

Vidiam Functional Specification for IMIX Demo-2

IMIX - VIDIAM

October 6, 2005

1 Preliminaries

We wish the dialogue manager to present the system in a "search engine" type of way, making explicit that the question answering mechanics are imperfect, search engine like, and not really understood by the dialogue manager. We decided that a prompt presenting an answer should be stated like: "I found the following answer: ...", instead of just presenting the answer. See also the functional specification.

The basic search task is described as follows:

```
1 obtain_domain_question
  DO ANSWER_STRATEGY
2 obtain_user_response
  if user_response_indicates_answer_is_bad:
3   obtain_reformulation_or_followup
    DO ANSWER_STRATEGY
  if user_response_is_followup_question:
    DO ANSWER_STRATEGY
  goto 3
```

ANSWER_STRATEGY:

```
  if question_analysis_ok:
    if answers_with_good_confidence:
      display_answers
    else:
      if can_ask_clarification_question:
4       ask_clarification_question
        DO ANSWER_STRATEGY
      else:
        prompt_no_answer_found
        DO ANSWER_STRATEGY
  else:
5   ask_reformulation
    DO ANSWER_STRATEGY
```

The numbered steps are the steps where a user response is expected (ask/obtain actions).

Each ask/obtain action tries to obtain the proper information from the user. When information is missing, the system will start a clarification dialogue until the information is supplied (or the clarification fails). We will call this clarification process a *representation-level* clarification dialogue. Such a dialogue may also be initiated by the user.

The user should be aware of the task progress, and should be able to direct it. In particular, the user should be able to restart the process (i.e. go back to step 1), and should be able to skip answering a system question (such as step 4, but also during any representation-level clarification dialogue) and go back to a meaningful user-initiative state (such as step 2).

The information state will contain some (epistemic) information about the following things:

- multimodal dialogue history
- progress of the tasks being performed
- the state of the communication itself, enabling conversation about and problem solving of speech, words, pictures, pointing, word relations (i.e. afkorting),
- meta information about queries in general (when a query is made, quality of a query, user perception of query), and each query made (confidence, no of hits, document source).

Ontological information in the dialogue manager will comprise the following:

- task knowledge
- conversational knowledge (how to talk about communication and solve communication problems)
- domain information

An overview of possible dialogue acts follows. All dialogue acts are candidates for being supported in the first demonstrator version. This classification is a pragmatic one, in which:

(1) the term "dialogue act" is interpreted as IMIX-specific operations on the system's information state, while

(2) the user is sufficiently able to identify the distinct intentional meaning of each act.

We associate the different acts with the dialogue act as defined in the functional specification. Here, a dialogue act is a 3-tuple (*acttype*, *contenttype*, *content*). *acttype* follows the Paradime classification, *contenttype* specifies the specific function that the act has within the IMIX dialogue. These dialogue acts are the same as those that are sent to Imogen, except that content will always be replaced by a literal text string, except for the "QA Answer" act.

2 Basic information transfer, grounding

2.1 Question (system, user)

acttype: WHQ,YNQ,REQ

contenttype: question

content: expression denoting items in information state

A system Question is a request for a specific piece of information in the information state.

2.2 Representation Request (system, user)

acttype: WHQ nauf,YNQ nauf,REQ nauf

contenttype: repreq_speech (system,user)

. repreq_dialogue (system,user)

. repreq_pen (system)

. rereq_parse (system)
. rereq_qa (system)
content: description of the problem details

Request for repetition and re-representation. It is a signal that something was not understood, and that exactly the same information has to be presented again, possibly using another style or information channel. The request may contain information about the desired representation.

For Demo-2, this covers not only the standard speech (and pen) recognition problem solving, and other understanding problem solving.

Note: it may be possible that non-understanding is signalled without meaning a request to repeat the information.

Note: the subtype, speech, dialogue, etc should become a parameter rather than a subtype later in the project.

Examples:

rereq_speech: "What did you say?"
rereq_pen: "Did you point to the arm or the hand?"
rereq_parse: "What does the word "RIS" mean?"
rereq_dialogue: "Did you ask me a question?"
rereq_qa: "Could you reformulate your question?"

2.3 Confirmation (system, user)

acttype: YNQ pauf, YNQ palf, INF pauf, INF palf
contenttype: confirm_explicit
. confirm_implicit
. confirm_ok
content: description of items in information state

Acts that are meant to verify any item in the common ground. This may be seen as similar to the representation request. There is a gliding scale, going from: "what did you say?" to "did you say X or Y?" (which are representation requests) to "did you say X?" to "So, you say X." to "Ok I understand." (which are confirmations).

Note: we may distinguish subtypes like those found in representation request. It would be nice to combine representation requests and confirmations.

Examples:

confirm_implicit: "I asked you a question" (possible confirmation when user has unexpected behaviour)
confirm_explicit: "Did you ask me a question?" (note that this happens to be a yn question and therefore may be classified as confirmation as well as representation request).
confirm_ok: "Ok"

2.4 Inform (system, user)

acttype: INF nalf
contenttype: inform
content: (system) expression denoting items in information state
. (user) the parsed text

Either party may spontaneously inform about a misunderstanding or possibly missing information. Informs usually either point to misunderstandings (the act will also be a correction), or are answers to requests (the act will also be an answer).

3 Augmentations of other types

These types do not fully specify a dialogue act but serve as supplemental information. This means these types only occur together with another type. They provide supplemental information which enables the system to better understand the purpose of the act.

3.1 Answer / Reply (system, user)

acttype: INF

contenttype: answer

content: pointer to the question/request being answered

An utterance that fulfills a previous specific (information) request of the other participant. It always also has another act type. For example, a request to repeat a question would have a reply that is also a question; a reply to a request for information would also be an inform; a reply to a request for reformulation would be a reformulation.

Examples:

"I asked you a question" (answer to dialogue state representation request)

3.2 Reformulation (user)

acttype: INF nalf

contenttype: reformulate

content: reference to the utterance/act being reformulated

Reformulation or repetition of a previous utterance segment or act in order to better convey the information contained therein. A reformulation always has the same act type as the act it is trying to reformulate.

Example: reformulation of a qa domain question.

4 QA Query and related

4.1 QA Question (user)

acttype: WHQ td, YNQ td, REQ td

contenttype: q_domain_new (new or initial domain question)

. q_domain_clarif (clarification domain question)

. q_domain_followup (follow up domain question)

content: the parsed text (with references)

The "initial user question", in the case of IMIX, is a question, requesting factual, general technical information (a "task domain" question).

Other task domain questions may be asked as a reaction to the system's answer. The questions concern the application domain. These questions need not be self contained, but are reactions to the system's answer, and implicitly or explicitly refer to previous utterances. Many multimodal and visual utterances are likely to fall in this category.

q_domain_new: The question is likely to be self contained. While a question describing a specific situation (a diagnostic question) need not be supported by IMIX, the question may contain references to the asker (like: "Can I get RSI?"). A QA Question may be a request rather than a question, for example, "Give me a list of ...". The QA may not be able to handle many of these questions, so we consider this an exceptional case which we will not handle separately.

q_domain_followup (follow up question): pose a new, related, question. The user wants more information because either the answer presented information that changed the user's information need, or the user proceeds to a next step in his search agenda. We may assume that the new question will be the new topic under discussion, using the previous question(s) as context to interpret it.

q_domain_clarif (domain clarification question): a request to clarify the answer. Here, the user is not quite satisfied with the answer, and wants the answer to be clarified. The user may doubt the validity or correctness of the answer. An appropriate system reaction would for example be to show more context around the answer.

4.2 Query Clarification Request (system)

acttype: WHQ td, YNQ td, REQ td
contenttype: req_clarif_query
content: ?

Requests by the system to provide more information for the QA request, in order to arrive at a useable QA query.

Example:

Ik begrijp het woord [woord] niet.

Example:

U:kan ik een ziekte krijgen?

S: Welke ziekte bedoel je?

4.3 QA Answer (system)

acttype: INF td
contenttype: answer_qa
content: the lists of answers from the QAs

The system's presentation of a QA query will be called QA Answer. It is more complex and specific than answer. It contains some meta information, such as confidence level, and a formulation of the question submitted to the QA.

4.4 Search Failure Inform (system)

acttype: NAUFEXE?, NAUF td?
contenttype: feedback_searchfail
content: failure information

The system may give feedback on search failures, using confidence information or (possibly) conditional results and presupposition problems.

4.5 Answer Evaluation (user)

acttype: NAUF,PAUF,PALF,NALF
contenttype: feedback_answereval
content: information indicating answer quality

The user gives an opinion about the QA answer. The user is likely to give (implicit) evaluative feedback during the dialogue, which may be used to refine the query.

Answer quality is basically something like: "good answer" (pauf), "bad answer" (nauf), "nonsensical answer" (nauf, nalf), "puzzling answer" (nauf, nalf, may be confirmation). The user will often embed a follow up question in his answer evaluation. This is modelled as two dialogue acts.

Ex. "ja maar, ik bedoelde ..."

Ex. "maar wat heeft dat te maken met ...?"

Ex. "gaat dit wel over ...?"

5 Task negotiation / Dialogue Control (system and user)

Generally, any acts that are meant to update the dialogue control status other than the "normal" dialogue flow through the other acts. This may mean explicitly talking about dialogue acts, or hinting at them.

5.1 Restart request (user)

acttype: REQ,NAUF,NALF
contenttype: req_resetdialogue
. req_exitclarif
content: none

The user may indicate he wants to break off the current subdialogue and return to the previous task, or start all over again.

Examples:

"I don't want to answer this question"

"Let's start over again"

6 Other

6.1 Social / politeness acts (user, system)

acttype: GRT,APO,GRA
contenttype: social
content: none

Dialogue openings and closings, thank yous, etc. While we probably won't use these acts for analysis, we should account for them by responding reactively.

6.2 out-of-domain utterances (user)

acttype: WHQ,YNQ,REQ
 contenttype: outofdomain
 content: none

This may be any kind of utterance that the system cannot handle. For example, out of domain questions, meta questions, and utterances in which the system or user is subject. We should account for the most often occurring ones and tell the user that they cannot be handled.

Examples:

What does this system do?

What does this button mean?

Do you know something about <general-non-domain-subject>?

Where can i find your maker?

7 Dialogue act summary table

All acts:

agt	cf	contenttype	content
s u	whq ynq req	question	VHypRef item (, Vector choices)
s u	whq ynq req nauf	repreq_dialogue	VHypRef act (, Vector choices)
s u	whq ynq req nauf	repreq_speech	VHypRef item (, Vector choices)
s	whq ynq req nauf	repreq_parse	VHypRef item (, Vector choices)
s	whq ynq req nauf	repreq_pen	VHypRef item
s	whq ynq req nauf	repreq_qa	VHypRef question
s u	pauf palf	confirm_ok	VHypRef [] items
s u	inf pauf palf	confirm_implicit	VHypRef [] items
s u	ynq inf pauf palf	confirm_explicit	VHypRef [] items
s u	INF	inform	VHypRef [] items
s u	INF	answer	VHypRef request
u	INF nalf	reformulate	VHypRef act
u	whq ynq req	q_domain_new	(parse is implicit par)
u	whq ynq req	q_domain_followup	(parse is implicit par)
u	whq ynq req	q_domain_clarif	(parse is implicit par)
s	whq ynq req	req_clarif_query	VHypRef act, Object topic
s	inf	answer_qa	VHypRef act (, qa answers implicit)
s	naufexe nauf	feedback_searchfail	VHypRef act, Object info
u	nauf pauf palf nalf	feedback_answereval	VHypRef act, Object info
u	req nauf nalf	req_exitclarif	VHypRef question
u	req nauf nalf	req_resetdialogue	none
s u	grt apo gra	social	none
u	whq ynq req	q_outofdomain	none