information need about the (medical) domain as indicated through his/her domain questions. The agent is triggered if a new information need is recorded in the Semantic Context. Once it has been triggered, the agent sends a request to the QA modules to come up with answers to a question asked, and evaluates the returned results. This evaluation is based on the number of answers received and the confidence scores of the answers; the confidence scores are also part of the output of the QA modules. If the QA did not find any answers or if the answers produced had confidence scores that were all below some *lower threshold*, the TO-Agent will not generate a dialogue act, but write an execution problem in the Own Processing State of the Cognitive Context (which causes the Auto-Feedback Agent to be triggered, see Section 4.1.2; an example can be found in the dialogue in Appendix A.2). Otherwise, the TO-Agent tries to make a selection from the QA answers to be presented to the user. If this selection will end up containing extremely many answers, again, an execution problem is written in the Cognitive Context (the question might have been too general to be answerable). Otherwise, the selection will be included in an answer dialogue act, either a WHANSWER, or UNCWHANSWER (uncertain wh-answer) in case the confidence scores are below some *upper threshold*. System utterances S1 and S2 in the example dialogue in Appendix A.1 illustrate this variation. The selection is narrowed down further if there is a subselection of answers with confidences that are significantly higher than those of the other answers in the selection.

4.1.2 The Auto-Feedback-Agent

The AUF-Agent is dedicated to the generation of auto-feedback dialogue acts. It currently produces negative auto-feedback acts on the levels of interpretation ("I didn't understand what you said"), evaluation ("I do not know what to do with this") and execution ("I could not find any answers to your question"). It may also decide to occasionally give positive feedback to the user. In the future, we would also like this agent to be able to generate articulate feedback acts, for example with the purpose of resolving reference resolution problems, as in:

U: what is RSI?

S: RSI (repetitive strain injury) is a pain or discomfort caused by small repetitive movements or tensions.

U: how can it be prevented?

S: do you mean 'RSI' or 'pain'?

4.1.3 The Allo-Feedback Agent

The *ALF-Agent* is dedicated to the generation of allo-feedback dialogue acts. For example, it may generate a feedback-elicitation act if it has reason to believe that the user might not be satisfied with an answer ("Was this an answer to your question?").

4.1.4 Interaction Management and Social Obligations Management Agent

The *IM-SOM Agent* is dedicated to the generation of social obligations management acts, possibly also functioning as dialogue structuring acts (opening resp. closing a dialogue through a greeting resp. valediction act). It gets triggered if communicative pressures are recorded in the Social Context. Currently it only responds to reactive pressures as caused by initiative greetings and goodbyes. The example dialogues in Appendices A.1 and A.2 illustrate this type of social behaviour.

4.1.5 Multi-agent Architecture of the Dialogue Manager

In Figure 2, a schematic overview of the multi-agent dialogue manager is given. It shows the context model with four components (for now, the Physical and Perceptual Context is considered to be of minor importance and is therefore ignored), a set of dialogue act agents, and an Evaluation Agent. The dialogue act agents each monitor the context model and may be triggered if certain conditions are satisfied. The TO-agent may also write to the Cognitive Context (particularly in case of execution problems). All agents may construct a dialogue act and write it in the candidates list in the Linguistic Context. The Evaluation Agent monitors this candidates list and selects one or more dialogue acts from it for presentation as system output. In this way, a control module may decide to take this combination of dialogue act for presentation *anytime* and send it to the presentation module to produce a system utterance.

With this initial design of a multi-agent dialogue manager, the system is able to support multifunctional output. The beginning of the example dialogue in Appendix A.1 illustrates multifunctionality, both in input interpretation and output generation. The system has recognised two dialogue acts in processing U1 (a conventional opening and a domain question), and S1 is generated on the basis of two candidate dialogue acts generated by different dialogue act agents: the IM-SOM-Agent (generated the react-greeting act) and the TO-Agent (generated the answer act).

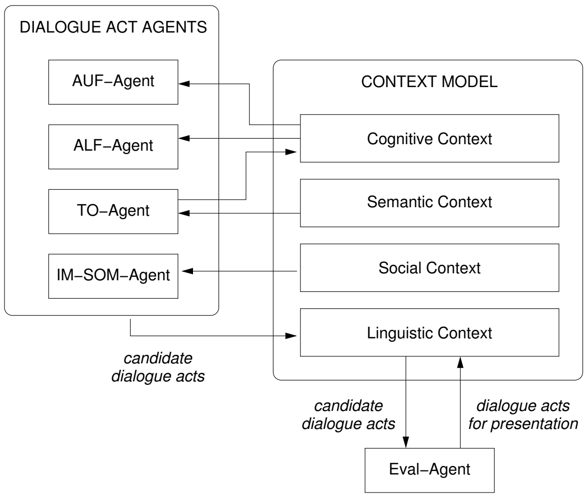

Figure 2: Architecture of the PARADIME dialogue manager.

5 Conclusions and future work

We have presented a dialogue management approach supporting the generation of multifunctional utterances. The approach builds on a dialogue theory involving a multidimensional dialogue act taxonomy and an information state on which the dialogue acts operate. Several dialogue acts from different dimensions are generated by dialogue act agents associated with these dimensions, and can thus be combined into multifunctional system utterances.

A first implementation of a dialogue manager following this multi-agent approach has been integrated into an interactive QA system and supports a limited range of dialogue acts from the DIT taxonomy, both for interpreting user utterances and generating system utterances. The system is able to attend to different aspects of the communication simultaneously, involving reactive social behaviour, answering domain questions and giving feedback about utterance interpretation and the question-answering process.

Future development will involve extending the range of dialogue acts to be covered by the dialogue manager, for a part following from the definition of an extended system functionality, and consequently, extending the set of dialogue act agents. This also has consequences for the Evaluation Agent: the process of combination and selection will be more complex if more dialogue act types can be expected and if the dialogue acts have a semantic content that is more than just a collection of QA-answers.

In terms of system functionality we aim at support for generating *articulate feedback*, i.e., feedback acts that are not merely signalling processing success or failure, but (in case of negative feedback) also contain a further specification of the processing problem at hand. For example, the system may have encountered problems in processing certain parts of a user utterance, or in resolving an anaphor; then it should be able to ask the user a specific question in order to obtain the information required to solve the processing problem (see the example in Section 4.1.2). The articulate feedback acts may also involve dealing with problems in the question answering process, where the system should be able to give specific instructions to the user to reformulate his question or give additional information about his information need.

In addition to supporting generation of articulate feedback acts, we also aim at dialogues between user and system that are more coherent and natural, i.e., the system should be more aware of the conversational structure, and display more refined social behaviour. Not only should it generate simple reactions to greetings, apologies, and goodbyes; it should also be able to generate initiative social acts, for example, apologies after several cases of negative auto-feedback.

The extended set of dialogue acts will also lead to an extended context model. Related to the context model and updating mechanism is ongoing work on belief dynamics and grounding in DIT (Morante and Bunt, 2005). The defined mechanisms for the creation, strengthening, adoption, and cancelling of beliefs and goals in the context model are currently being implemented in a demonstrator tool and will also be integrated in the information state update mechanism of the PARADIME dialogue manager.

Acknowledgement

This work is part of PARADIME (Parallel Agent-based Dialogue Management Engine), which is a subproject of IMIX (Interactive Multimodal Information eXtraction), a multiproject on Dutch language and speech technology, funded by the Dutch national science foundation (NWO).

We would like to thank the reviewers for their valuable comments, which really helped us to improve our paper.

References

R. op den Akker, H. Bunt, S. Keizer, and B. van Schooten. 2005. From question answering to spoken dialogue: Towards an information search assistant for interactive multimodal information extraction. In *Proceedings of the 9th European Conference on Speech Communication and Technology, Interspeech 2005*, pages 2793–2796.

H. Bunt and S. Keizer. 2005. Dialogue semantics links annotation to context representation. In *Joint TALK/AMI Workshop on Standards for Multimodal Dialogue Context*. `http://homepages.inf.ed.ac.uk/olemon/standcon-SOI.html`.

H. Bunt. 1996. Dynamic interpretation and dialogue theory. In M.M. Taylor, F. Néel, and D.G. Bouwhuis, editors, *The Structure of Multimodal Dialogue, Volume 2*, pages 139–166. John Benjamins.

H. Bunt. 2000a. Dialogue pragmatics and context specification. In H. Bunt and W. Black, editors, *Abduction, Belief and Context in Dialogue*, Studies in Computational Pragmatics, pages 81–150. John Benjamins.

H. Bunt. 2000b. Non-problems and social obligations in human-computer conversation. In *Proceedings of the 3rd International Workshop on Human-Computer Conversation*, pages 36–41.

H. Bunt. 2006. Dimensions in dialogue act annotation. In *Proceedings Fifth International Conference on Language Resources and Evaluation (LREC 2006)*.

J. Geertzen and H. Bunt. 2006. Measuring annotator agreement in a complex hierarchical dialogue act annotation scheme. In *7th SIGdial Workshop on Discourse and Dialogue*.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering: Special Issue on Best Practice in Spoken Language Dialogue Systems*, 6(3–4):323–340.

C. Matheson, M. Poesio, and D. Traum. 2000. Modeling grounding and discourse obligations using update rules. In *Proceedings of the First Annual Meeting of the North American Chapter of the ACL.*

R. Morante and H. Bunt. 2005. Micro-analysis of the belief transfer in information dialogues. In C. Gardent and B. Gaiffe, editors, *DIALOR'05 Proceedings of the Ninth Workshop on the Semantics and Pragmatics of Dialogue*, pages 135–138.

E.P. Salonen, M. Hartikainen, M. Turunen, J. Hakulinen, and J.A. Funk. 2004. Flexible dialogue management using distributed and dynamic dialogue control. In *Proceedings of ICSLP 2004*.

A.J. Stent. 2002. A conversation acts model for generating spoken dialogue contributions. *Computer*

Speech and Language, Special Issue on Spoken Language Generation, 16(3–4):313–352.

D.R. Traum and E.R. Hinkelman. 1992. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 3(8):575–599.

M. Turunen and J. Hakulinen. 2000. Jaspis – a framework for multilingual adaptive speech applications. In *Sixth International Conference on Spoken Language Processing (ICSLP 2000)*, volume 2, pages 719–722.

A Example dialogues

A.1 Example 1: greetings, answers and unc-answers

U1: hallo, ik wilde graag weten wat rsi is

(hello, I wanted to know what is rsi)

[INIT-OPEN; INDWHQ, *task-domain*]

S1: Hallo. Ik heb de volgende informatie gevonden:

(repetitive strain injury) Algemene benaming voor (blijvende) beschadigingen aan het lichaam ten gevolge van langdurige, eentonige werkzaamheden.

(Hello. I found the following information: ...)

[REACT-OPEN; ANSWER, *task-domain*]

U2: wat moet ik doen als ik denk rsi te hebben

(what should I do if I think I have RSI)

[WHQUESTION, *task-domain*]

S2: De volgende informatie levert mogelijk een antwoord op uw vraag:

Ontspanning betekent minder stress, minder aangespannen spieren en dus minder kans op RSI.

(The following information might answer your question: ...)

[UNCANSWER, *task-domain*]

U3: helpt ontspanning ook als ik eenmaal rsi heb

(does relaxing help as well, once I have RSI)

[YNQUESTION, *task-domain*]

S3: Ik heb de volgende informatie gevonden:

Behandelmethoden bij RSI Er zijn vele wegen die naar Rome leiden. Verbetering van de doorbloeding ...

(I found the following information: ...)

[ANSWER, *task-domain*]

U4: oke, bedankt

(okay, thanks)

[INIT-CLOSE]

S4: tot ziens. U kunt de buttons gebruiken om te stoppen of om een nieuwe dialoog te beginnen.

(Goodbye. You may use the buttons to stop or to start a new dialogue.)

[REACT-CLOSE]

A.2 Example 2: negative execution-feedback

U1: hallo!

(hello!)

[INIT-OPEN]

S1: Hallo. Stel uw vraag.

(Hello. Ask your question.)

[REACT-OPEN]

U2: wat kan ik als beelschermwerker tegen rugpijn doen

(what can I, a person working behind a computer screen all the time, do against pain in the back)

[WHQUESTION, *task-domain*]

S2: Ik heb helaas geen antwoorden kunnen vinden; probeer de vraag te herformuleren.

(Unfortunately, I could not find any answers; please try to reformulate the question)

[NEGAUTOFB-EXEC]

U3: wat kan men tegen rugpijn doen

(what can be done against pain in the back)

[WHQUESTION, *task-domain*]

S3: ik heb de volgende informatie gevonden:

Deze is afhankelijk van de oorzaak. Rugpijn die door instabiliteit of door tekortschietende spieren en banden ontstaat, kan behandeld worden met houdingscorrigerende oefentherapie.

(I found the following information: ...)

[ANSWER, *task-domain*]

U4: bedankt!

(thanks!)

[INIT-CLOSE]

S4: Tot ziens. U kunt de buttons gebruiken om te stoppen of om een nieuwe dialoog te beginnen.

(Goodbye. U may use the buttons to stop or to start a new dialogue.)

[REACT-CLOSE]