Master thesis

Making a dialogue annotation scheme more usable

A usability study of the DIT++ taxonomy of dialogue acts with the think aloud method

Faculty of Humanities
Department of Communication and Information Sciences

Daphne Brabers
Grevelingen 47
5032 TC Tilburg
0646736714
d.brabers@uvt.nl
ANR: 987941

Supervisor: Prof. Dr. Harry Bunt

25 November 2008
Making a dialogue act annotation scheme more usable

A usability study of the DIT++ taxonomy of dialogue acts
with the think-aloud method

Daphne Brabers
HAIT Master Thesis series nr. 08-05

Thesis committee:
Prof. Dr. Harry Bunt
Volha Petukhova, MA
Dr. Menno van Zaanen
Prof. Dr. Eric Postma

Tilburg University
Faculty of Humanities
Department of Communication and Information Sciences
Tilburg, The Netherlands
November 2008
**Summary**

Dialogue act annotation is very important in making software that is able to analyze our natural language. The DIT++ model is a comprehensive dialogue act annotation scheme based primarily on the Dynamic Interpretation Theory from Bunt (2000). The DIT++ model is a multidimensional model that allows its users to assign multiple dialogue acts from the various dimensions to an utterance in a dialogue. In this paper a usability test is performed on the DIT++ model with the aim to make the model more usable and clear for its users. A meta-aspect of this research is to find out whether a usability analysis can be applied on a dialogue annotation model or not. The think aloud method is applied in an experiment performed by students. After one pilot experiment, three subjects were asked to think aloud while annotating a DIAMOND dialogue for one hour with the help of the video-annotation program ANVIL. The subjects annotated more than experts would do in the same amount of time, but had no more than 50 percent of the annotations correct. The results showed that annotation tags with unambiguous examples and descriptions in the model or in the help function of ANVIL were assigned correctly the most. Tags where descriptions were vague or extensive were wrongly assigned. Incorrect annotations were made because subjects were reasoning in the right direction but did not get the correct annotation; the subject made a completely wrong decision; or the subject made one annotation but failed to assign more tags to that segment of dialogue. The consult of the help function by the subjects was mainly as support or feedback of their already made decision about the annotation; therefore the classic view of using a help function only when you do not know what to do is outdated. The subjects stated in an interview afterwards that they miss certain tags, for example to indicate politeness. The DIT++ usability is tested against the ISO 9241 usability standard, which focuses on effectiveness, efficiency and user satisfaction. The DIT++ model scores low on effectiveness and high on efficiency and user satisfaction. Given the observation that the subjects annotated much faster than the expert annotators who annotated the same dialogue before, these results should be viewed in the light of a trade-off between speed and accuracy. The process that the subjects follow while using the model is not from the top (a dimension) to the bottom (an annotation tag). A user that used the model before, and knows the tag he wants to assign to a segment, does not take the steps from dimension to tag that the multidimensional character of the model invites him to, and the annotation tool in fact forces him to.
1. INTRODUCTION

Humans have a natural need to communicate with each other. Most of the time this communication develops smoothly and we do not experience many difficulties with human to human communication. From a scientific point of view, communication is full of noise and ambiguity and there is a need to clarify what really is said and meant in a dialogue. A description of what is happening in a dialogue can be made with the help of dialogue acts. “A dialogue act is a unit in the semantic description of communicative behavior in dialogue, specifying how the behavior is intended to change the information state of the addressee (“the context’) through his interpretation of the behavior” (Bunt 2007). It is difficult to give a less abstract definition of dialogue acts; Kipp (1998) states that “dialogue acts are a widely used means of representing the intention of a speaker”. Dialogue acts are important in science for building dialogue systems, for defining inter-agent communication between software agents, and for annotating dialogues. The latter is where dialogue acts are used for in this paper.

According to the Dynamic Interpretation Theory (DIT) as described by Bunt (2000) a dialogue is formed by dialogue acts that have an effect on the information state of the participants. The DIT++ model is the accompanying dialogue act annotation scheme, used to analyze dialogues. Dialogue act annotation can be used for the training of natural language processing systems; i.e. make computers “understand” language that we humans use naturally. In a dialogue the hearers’ information state is updated after hearing an utterance, and this so-called context-change approach is used in the DIT++ model. In a conversation analyzed by DIT a hearers’ information state is called the hearers’ context model. The context-change (the update of the information state) of a participant is described using dialogue acts. The context model of a hearer comprises all information the hearer has for the interpretation and generation of dialogue behaviour. The context model has five components (Keizer & Bunt, 2006):

1. Linguistic Context: this context contains stores the dialogue acts that have been performed so far in the conversation, and information about planned dialogue acts. Generally, the past and the future of the dialogue are saved here;
2. Semantic Context: current information about the task or domain is stored. Also assumptions of the information state of the dialogue partner;
3. Cognitive Context: the processing states of both participants of the dialogue, as viewed by the speaker. The processing state indicates the level of success in understanding;
4. Physical and Perceptual Context: the perceptible aspects of the communication and the task and domain;

5. Social Context: current communicative pressures as a result of the role the participants have in a dialogue; communicative rights and obligations (Bunt, 2000).

The model distinguishes several different dimensions, with tags describing the communicative functions of dialogue acts that are organized hierarchically (Appendix A). The multidimensionality of the model means that “participants select dialogue acts from different dimensions simultaneously and independently, and then combine them into multifunctional utterances” (Keizer & Bunt, 2007). Each utterance can have several functions but at most one in each dimension in the DIT++ model (Appendix B). This is in contrast to one-dimensional annotation schemes that follow the principle Searle (1969) stated: one utterance can form only one speech act. The DIT++ model is organized as a taxonomy as a result of this multidimensionality; there are two parts: a ten-dimensional taxonomy of dimension-specific communicative functions and a taxonomy of general-purpose functions. The dimensions in the DIT++ model are divided into different layers; which Keizer and Bunt (2006) describe as follows:

*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 three layers: Feedback (two dimensions), Interaction Management (seven dimensions), and a layer coinciding with the Social Obligations Management dimension.*

This multidimensional approach to dialogue acts and their annotation makes the DIT++ schema complicated, and is likely to cause problems. Because there are many dimensions in the DIT++ model and one utterance can have many tags, annotating is quite complicated. In this paper a usability analysis is performed on this complex DIT++ model to investigate how well it can be used by untrained annotators.

“Usability testing is primarily intended to provide information for the upgrading and maintenance of existing systems” (Preece, 1993). This is applicable to the DIT++ model; it is an existing model that may need to be upgraded. Usability is generally accepted as one of the main criteria in the development of computer programs, websites and other products. Usability has its own ISO (ISO 9241) certified definition, which focuses on effectiveness, efficiency and user satisfaction.

Geertzen, Petukhova and Bunt (2008) already compared the effectiveness per dimension of the model for naïve and expert annotators. The DIT++ model had a high tagging accuracy for
expert annotators. In this research the accuracy for tagging by beginning annotators will be measured as part of analyzing its usability.

Traditionally, usability tests are performed on industrial products, including computer software. Applying this method to the DIT++ model is a methodologically interesting enterprise, investigating the possibility of applying usability testing to measure the utility of models and methods rather than products. In measuring the usability of this model a distinction will be made between task completion and task success. The multidimensionality of the DIT++ model has a great effect on the complexity of the layout and user interface of the model. How does this contribute to the usability of the model? Does the DIT++ model meet the ISO standard and additional rules for usability? And, if not, how can this be improved? The answers to these questions are very relevant for the developers of annotation models and of importance for the users of these dialogue annotation models.

2. METHODS
As mentioned before, usability is usually applied to products or computer interfaces but since an annotation scheme can also be viewed as a kind of product it may also be applicable to the DIT++ taxonomy. Usability testing is an aspect of user-centered design; the users must be the focus of the design activity (Preece, 1993). The DIT++ model is intended to support annotation, and therefore it is crucial to make the task of the user as easy as possible. The ISO definition of usability is formulated as: “The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments” ¹. Evaluating the usability of the DIT++ model will be done by participants annotating dialogues while thinking aloud.

The think aloud method is an interesting way of getting to know what is going on in the mind of the participant. Van Someren, Barnard and Sandberg (1994) describe this method as “the subject is asked to talk aloud, while solving a problem and this request is repeated if necessary during the problem-solving process thus encouraging the subject to tell what he or she is thinking”. The subject is solving the problem and says whatever comes to mind, there are no questions asked or interruptions of the experiment leader. Think aloud as a method for usability evaluation of the DIT++ model is an empirical method.

¹ Source: www.w3c.org
There are several reasons to choose this method for evaluating usability. First, vocalization of our thoughts is something that comes natural to human adults; it is not unusual to speak out one’s thoughts aloud while solving difficult tasks, without the intention to communicate. This “spontaneous vocalization provides evidence for a close relation between internal activation of orally coded information and the corresponding vocalization” (Ericsson & Simon, 1993). Second, thinking aloud has little effect on the thought process; the performance of the task will not be affected (Van Someren et al., 1994). Third, it not only becomes clear what problems arise during the task but also why. Van den Haak, De Jong and Schellens (2003) have summed up a lot of questions on the research performed on the think aloud method; both positive and negative evaluations are being questioned.

For the usability test on the DIT++ model, three participants are needed. Jordan (1998) states that think aloud protocols are an efficient way of “obtaining a lot of information from only a few participants”. Kerr and Jordan (1994) were able to draw useful conclusions with only two participants in their think aloud experiment.

Faerch and Kasper (1987) suggest that an experiment with the think aloud protocol should start with a warm up. In this phase the subjects should do some initial practice, in this case with the ANVIL program and with thinking aloud. During this warm up subjects will discover what strategy they will use during the experiment while solving a few tasks thinking aloud, the tasks will be relatively easy like determining the number of windows in their homes (Ericsson & Simon, 1993). This warm up is essential to prevent silences due to misunderstanding of what think aloud really implies.

Although the experiment leader has a restrained role during the experiment, he or she does have a key role to make sure the experiment succeeds. Ericsson and Simon (1993) note that “during the warm up, the experimenter feels free to interfere with and disrupt the subject, while during the experiment, he is very concerned not to interfere”. Only when the participant stops talking and thinking aloud, the experimenter should give the subject reminders to speak. Van Someren et al. (1994) noticed that it is hard for the experimenter to only give these prompts to the subjects, especially when the experimenter is familiar with a domain. More recently Boren and Ramey (2000) questioned the role of the experimenter in a usability testing experiment Ericsson and Simon (1993) advocated. They found that the experimenter in the background principle is hardly ever complied in practice and plead for more freedom to interact with the subjects. The experimenter who remains in the background is useful for research into cognitive processes, and not for usability testing. Because the subjects are all students and acquainted to the experimenter, they would feel more comfortable if the
the experimenter is allowed to talk with them when needed. Note that the experimenter is not allowed to help the participants with annotating; the experimenter is only allowed to speak in order to help with ANVIL-related problems or support the subject to think aloud. Besides the think aloud protocol other methods are used to measure the usability of the DIT++ model. The efficiency and effectiveness are measured with the annotation time per unit of dialogue. Task completion and task success is measured in comparison with the gold standard and the use of the help function by the subjects. User satisfaction is measured with an interview after the experiment to evaluate how the subject experienced the usage of the model.

Subjects
There are four user characteristics that have an effect on usability. The first one is user experience, which means if the user has had previous experience with the product or not. The subjects for this experiment have little experience; they have followed classes on the DIT++ model and made two exercises using the model. Note that these exercises were made on paper and not with the help of the ANVIL program. The second characteristic is domain knowledge, the knowledge of the domain around the task without the DIT++ being used to complete the task. In the case of dialogue annotation everyone has domain knowledge; every human knows how a dialogue works and understands what a speaker means when he or she says “yes” or “uhmm”. The difficulty with the usage of the DIT++ model is that everybody unconsciously knows what is intended by the speaker but never really thought about it explicitly. The DIT taxonomy shows that the same part of a sentence can have multiple meanings at once, writing this exact meanings down can be difficult because hearers process this data automatically. The third characteristic, cultural background, does not really apply for this experiment. The cultural background is the same for all the subjects; they all have the Dutch nationality. Finally there is the age and gender of the user that need to be taken into account when a product is designed. The DIT++ model is a product that already is designed, probably without taking the age and gender of the users in account. The subjects are all students of the Tilburg University, male and female, in their early twenties.

Procedure
The terms of the ISO definition of usability mentioned earlier will be used for evaluating the DIT taxonomy. We will measure what the effectiveness of the model is; “the extent to which a goal, or task, is achieved” (Jordan, 1998). Efficiency stands for the time that is needed to
complete a task or to reach a goal. User satisfaction is described as “the level of comfort that the users feel when using a product” (Jordan, 1998). To measure the usability of the DIT++ model the following aspects are measured:

- Annotation time per unit of dialogue (per minute, experts annotate 1 minute of dialogue in 25 minutes);
- How many times the subject uses the help function in ANVIL;
- For what dimensions is the help function mostly consulted and for what tags is the help never used (self-explanatory names of the dimensions);
- Number of errors made (compared to the gold standard);
- Certainty of errors and certainty of correctly assigned tags compared;
- Number of correct annotations;
- Certainty of correct annotations;
- Comparison of the correct answers and errors, find out why a mistake is made;

These measures are combined with another way of testing usability, the think aloud method. There will also be a post-task walkthrough at the end of the experiment to evaluate what the user satisfaction is.

To decide whether an annotation is correct or incorrect a gold standard is used; but an annotation can also be partly correct. Geertzen and Bunt (2006) suggested the weighted kappa, which punishes disagreement in terms of how far the tags are away from each other in the DIT++ annotation scheme. Annotators fully disagree when:

1. The two tags each of them assigned belong to different dimensions in the DIT++ scheme;
2. one of the two tags is general-purpose and the other one belongs in a specific dimension;
3. the two tags belong to the same dimension but to a different hierarchy;
4. the two tags belong to the same hierarchy but are not in the same branch.

Thus, there can be disagreement between the annotators but it will be punished less if one gives a more “detailed” or specific tag than the other. Consider the following example with the hierarchy reflected in Figure 1.
Consider the following example where the subjects will be presented the following dialogue:

A. “I’m sorry, your plane will leave tomorrow morning on seven o’clock a.m.”
B. “Not tonight?!”

The correct annotation tag for B would be *negaCheck*. If a subject would choose *propositionalQuestion* instead it would be punished mildly because the two tags are from the same family. Geertzen and Bunt (2006) call this a parent-child relation between the two tags and allot partial disagreement between the annotators. For this experiment there will be no measurement of agreement between annotators; but this rule of partial disagreement will be used to measure agreement between the annotator and the gold standard and allow an answer to be partly right.

In their book, Van Someren et al. (1994) made an overview for the analysis of think aloud protocols, which are showed in Figure 2.
This scheme shows the main objects in the analysis of think aloud protocols and how they are connected to each other. The disadvantage of this model is that it is based upon modeling cognitive processes, which is needed in psychological research. The usability analysis of the DIT++ model will not go this far, so several steps in the model can be skipped. The aspects of the model that are relevant for evaluating the usability of the DIT++ model are: the task analysis, coded protocols, coding schemes, segmented protocols and raw protocols.

**Task analysis**

The analysis of the task is constructing a model from information about the task without considering psychological factors. For a lot of knowledge-based systems there already are existing or similar model to consult or use. A difficulty with the DIT++ model is that it is a
The goal of the users of the DIT++ model is making correct annotations of a dialogue. These goals lead to tasks and these tasks lead to actions (Hackos & Redish, 1998). A user of the DIT++ model chooses which segment of the dialogue he or she wants to annotate with dialogue acts. When the segment is selected the user preferably decides to which dimension(s) the segment belongs and next what the function of the segment is. This decision process has no fixed order; it is also possible for the user first to decide which tag the segment has and learn later to which dimension this function belongs.

**Coding scheme**

Coding schemes are a very important part of protocol analysis. For this experiment the coding scheme will exist of the features that are a part of the DIT++ model’s construction. The utterances of the subject will be coded with a dimension or function tag, to decide which one of the two led to the completion of the task. When there is doubt between two functions or dimensions this will be also be indicated. The steps that the subject takes towards the solution of the task will be displayed by this coding system. The codes will also be given to utterances where the subject does not name a DIT++ feature at all; but in a sense is talking about it unconsciously (Appendix C). For example when a subject says: “I think he wants to offer him to help” the proper tag from the DIT++ model would be *offer*. For some reason the subject knows what the speaker wants but cannot give the utterance the correct tag (yet). This proactive way of DIT tag attributing helps understanding the path the thoughts of the subject goes and if it leads to correct or incorrect answers.

This coding scheme will be used for the parts of the experiment where the subject talks about the dialogue and the DIT++ model; the parts where the subject talks about the ANVIL program are left out (marked with *ANVIL*). The application of this coding scheme on the segmented protocols will make clear what type of problems occur at what place in the DIT++ model.

**Coded protocols**

After the experiment the recorded protocol should be transcribed, segmented and coded. This is a time-consuming task; transcribing takes ten times as much time as the original recording (Van Someren et al., 1994). In case of problem solving with the help of the DIT++ model not everything the subject says is equally important, like in a psychological experiment. The
emphasis in this experiment is on the information source the subject uses, the DIT++ model. All possible problems that occur in combination of the use of ANVIL are irrelevant for this experiment. In this case the interest goes to those sections of the protocol where a reference is made to the information source (Van Someren et al., 1994). The coding scheme is the guideline to code the protocols with the right type of statement. Again, the codes show where in the use of the model problems occur and of what kind these problems are. The notes of the experimenter made during the post-task walkthrough with the subject can also be taken into account. There is an evaluating interview directly after the experiment to make sure that everything is still fresh in the mind of the subject.

- Predicted coded protocols. Dependent of psychological model, this makes predictions for the coded protocols. Since there is no psychological model involved in this experiment this step is skipped.

- Segmented protocols. Divide the protocol into segments; speech has boundaries that are usually marked by pauses. So it is not only important to write down what the subjects are saying but also write down when they are not speaking. These silences are a natural boundary for speech and should be taken into account while analyzing the protocols. The pauses in the speech of the subject are marked with a new line in the protocol; when the subject has a natural pause for longer than one second a new line is inserted. After this first segmentation there will be a second one; separating ideas (Smith, 1971) where the emphasis is on the content of the utterances. The protocols will also be separated on the segment in the DIAMOND dialogue that they concern.

- Raw protocols. Raw protocols are the transcribed protocols from the audio recording and perhaps some notes from the experimenter. This will not be a separate step in the transcribing of the protocols because the segments are inserted immediately during the transcribing of the audio material.

ANVIL
ANVIL (Annotation of Video and Language) originally is designed as a tool for audiovisual annotation. For this research the video component was not necessary, the subjects only use the text and sound of the dialogue for annotation. The subjects can, after they assigned tags to segments, rate how sure they are of their decision on a five-level Likert-scale. In ANVIL the annotation is divided into multiple layers, called tracks. In these tracks subjects can make segments; they can point from where to where the dialogue act reaches and the layers allow
the subjects to give multiple tags to one segment. In ANVIL these different segments are called elements, and each element has to receive a value; in this case a tag of the DIT++ model.

It is important not to confuse the use of the ANVIL annotation tool with the DIT++ model; the interest of this experiment goes to the usability of the DIT++ annotation model and not the usability of ANVIL.

The user interface of ANVIL when used for annotation with the DIT++ model is displayed in Figure 3.

**Figure 3.**
*Interface of ANVIL*

In the upper center is the video screen which not will be used in this experiment. At the bottom there are tracks; firstly divided into participant A and B, then divided into eleven other tracks; ten for the dimensions in the model and one for “other”. The reason for why there are so many tracks is because of the multidimensionality of the DIT++ model; as explained earlier more than one tag can be assigned to a segment of an utterance. In the tracks there are utterances which are pre-segmented.
The subject annotates the dialogue by clicking in the right track titled as one of the dimensions of the DIT++ model. It is very important that the subject clicks within the borders of the segment of the dialogue that the subject want to annotate. Then the subject can click the Start-button and after that the End-button; a pop-up window occurs. It is this window where the subject makes a choice in dimension or function and how certain he or she is of the tag.

**Figure 4.**
*Pop-up window for filling in tags and certainty*

The light blue highlighted track is the dimension that the subject has chosen as the dimension the text segment belongs to. After pressing the start- and end-button the pop-up window occurs; the subject has chosen *signalSpeakingError* of the dimension *ownCommunicationManagement* (ocm) with a certainty of 3.

**Screenplay for the pilot experiment**

To make sure everything runs smoothly during the real experiment, a pilot experiment was done first. This pilot was on a smaller scale than the actual experiment, and provided feedback
about what might go right and what might go wrong during the experiment. In the pilot only one subject was thinking aloud for an hour while using the DIT++ model.

Thinking aloud must come as natural as possible for the subject, which is why the experiment was held in Dutch because this is the first language of the subject. The audio and written material that was annotated was in Dutch, as well as the instructions and the aloud thinking. The only thing that was in English is the ANVIL interface and the terminology of the DIT++ model. Since the subjects are all familiar with the DIT++ model and its English terminology, and the lectures about the model were in English, this should be no problem. The subject sat behind a laptop with a separate mouse so the operation of the laptop was optimal. Speakers were attached to the laptop to make sure the subject can hear the audio of the conversation well. The experimenter sat obliquely behind the subject so the subject can be heard and his clicks and move can be witnessed. It was important that the subject did not experience any inconvenience from the experimenter while performing the task. Firstly, the experimenter checked if the recording device is working and then started recording the experiment. The subject was given an instruction about the experiment and a short overview of the DIT++ model to freshen up his memory. In the instruction the subject read what data the experimenter wants to gather with the experiment and what is expected from the subject. After the subject has read the instruction it is time to do the warming up task. This warming up helped to make the subject comfortable with thinking aloud and refreshes the subjects’ memory about the DIT++ model. During the warming up task the subject became familiar with the use of ANVIL. The duration of this warm up depended on the subject; when the experimenter is confident that the subject is comfortable with thinking aloud, the real experiment can begin. For the warming up a simple dialogue was used called Schiphol Information Office. This dialogue was text only and does not contain audio. This was convenient for the warm up because the experimenter was talking to the subject to explain things and motivating the subject to think aloud. For the actual experiment a dialogue from the DIAMOND project is used.

*Dialogues from both corpora are two-agent human-human dialogues. DIAMOND dialogues have an assistance-seeking nature with one participant playing the role of an instructor explaining to the user how to configure and operate a fax-machine. Schiphol Information Office dialogues are information-seeking dialogues where an assistant is requested to provide a client the information all around the airport activities and facilities (e.g. timetable, security, etc.). The original DIAMOND*
dialogue is pre-segmented per dialogue utterance for each speaker with indication of utterance start- and end time. The original Schiphol dialogues are pre-segmented per speaker turn without authentic turn timings.\(^2\)

The DIAMOND dialogue is segmented into utterances. During the experiment the experimenter encouraged the subject to speak his/her mind. While the subject was solving the annotation tasks the experimenter watched the times the subject used the help function and for what DIT++ tags the help function was consulted. The actual experiment took one hour. After the experiment there was a post-task walkthrough; the subject was asked some evaluating questions about the course of the task. The recorded audio was played again and discussed with the subject. Questions like “why did you hesitate there?” and “why did you made that choice?” are asked.

**Pilot experiment**

The experiment overall went very well. The subject was comfortable with thinking aloud and gave enough useful data for analyzing the DIT++ model. There was one big drawback throughout the whole hour; ANVIL did not work properly. The program became an obstacle in the experiment as a result of which measuring the use of DIT++ became difficult. The sound and utterances in the program were not synchronous. Sometimes the END-button could not be pushed so that no tag could be assigned to the segment. When the pop-up did appear to assign a tag, the last chosen tag in that dimension was already filled in, so the subjects had to change that tag into the right one and adjust the certainty scale. This certainty was rarely changed, for example if it was at three the subject left it for what it was. There were also tags that were filled in already when the subject came further into the dialogue, when the file was re-opened these tags were gone. The ANVIL interface is also too big when using the DIT++ model because all ten dimensions have to be on the screen with tracks for each participant. The subject has to scroll up and down to see whether or not participant A or B has said something. The horizontal scrolling, the scrolling from one second to another in the timeline, goes too slow. And when the subject wants to drag the scrollbar to the right this goes too fast and the subject passes lots of segments. Ultimately the program worked just well enough to use it, but too much time was wasted with dealing with ANVIL problems. After the pilot experiment with a few changes in the version and properties of the program almost all errors were repaired, only the synchrony of text and sound and easier scrolling could not be realized.

Results of the pilot experiment

The first thing that stands out of the annotations of the subject compared to the gold standard is that the subject does not use the multidimensionality of the model as often as used in the gold standard. *Stalling* seems to be the most obvious case throughout the dialogue. It is something that occurs very often in a dialogue and is the most easy to tag; the certainty ratings of the subject for the *stalling* tags were 4 or 5. Compared to the gold standard, all of them were correct. It may be concluded that *stalling* is one of the easiest tags to assign in a dialogue; this is probably the case because *stalling* has very obvious signal words, like “hmm” and “uhm”. A big drawback of this easiness of the *stalling* tag is that the subject never assigns any other tags next to the *stalling* tag; the subject assigns just the *stalling* tag and thinks he is done. In the gold standard *stalling* never occurs by itself and always is accompanied with the *turnInitial* and *turnFinal* dimensions. Maybe it is not clear to the subject how far they must go with the segments of the dialogue and how to use the models’ multidimensionality to the fullest.

It is difficult to look up afterwards for what function or dimension the help function was used. The subject does not always talk about using the help function and for what part of the dialogue the function is consulted. Another puzzle that is hard to solve is matching the annotations of the subject to the gold standard, because the segmentation of the dialogue in the DIAMOND file are a bit different from the segments in the gold standard. It is a shame that the first annotations with ANVIL were not saved due to problems with ANVIL. During the second try the subject already had made the first annotations once and corrected some of his errors made earlier and did not have to think about his answer for very long, so there is not a lot of audio about these decisions. The help function was used in total for twenty-one times, but unfortunately the notes made during the experiment are not profound enough to derive exactly when and for what segment the help was used.

The annotation time per unit of dialogue could not be measured precisely because of the problems with ANVIL; a lot of the time was wasted getting ANVIL to work properly during the whole experiment. The subject started over again at 22 minutes, so the subject was annotating on and of for 38 minutes. During this period of time the subject annotated 1 minute and 45 seconds of dialogue. This corresponds to 1 minute of dialogue annotated in 21 minutes and 42 seconds; this seems fast compared to the 1 minute of dialogue experts annotate in 25 minutes. It is important to keep in mind though, that the subject only assigned less than half of the annotation tags an expert would in the same length of dialogue. So the subject may seem
faster but has made far less annotations; an expert would make almost perfect annotations like the gold standard, which had over 66 annotation tags assigned to the fragment. The subject of the pilot has only assigned 28 tags that are in many ways the easiest tags to assign; a lot of stalling and no use of multidimensionality. While an expert would go deeper and use the multidimensionality and indicate what segment is an answer to which question and so on. From a total of 28 assigned tags during the pilot experiment, 11 of them were completely wrong. There were 3 partly correct annotations; according to the previously mentioned parent-child relation and 14 correct annotations. The number of errors made compared to the gold standard is hard to measure; do you only measure on the basis of the annotated tags and check those for errors or do you also want to count all the “forgotten” assigned tags of the subject as errors? Concerning the inexperienced subject the tags that are assigned will be checked for errors and the type of errors made. As noted before, all of the stalling tags were filled in correctly, with a lot of confidence of the subject. To find out what goes wrong in the reasoning and errors made by the subject, incorrect answers are compared with the correct DIT++ tags of the gold standard.

**Changes in the experiment**

Firstly, notes about the help function usage must be done better. The experimenter must sit more closely to the subject to notate the time in the dialogue with in addition the segment and the dimension or function for which help is consulted. The multidimensionality of the model is something that should be really pointed out to the subjects, so they will make more complete annotations. The instructions of the experiment are adjusted so the subject is aware of this multidimensionality and will apply it during the experiment.

The content of the DIT++ model is not something that is fresh in the memory of the subjects; the material was discussed over a half a year ago. Therefore the subject will be given a short overview of the DIT++ model to freshen up his or her memory. As mentioned before the synchrony of the text in ANVIL and the speech sound could not be achieved, consequently the subject had to read the text on the screen. Because of that the subjects were not aware of the gender of participant A, the subject in the dialogue. Also, the subject will miss the intonation of the utterances.

In addition to the screenplay for the pilot experiment all above-mentioned remarks are taken into account in doing the real experiment.
3. RESULTS

Unfortunately, for the real experiment only two with DIT++ experienced students were willing to take part. As third subject a master student from the Tilburg University’s Human Resource Studies took part in the experiment. The subject had equally experience with English because her master course is in English. The subject is familiar with the analysis of dialogues because this takes up a big part of her master thesis. Like the other subjects she is acquainted with the experimenter so she was comfortable with thinking aloud. To make her familiar with the DIT++ model she received the same lessons about the model as the other subjects did in their lectures. For the rest of the procedure she was treated the same way as the other participants.

The quantitative numeric results of the experiments are given in the table below.

<table>
<thead>
<tr>
<th>Participant number</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation time per one minute of dialogue (in minutes)</td>
<td>20:00</td>
<td>18:14</td>
<td>28:00</td>
</tr>
<tr>
<td>Use of help function in ANVIL</td>
<td>1</td>
<td>14</td>
<td>20</td>
</tr>
<tr>
<td>Number of annotations possible</td>
<td>106</td>
<td>111</td>
<td>79</td>
</tr>
<tr>
<td>Number of annotations made</td>
<td>99</td>
<td>86</td>
<td>62</td>
</tr>
<tr>
<td>% of annotations possible</td>
<td>93.4</td>
<td>77.5</td>
<td>78.4</td>
</tr>
<tr>
<td>Number of partly correct annotations</td>
<td>22</td>
<td>36</td>
<td>29</td>
</tr>
<tr>
<td>Number of correct annotations (parent-child)</td>
<td>9</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>Total number of correct annotations</td>
<td>31</td>
<td>48</td>
<td>31</td>
</tr>
<tr>
<td>% of correct annotations possible</td>
<td>29.3</td>
<td>43.2</td>
<td>39.2</td>
</tr>
<tr>
<td>% of correct annotations made</td>
<td>31.3</td>
<td>55.8</td>
<td>50.0</td>
</tr>
<tr>
<td>Number of erroneous annotations</td>
<td>67</td>
<td>38</td>
<td>31</td>
</tr>
</tbody>
</table>

Subject #1 filled in 99 of the 106 annotations possible in the part of the dialogue they processed in one hour. There are 31 correct annotations of the 99 annotations made; this is 29.3 percent of the 106 possible correct annotations from the gold standard, and 31.3 percent of the number of annotations made. The participants vary greatly in the time that they annotate one minute of dialogue. Experts need twenty-five minutes to correctly annotate one minute of dialogue; the subjects in this experiments annotate faster. In contrast, their
percentage of correct answers is not even close to one hundred percent, of all of the filled in annotations only 31.3 to 55.8 percent was correct or partly correct (according to the parent-child relation mentioned before).

The consultation of the help function varies greatly among subjects; participant #1 only used the function once. The participants who used help have 50% or more annotations correct, but fill in less than 80% of the practicable (correct) annotations from the gold standard. Subject #1 did not annotate as effective as the other two subjects; in terms of the number of annotations that could have been filled in and the number of correct annotations.

Correct and incorrect annotations

The subjects were also asked to rate the certainty of their choice of a tag. For this analysis the results of the subjects are combined to get an overall picture. From all erroneous answers the certainty values were collected and the zero values are not taken into account because that means that the subject forgot to fill in the certainty. The top three of most wrongly chosen tags are represented in the table below.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Number of times assigned</th>
<th>Certainty mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>inform</td>
<td>25</td>
<td>3.8</td>
</tr>
<tr>
<td>setQuestion</td>
<td>20</td>
<td>4.2</td>
</tr>
<tr>
<td>Confirm</td>
<td>8</td>
<td>3.5</td>
</tr>
</tbody>
</table>

The tags inform and setQuestion are obviously the hardest to correctly assign. The description of the two tags must seem suitable and appealing for a lot of segments. The means of their certainty are rounded both four; the second highest number on the certainty scale. The third place which is for confirm is chosen only eight times, a lot less than the two highest. The highest mean of the wrong annotations is 4.5 for turnAccept. These results show that the subjects thought they were annotating correctly when they filled in these certainties; with a lowest score of one and none of the means coming close to that.

Next, it is also important to know what tags of the DIT++ model were correctly ascribed. The top three of these correct annotations are shown in the table below.
Table 3.
Top three correctly assigned tags and their certainty scale mean

<table>
<thead>
<tr>
<th>Tag</th>
<th>Number of times assigned</th>
<th>Certainty mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>stalling</td>
<td>49</td>
<td>4.7</td>
</tr>
<tr>
<td>inform</td>
<td>12</td>
<td>3.8</td>
</tr>
<tr>
<td>informElaborate</td>
<td>9</td>
<td>4.2</td>
</tr>
</tbody>
</table>

Stalling was the easiest for the subjects to correctly assign during the experiment. It must be said that stalling has a very frequent occurrence in the DIAMOND dialogue. Segments of dialogue that should be ascribed as stalling are very distinguishable; utterances like “ehm” or “er” are among other tags always annotated as stalling. Most likely, this is merely attributed to the clear and short definition and examples of stalling:

“S needs a little bit of time to formulate an utterance.”
(Definition of stalling on the DIT++ website (http://dit.uvt.nl/))

““Let me see...”, “Erm””
(Example in the help function of the ANVIL program)

In addition, stalling also was rated with very high certainty mean (4.7), most likely because of the clear definition of the tag. When a subject would encounter an utterance like “uhm” again it will be highly unlikely that he or she will consult help again; it is clear that this is stalling. It is remarkable to see that inform also is in the top three of correct annotations; as it was also on the number one spot in the wrongly made annotations. This points at the fact that the inform definition is unclear and the subjects do not know whether to annotate with this tag or not. It is not a matter of gambling, because the erroneously assigned informs have a certainty mean of 3.8, which is relatively high. The subjects were partly sure inform was the right tag to pick. To compare this seemingly unclear inform to stalling the definition and examples of inform are shown below:

“Dialogue act where the speaker, S, wants to make certain information known to the addressee, A; S believes that the information is correct. For example: “The 6.34 to Breda leaves from platform 2.”
(Definition and example of inform in the help function of ANVIL)

This definition might seem trivial to annotation experts, but may appear very vague to a novice. As one subject remarked on the inform definition in the evaluating talk: “For me, communicating is totally based on providing each other information. Whether the info is true
or not cannot be perceived by an annotator, it is only known by the sender.” The inform definition should be more articulate so the tag does not seem too suitable for the wrong segments.

Surprisingly a tag with a similar name, informElaborate, is also one of the most correctly assigned. Likely because of the addition elaborate, which is good for the self-explanatory character of the name. The term elaborate triggers a certain expectation of the tag, where the consultation of help can contribute to the decision making.

The most difficult part of the research is to find out where and why a correct or incorrect decision is made on the basis of the thinkaloud coded protocols of the experiment. Furthermore it is also important to know why the subject was not able to get to the correct annotation or why his/her annotation is incomplete regarding the multidimensionality. The wrong or incomplete annotations are mainly made because of three reasons:

- The subject was thinking and reasoning in the right direction, but could not come to the right conclusion;
- the subject made a completely wrong decision and there is no clear cause why his/her decision went that way;
- the subject has already made one annotation and “forgets” to make a second or third one.

An example of the first reason can be seen in a part of the protocol below:

"informatie geven dat je dus eigenlijk al… eigenlijk je vraag wil inluiden even denken
dan ga ik eventjes kijken (..)
eehmmm
ja nou het zou dus kunnen zijn inform justify
dus god ik ben de hele tijd zo stil geweest en ik ben pas net begonnen en nu heb ik al een vraag dus dat ze dat..
ja, ik denk dat het die van justify is rechtvaardigen van ik heb nu al een vraag en ik ben nog niet eens.. Dat weet ik niet zo heel zeker"

The subject knows that the utterance is a sort of inform, but suspects it is something more than just inform, which is wrong; the correct annotation is inform, while the subject chose informJustify.

The second reason has very different expressions; the answer is not even close to the correct annotation and it is not understandable why.

"ik probeer nu opdracht 2 te doen"
The right annotation here is inform, but the subject chose indirectSetQuestion, which is slightly odd because there is no question mark in sight.

The third reason is caused by inaccuracy of the subject; one annotation is made but the subject fails to make another one.

"prima" ja nou eh, nou is het misschien wel agreement ja dat weet ik niet zo zeker dat is weer stalling "en kies nu het nummer waarnaar.."

The problem with the easiness of stalling is that the subject rushes by these utterances and forgets to consider other possible annotations. Utterances like “eh” are quickly annotated as stalling, but often they also need other annotations such as turnAccept or turnKeep. Maybe that is because the DIAMOND dialogue contains a lot of stalling and the subjects might get irritated with that. Many dialogue parts require a second or third look when it comes to annotating and maybe it is wise to point out the multidimensionality of the model to the subjects. After all, the multidimensionality is what the model is known for and what distinguishes the model from others.

The use of the help function

It is also important to know for what dimensions the help function in ANVIL was used and how many times. It was difficult for the researcher to find out for what part of the dimension help was used; when this was unclear only the name of the dimension was taken down. When the researcher was able to find out for what tag the function was used it was because the subject spent more time reading the explanations and examples. It is safe to say that these definitions are probably the trickiest ones. The dimensions of the DIT++ model vary widely in the number of times of occurrence in a dialogue and in the number of tags that are a part of them. Help was consulted the most for the information providing functions and two of the most consulted tags from that dimension were confirm and elaborate. The coded protocols are helpful to find out why they are studied more thoroughly and longer and why these tags require the help function in the first place. One subject consulted help because he doubted
between the previously discussed inform and confirm. But before he clicked on the help-button the subject already mentioned the correct answer:

“...autofeedback en er zit ook confirm in.”

He consulted help to confirm afterwards his expectation, just to be sure.

The use of the help function gives no guarantee that a correct annotation will be made; Inform is on the first place of wrong annotations, on the second place of correct annotations and regularly was looked up in help. InformElaborate is looked up as much as inform but scores high when it comes to be correctly ascribed. The coded protocols show that the help function is used as feedback to confirm the subject is right; the subject looks the definition up just to be sure. Or the subject has not yet made a decision about the segment and decides which tag he/she will assign. The latter is the conventional concept of what a help function is for, but proves to be wrong. The help function in this experiment is also used to freshen up the subject’s memory and affirm their choices.

Evaluating talk
After the hour of annotating the subjects where asked their opinion about the use of the DIT++ model. They were free to make remarks about what they thought the model was missing. In this way, not only implicit suggestions for improvement of the model come to light but also explicit suggestions. All three of the subjects noted that they missed a tag that annotated politeness, as also appeared in the think aloud protocols. The subjects found other tags that expressed sociality in the model like thanking and apologizing but missed politeness. A comment on the explanation of inform is already discussed. Another comment was that there is no check answer in response to a the checkQuestion and that the name of the dimension discourse structure management was not self-explanatory. Overall the subjects thought they did well, aside from a few problems with finding the right tag.

4. DISCUSSION
The usability of the DIT++ model was investigated in a think aloud experiment; now it is time to answer the questions concerning this usability. To what degree does this model meet the ISO criteria of usability?
Effectiveness is the first criterion of the ISO definition; this concerns the accuracy and completeness with which the specified users can reach their goals. The usage of the DIT++ model by novices is far from accurate or complete; the subjects were not able to annotate over 60 percent of their annotations correctly and compared to the gold standard they failed to make 56 percent or more of the possible annotations. This reflects a low awareness of the possibilities the multidimensionality of the DIT++ model offers. The wrongly annotated segments were due to too extensive or vague explanations and examples in the model.

Second, there is efficiency, a part where the DIT++ model scores highly. The subjects were able to annotate very quickly; they made more annotations than an expert would do in one hour. The DIT++ model supports annotating effortlessly.

Finally, there is user satisfaction; the comfort and acceptability of its users. The users where asked what their opinion was about the model; a few remarks concerning the descriptions and suggestions for new tags were the only remarks users gave. Aside from their own doubts between two alternatives the model gave them enough information to keep on annotating. The value of this opinion is very relative; the users never used another annotation model before and are little experienced. On the other hand, when a model is so efficient to use for novices it will be very efficient for experts. Concerning the certainty scale means the subjects were confident that they were annotating correctly, which must have been satisfying.

The adjustments that could be made in the DIT++ model must be focused on the comprehensibility of the terms used in the model. The explanations of some of the tags should be less abstract and more concrete with clear examples that the users encounter in their daily lives; like the self-explanatory terms **greeting** and **stalling**. To outsiders this may be obvious, but this kind of problems is often overlooked by experts. They master their skills to such extent that they are not able to enter a novice’s thoughts. Experts maybe rarely consult the model and when they do, just a few words can bring them back on the right track. For novices, when the explanation in the model in unclear they will enter a vicious circle where they keep making the same errors over and over; because of the lack of sufficient examples or explanations. The subjects all indicated in their interview and showed with their certainty scale ratings that they were satisfied with the use of the model in terms of correct answers that they were able to give. But when they do not understand the model without knowing it they will make systematic errors. When an explanation is not sufficient for a user; more examples will give more insight. Maybe this problem will occur to a lesser extent when this experiment is done with subjects of whom English is their first language. The current subjects all mastered English very well, but they were not native speakers. The multidimensionality of the
The DIT++ model is a taxonomy with hierarchically organized tags, but the users’ decisions do not follow the route the model ideally wants them to take. Users follow a bottom-up approach towards the model; they reason from tag to dimension instead of the other way around. The use of ANVIL enforces them to think unnaturally. For example, the user knows a segment must be assigned as *stalling* but first has to figure out under what dimension *stalling* is placed in order to make this annotation. He or she has to click on the row of the right dimension and only then can find the right tag. Therefore the efficiency of the model is set back; the DIT++ model ideally wants its users to firstly decide to what dimension a segment belongs and then choose a tag from that dimension. There were no complaints from the subjects, but maybe that is because it was ordered them to do it in that manner. And maybe subjects thought this order of annotating was inherent to working with ANVIL. In practice that is not how it always works, maybe when assigning a tag for the first time; users may know an utterance has something to do with the dimension *discourse structure management* and then decide that a segment should be tagged *opening*. But when a user finds such an utterance for the second time, *opening* is the first that comes to mind and the user has to think about what dimension that tag was in.

It is important to notice that a usability analysis, notably the thinking aloud method, is very well applicable to the evaluation of an annotation scheme like the DIT++ model. Maybe in the future similar experiments can be performed with experienced annotators or on other dialogue annotation models.

A suggestion for further research; this experiment could be very interesting in terms of high and low context cultures as described by Hall (1976). The distinction between high and low context cultures is among other things distinguished by the manner of communicating. In a low context culture like the Netherlands, spoken language has an emphasis on communication; people say what they mean. A more recent study from Hall, De Jong and Steehouder (2004) showed that collectivistic cultures (high context culture) and individualistic cultures (low context cultures) differ in two ways when it comes to usability testing. Individualistic subjects provide a more direct feedback and tend to provide comments that are not related to the task. High context culture subjects maybe will annotate low context...
culture dialogues like the DIAMOND dialogue completely differently and provide different and more relevant information about the usability of the model.

As the DIT++ model is open to changes and adjustments, it may evaluate into an ideal model for dialogue annotation.
References


Appendix A: The DIT++ taxonomy
(Source: http://dit.uvt.nl, November 6, 2008)

The hierarchical relations in the taxonomy, indicated by indentation, represent relative degrees of \textit{specificity} of dialogue acts, in the sense that a more specific act has stronger preconditions than a less specific act (which dominates it in the taxonomy); in other words, the preconditions of more specific dialogue act logically entail those of any dominating act in the hierarchy. A communicative function inherits all the preconditions of its ancestors in the hierarchy. For instance, a Check Question is more specific than a Propositional Question because it has an additional precondition, concerning the speaker's expectation of the answer. Similarly, a Confirm(ation) is more specific than a Propositional Answer. This is reflected in the taxonomy by Check Question being dominated by Propositional Question, and Confirm by Propositional Answer.

\textit{General-Purpose Communicative Functions}

- \textit{Information Transfer Functions}
  - \textit{Information Seeking Functions ("Questions")}
    - \textit{Direct Questions}
      - Set Question
      - Propositional Question
        - Check Question
          - Nega-check
      - Alternatives Question
    - \textit{Indirect Questions}
      - Indirect Set Question
      - Indirect Propositional Question
      - Indirect Alternatives Question
  - \textit{Information Providing Functions}
    - \textit{Informing Functions}
      - Uncertain Inform
        - Inform
          - Agreement
          - Disagreement
• Correction
  ▪  *Informs with a Rhetorical, Emotional, or Evaluative Function*

  **Rhetorical:**
  ▪  Elaborate
  ▪  Exemplify
  ▪  Justify
  ▪  ...

  **Emotional/evaluative:**
  ▪  Warning

  ▪  *Answer Functions*
    ▪  Uncertain Set Answer
      ▪  Set Answer
    ▪  Uncertain Propositional Answer
      ▪  Uncertain Confirm
      ▪  Uncertain Disconfirm
      ▪  Propositional Answer
        ▪  Confirm
        ▪  Disconfirm

  ▪  *Action Discussion Functions*
    ▪  Commissives
      ▪  Offer
        ▪  Promise
        ▪  Address Request
          ▪  AcceptRequest
          ▪  DeclineRequest
        ▪  Address Suggestion
          ▪  AcceptSuggestion
          ▪  DeclineSuggestion
• Other commissives, as expressible by means of performative verbs or by addressing other directives
  o Directives
    • Indirect Request
    • Request
    • Instruct
    • Address Offer
      • AcceptOffer
      • DeclineOffer
    • Suggestion (a.k.a. Open-option)
    • Other directives, as expressible by means of performative communicative verbs, such as Advice, Proposal, Permission, Urge,…

Dimension-Specific Communicative Functions

• Activity-Specific Functions
  o Functions, expressible either by means of performative verbs denoting actions for performing activities in a specific domain, or by means of graphical actions such as highlighting, or pointing to something in a picture. For example:
    o Open Meeting, Suspend Meeting, Resume Meeting, Close Meeting (in meeting situations)
    o Bet, AcceptBet
    o Congratulation, Condolance
    o Hire, Fire, Appoint,… (in a human resource management domain)
    o Show, Highlight, Point, List,… for performing graphical/multimodal dialogue acts

• Dialogue Control Functions
  o Feedback Functions
    • Auto-Feedback Functions
      • Positive Auto-Feedback (= Unspecified Positive)
        • Attention Positive Feedback
          • Perception Positive Feedback
- Interpretation Positive Feedback
  - Evaluation Positive Feedback
    - Execution Positive (Overall Positive)
  - Negative Auto-Feedback (= Unspecified Negative)
    - Execution Negative Feedback
      - Evaluation Negative Feedback
        - Interpretation Negative Feedback
          - Perception Negative Feedback
            - Attention Negative (Overall Negative)
  - Negative Allo-Feedback (= Unspecified Negative)
    - Evaluation Negative
      - Execution Negative
- Attention
- Perception
- Interpretation
- Evaluation
- Execution

\textit{Allo-Feedback}

\textbf{Feedback-Giving Functions}
- Positive Allo-Feedback (= Unspecified Positive)
  - Perception Positive
    - Interpretation Positive
      - Evaluation Positive
        - Execution Positive (Overall Positive)
  - Negative Allo-Feedback (= Unspecified negative)
    - Evaluation Negative
      - Execution Negative
        - Interpretation Negative
          - Perception Negative
            - Attention Negative

\textbf{Feedback Elicitation Functions}
- Attention
- Perception
- Interpretation
- Evaluation
- Execution
Allo-Feedback

- Interaction Management Functions
  - Turn Management

  **Turn-unit-initial functions**
  - Turn Accept
  - Turn Grab
  - Turn Take

  **Turn-unit-final functions**
  - Turn Assign
  - Turn Keep
  - Turn Release

- Time Management
  - Stalling
  - Pausing

- Contact Management
  - Contact Check
  - Contact Indication

- Own Communication Management
  - Error signaling
    - Retraction
      - Self-correction

- Partner Communication Management
  - Completion
  - Correct-misspeaking

- Discourse Structure Management
  - Opening
  - Preclosing
  - Topic Introduction
  - Topic Shift Announcement
    - Topic Shift
- **Social Obligations Management Functions**

  <>

  o **Salutation**
    - Initial greeting
    - Return greeting
  
  o **Self-introduction**
    - Initial self-introduction
    - Return self-introduction
  
  o **Apologizing**
    - Apology
    - Apology-downplay
  
  o **Gratitude Expression**
    - Thanking
    - Thanking-downplay
  
  o **Valediction**
    - Initial goodbye
    - Return goodbye
Appendix B: The guidelines of the DIT++ model
(Source: http://let.uvt.nl/general/people/bunt/docs/AnnoGuide.html)

Annotation guidelines for applying DIT dialogue act tags

Dialogue act annotation is about indicating the kind of intention that the speaker had; what kind of thing was he trying to achieve? When participating in a dialogue, this is what agents are trying to establish. The first and most important two guidelines follow from this.

1. First and most important guideline: "Do as the Addressee would do!"
When assigning annotation tags to a dialogue utterance, put yourself in the position of the participant at whom the utterance was addressed, and imagine that you try to understand what the speaker is trying to do. Why does (s)he say what (s)he says? What are the purposes of the utterance? What assumptions does the speaker express about the addressee? Answering such questions should guide you in deciding which annotation tags to assign, regardless of how exactly the speaker has expressed himself. Use all the information that you could have if you were the actual addressee, and like the addressee, try to interpret the speaker's communicative behaviour as best as you can.

2. Second and equally important guideline: "Think functionally, not formally!"
The linguistic form of an utterance often provides vital clues for choosing an annotation, but such clues may also be misleading; in making your choice of annotation tags you should of course use the linguistic clues to your advantage, but don't let them fool you - the true question is not what the speaker says but what he means.
For example, SetQuestions are questions where the speaker wants to know which elements of a certain domain have a certain property. In English, such questions often contain a word beginning with "wh", such as which as in Which books did you read on your holidays? or where in Where do your parents live?. But in other languages this is not the case; moreover, even in English not all sentences of this form express a SetQuestion: Why don't you go ahead is for instance typically a Suggestion rather than a question.
Similarly, PropositionalQuestions are questions where the speaker wants to know whether a certain statement is true or false. Such sentences typically have the form of an interrogative statement, such as Is The Hague the capital of the Netherlands? or Do you like peanut butter? But not all sentences of this form express a PropositionalQuestion; for example, Do you know what time it is? functions most often as in IndirectSetQuestion (What time is it?), and Would you like some coffee? is an Offer; Shall we go? is a Suggestion.

3. Another important general guideline is: "Be specific!"
Among the communicative functions that you can choose from, there are differences in specificity, corresponding with their relative positions in hierarchical subsystems. For instance, a CheckQuestion is more specific than a PropositionalQuestion, in that it additionally carries the expectation that the answer will be positive. Similarly, a Confirm is more specific than a PropositionalAnswer, in that it carries the additional speaker that the addressee expects the answer to be positive.
In general, try all the time to be as specific as you can. But if you're in serious doubt about specific functions that you could choose between, then simply use a less specific function tag that subsumes the more specific tags.

Standard speech act theory regards indirect speech acts, such as indirect questions, as just an indirect form of the same illocutionary acts. By contrast, the DIT++ taxonomy incorporates the idea that indirect dialogue acts signal subtly different packages of beliefs and intentions than direct ones. For example, the direct question What time is it? carries the assumption that the addressee knows what time it is, whereas the indirect question Do you know what time it is? does not carry that assumption (it does at least not express that assumption; in fact it questions it).

5. On implicit functions: "Do not code implicit communicative functions, that can be deduced from functions that you have already assigned."
Implicit communicative functions occur in particular for positive feedback. For example, someone answering a question may be assumed to (believe to) have understood the question. So any time you annotate an utterance as an answer (of some sort), you might consider annotating it also as providing positive feedback on the interpretation of the question that is answered. Don't! It would be redundant.
Notice also that the definition of a positive (auto-) feedback act concerning interpretation stipulates that the speaker wants the addressee to know that he (speaker) has understood the question. A speaker who answers a question does not so much want to tell the addressee that his question was understood -- that's just a side-effect of giving an answer, that no speaker can avoid. Similarly for reacting to an offer, a request, a suggestion, etc.

6. Guidelines for the annotation of feedback functions.

Negative feedback, where the speaker wants to indicate that there was a problem in processing a dialogue utterance, is always explicit and as such mostly easy to annotate.

6.1 Implicit and explicit positive feedback.

Positive feedback is sometimes given explicitly, and very often implicitly. Examples of explicit positive auto-feedback are the following utterances by B, where he repeats part of the question by A:

A: *What time does the KLM flight from Jakarta on Friday, October 13 arrive?*
B: *The KLM flight from Jakarta on Friday, October 13 has scheduled arrival time 08.50*
B: *The flight from Jakarta on Friday has scheduled arrival time 08.50*
B: *The KLM flight from Jakarta on October 13 has scheduled arrival time 08.50*
B: *The flight from on October 13 has scheduled arrival time 08.50*

In such cases, the utterance by B should be annotated as having, besides the general-purpose function SetAnswer in the Activity dimension, also a function in the Auto-Feedback dimension (see below).

By contrast, the short answer: *At 08.50* would carry only implicit feedback information, and should therefore, following Guideline 5, not be coded in the Auto-Feedback dimension.

6.2 Levels of feedback.

The DIT++ taxonomy distinguishes 5 levels of feedback:
1. participant A pays attention to participant B's utterance.
2. A *perceives* B's utterance, i.e. A recognizes the words and nonverbal elements in B's contribution.

3. A *understands* B's utterance, i.e. A assigns an interpretation to B's utterance, including what A believes B is trying to achieve with this utterance (what are his goals and associated beliefs about the task/domain and about A).

4. A *evaluates* B's utterance, i.e. A decides whether the beliefs about B that characterize his understanding of B's utterance, can be added to A's model of the dialogue context, updating his context model without arriving at inconsistencies.

5. A *executes* B's utterance, i.e. A performs actions which are appropriate for achieving a goal that he had identified and added to his context model. (For instance, executing a request is to perform the requested action; executing an answer is to add the content of the answer to one's information; executing a question is to look for the information that was asked for.)

There are certain relations between these levels: in order to execute a dialogue act one must have evaluated it positively ("accepted" it); which is only possible if one (believes to) have understood the corresponding utterance; which presupposes that one perceived the utterance in the first place, which, finally, requires paying attention to what is said. So for instance positive auto-feedback about the acceptance of the addressee's previous utterance implies positive feedback at the "lower" levels of understanding, perception, and attention. *For positive feedback functions a higher-level function is more specific than the lower-level functions.* (Remember that a function is more specific if it implies other functions.)

For negative feedback the reverse holds: when a speaker signals the impossibility to perceive an utterance, he implies the impossibility to interpret, evaluate and execute it. *So negative feedback at a lower level implies negative feedback at higher levels.*

Since, following Guideline 3, you should always be as specific as possible, you should observe the following guideline for annotating feedback functions:

*Guideline 6*: When assigning a feedback function, choose the most specific level of feedback in the case of positive feedback that you feel to be appropriate, and choose the least specific level in the case of negative feedback.

While this guideline instructs you to be as specific as possible, sometimes you'll be in
serious doubt. You may for instance find yourself in a situation where you have no clue whether a feedback signal (such as OK) should be interpreted at the level of interpretation or that of evaluation. In such a case you should use the less specific of the two, since the more specific level would mean that you "read" more into this utterance than you can justify.

In practice, it is often difficult to decide the level of feedback that should be chosen. One of the reasons for this is that the same verbal and nonverbal expressions may be used at most of the levels (with a tendency to signal feedback (positively or negatively) with more emphasis as higher levels of processing are involved). It may happen that you encounter a feedback signal and you have no clue at all at which level you should interpret that signal. In this situation the annotation scheme allows you to use the labels 'Positive' and 'Negative', which leave the level of feedback unspecified.

7. Guidelines for the annotation of Interaction Management functions.

7.1 Turn Management.

General guideline: "Code Turn Management functions only when these are not just implied."

In a spoken dialogue, the participants take turns to speak. (Their nonverbal behaviour is not organised in turns; both participants use facial expressions and gestures more or less all the time.) A turn, that is a stretch of speech by one of the participants, in general consists of smaller parts that have a meaning as a dialogue act; these parts we call "utterances". Turn Management acts are the actions that participants perform in order to manage the allocation of the speaker role. These acts are subdivided into acts for taking the turn (utterance-initial acts) and those for keeping the turn or giving it away (utterance-final acts). Usually only the first utterance in a turn has an utterance-initial function and only the last an utterance-final one. The non-final utterances in a turn do not have an utterance-final function, except when the speaker signals (for example by using a rising utterance-final intonation) that the utterance is not going to be the last one in the turn, that he wants to continue. In that case the utterance has a Turn Keeping function. Except for the first one, the utterances in the turn do not have an utterance-initial function; the speaker does not have to perform a separate act in order to continue; all he has to do is to continue speaking.

When a speaker accepts a turn that the addressee has assigned to him through a Turn
Assign act, the utterance should be annotated as having the utterance-initial function
Turn Accept only when the speaker performs a separate act for the purpose of
accepting the turn, so don't code this when the turn is accepted implicitly by simply
starting to speak.

Similarly, an utterance should be annotated as having the utterance-initial function
Turn Take only if the speaker performs a separate act to that effect. If he just goes
ahead and makes a contribution to the dialogue, without first signalling his intention to
do so, then the utterance should not be marked with an utterance-initial Turn
Management function.

The verbal as well as nonverbal activities that a speaker performs to seize the turn
should be marked as Turn Grabbing, but the utterance that follows after he has seized
the turn should not be marked as having an utterance-initial Turn Management
function.

7.2 Time Management.
When a speaker is buying time, using fillers such as Well,...; Let's see,..., then the
utterance should be annotated as having the Stalling function in the Time Management
dimension. There may be several reasons why a speaker wants to have more time; it
may be that the speaker has trouble completing his current utterance, or that he is
interrupted by some urgent event that requires his attention before he can continue the
dialogue. But it may also be that he needs some time to find some information (for
instance, for answering a question). So when you encounter a Stalling act, you may
well pay attention to the reason why the speaker is stalling. (For instance, Stalling
often goes hand in hand with turn acceptance or turn keeping.) However, don't
speculate; only code additional functions for which you have evidence.

7.3 Topic Management.
During a dialogue, the topic is often changed implicitly, simply by talking about a new
topic. This happens especially if the new topic is closely related to the previous one,
for instance by being a subtopic of the previous topic, or by both being a subtopic of a
more general topic. Implicit topic management should not be encoded; it would be
redundant. Topic Management functions should be annotated only if the speaker explicitly introduces or closes a topic, or signals his intention to do so.

7.4 Contact Management.
The management of contact in the sense of both partners being ready to send and receive messages to and from each other, is important especially in other than face-to-face situations, such as telephone conversations, video-conferencing, and internet chatting.

Note that in many languages expressions used for establishing contact can often be used for other purposes as well, for example for greeting (Hello!). When annotating a dialogue where this happens, the utterance in question should be marked as having both a Contact Management function and a Social Obligation Management function.

7.5 Own Communication Management.
Own Communication Management (OCM) functions should be coded whenever a speaker signals that he made a speech error and/or wants to edit what he is saying. Since this typically requires some extra effort and time, OCM acts often go hand in hand with acts whose function is to win time, such as hesitations (Ehm...), which have a Stalling function. See also 7.2.

7.6 Partner Communication Management.
Partner Communication Management (PCM) functions should be coded whenever a speaker signals that he believes the addressee made a speech error or has difficulty in completing an utterance, for instance being unable to recall a name or to find the right words to express something. The use of dimension-specific PCM functions for this purpose is typically only possible by interrupting the dialogue partner or in immediate response to a partner utterance.

7.7 Dialogue Structuring.
These functions should be coded only when the speaker explicitly signals something about his intention to open or close the dialogue, or to continue in a particular way.

Across the board, the following guideline applies to the encoding of Interaction
Management functions:

Guideline 7: *"Code only explicit Interaction Management functions."*

8. Guidelines for annotating Social Obligation Management (SOM) functions. Utterances that serve a 'social' purpose such as greetings, thanks, and apologies can often be used for other purposes as well. Greetings like *Hello!*, for example, can be used also for establishing contact (Contact Management function) and/or for opening a dialogue (a Dialogue Structuring function). Also, an expression of thanks can be used to signal that the speaker wants to soon end the dialogue (Dialogue Structuring function PRE-CLOSING), and can also be used for overall positive feedback. In such cases, the utterances should be coded with the appropriate functions in all these dimensions.

Guideline 8: *"When coding an utterance as having a SOM function, look out for additional functions in other dimensions."*
### Appendix C: Example of coded protocols

<table>
<thead>
<tr>
<th>tokens</th>
<th>utterances</th>
<th>code DIT++</th>
</tr>
</thead>
<tbody>
<tr>
<td>hmm...</td>
<td>ehm dit is de eerste uiting ik denk dat ie is om de eh de beurt te behouden om de tegenstander eh de deelnemer de tijd te geven om na te denken (ANVIL) information seeking, information providing information providing nee nee nee dit is het niet dan denk ik dat het ehm.. contact zou kunnen zijn en dan is het set answer dat denk ik wel om hem de tijd te geven om een antwoord te kunnen bedenken daar ben ik, niet echt zeker van dat ben ik tot op zekere hoogte toch nog 2 (ANVIL)</td>
<td>turnkeep</td>
</tr>
<tr>
<td>even kijken hoor.</td>
<td>“even kijken hoor” dat is hetzelfde, om hem nog tijd te geven om na te denken dat hoort nog steeds bij &quot;hm&quot; set answer, daar ben ik, nou iets meer zeker van dus ik zet ‘m op 3</td>
<td></td>
</tr>
<tr>
<td>nou heb ik al een vraag. ja</td>
<td>er is er al eentje ingevuld &quot;nou heb ik al een vraag&quot; dat is inform &quot;ja waarmee kan ik..“ (…) Dat is om de beurt te.. In te luiden (…) dat is in ieder geval wat, turn initial (ANVIL) turn initial (ANVIL) dat is een turn grab, dan omdat hij de beurt zo snel mogelijk wil pakken daar ben ik vrij zeker van dus ik zet ‘m op 4</td>
<td>turntake &lt;OK&gt; turngrab</td>
</tr>
<tr>
<td>waarmee kan ik je helpen?</td>
<td>en dan komt er nog een vraag &quot;waarmee kan ik..” Oh &quot;waarmee kan ik je helpen?&quot; dat is een vraagstelling, informatie zoeken dus ik denk dat er dat staat dat is ah kijk, dat is informatie zoeken, set question is dat daar ben ik eh 100% zeker van dat dat (…) is om informatie te zoeken de volgende uiting, dat is om de tijd te geven om een vraag te formuleren en uit beleefdheid denk ik dus dat zijn er meerdere, dat mocht he? experimenter: Ja (ANVIL)</td>
<td>&lt;POLITE&gt;</td>
</tr>
<tr>
<td>ik wil graag een kopie maken ook van de achterkant.</td>
<td>dat is om een vraag te stellen, daar ben ik ook vrij zeker van het is ook uit beleefdheid</td>
<td>&lt;POLITE&gt;</td>
</tr>
</tbody>
</table>