AMIDA

Augmented Multi-party Interaction with Distance Access

`http://www.amidaproject.org/`

Integrated Project IST–033812

Funded under the 6th FWP (Sixth Framework Programme)

Action Line: IST-2005-2.5.7 Multimodal interfaces

Deliverable D6.7: AMIDA Proof-of-concept System Architecture

Due date: 31/03/2008 **Submission date:** 31/03/2008
Project start date: 1/10/2006 **Duration:** 36 months
Lead Contractor: IDIAP Research Institute **Revision:** 1

Project co-funded by the European Commission in the 6th Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

D6.7: AMIDA Proof-of-concept System Architecture

Abstract:

Two meeting assistance systems have been built around the Hub client-server architecture for annotation exchange, in order to demonstrate the meeting processing capabilities of the AMIDA project, the integrative power of the Hub, and the potential utility of the envisaged applications. The first system – the AMIDA Automatic Content Linking Device – is a real-time query-free document retrieval system, while the second one – the AMIDA Mobile Meeting Assistant – is a mobile device that allows remote participants to attend a meeting and improves their engagement with respect to voice-only solutions.

The ACLD is a system that constantly retrieves items from a document repository, which includes meeting related documents together with excerpts from previous meetings of the group, and displays them to a participant or to all of them; the device can be used online, during a meeting, or offline, integrated in a meeting browser. Its main components are the Document Bank Creator, the Indexer, the Query Aggregator, the User Interface, the Monitor, and the Meeting Simulator.

The MMA takes advantage of the automatic annotation of the visual focus of attention of meeting participants, and other annotations, to reconstruct a 2D or 3D representation of the meeting on the mobile device (smart phone) of the remote participant. The audio and slides are also conveyed.

Results and feedback for the first versions of the systems are given at the end of each description, together with plans for the future development of the systems within the AMIDA project.

Contents

1 Introduction **4**

1.1 Outline of the demonstrators 4

1.2 Overview of the Hub 5

2 The AMIDA Automatic Content Linking Device **6**

2.1 Objectives 6

2.2 Overall Architecture 7

2.3 Components of the Automatic Content Linking Device 9

2.3.1 User Interface 9

2.3.2 Document Bank Creator 9

2.3.3 Document Indexer 10

2.3.4 Query Aggregator 11

2.3.5 Work in Progress: Meeting Simulator and Meeting Monitor . . . 13

2.4 Implementation and Results 14

2.4.1 Software Prerequisites 15

2.4.2 Integration Effort 15

2.4.3 Controlling the Application 17

2.4.4 Results of the AMIDA ACLD 18

2.5 Evaluation, Feedback and Future Plans 18

2.5.1 Feedback from Potential Users and Customers 20

3 The Mobile Meeting Assistant **23**

3.1 Requirements for a Mobile Meeting Assistant 23

3.1.1 Other Devices and Representations 24

3.2 Design of the MMA 24

3.3 Implementation of the MMA and Results 26

3.4 Evaluating the MMA through a Pilot User Study 27

3.4.1 User Study: Questions and Answers 28

3.4.2 User Study and COI Workshop: Suggestions 29

3.5 Future Work 31

1 Introduction

This deliverable describes the specification and implementation of two integrative applications that serve as proof of concept for AMIDA technologies: the Content Linking Demo (Section 2) and the Mobile Meeting Assistant (Section 3). These two applications are both using the Hub (see Section 1.2 and (AMIDA, 2007)), a client/server architecture for real-time sharing of annotations over multimodal data. The applications constitute the first AMIDA proof-of-concept system architectures, and they define the bases for integrating the existing AMIDA technologies within complex, real-time systems.

The successful implementation of the applications, accompanied by a preliminary assessment of their qualities, demonstrate that meeting assistant technologies based on automatic meeting analysis have reached significant maturity within the AMIDA project. In particular, the Hub is shown to be an adequate architectural infrastructure for technology demonstrations. Feedback from users and observations from developers indicate how these demonstrations can be improved in the next phase of the AMIDA project.

In the remainder of this section, we describe in more detail the objectives of the two applications – which were successfully demonstrated at the February 7-8, 2008 annual review and at the AMI Community of Interest Workshop, February 4-5, 2008 – and provide an outline of the Hub technology which supports both demonstrators[1].

1.1 Outline of the demonstrators

The *Automatic Content Linking Device (CLD)* is designed to help meeting participants who discuss specific facts or topics mentioned in documents, but who do not have the time to perform actual searches to retrieve the documents (and respective facts) in real-time during the meeting. The CLD searches at regular intervals a database of documents that is prepared before a meeting – including reports, emails, presentations, fragments and minutes of past meetings – based on a criterion constructed from the words and terms that were recognized automatically in the meeting discussion.

Results are presented as a list of document names ordered by relevance (possibly an empty list) using a persistence mechanism to limit the variations of speech-based retrieval. A user interface offers the participants quick access to the content of the documents, whenever they need it. Hovering over a document gives some more information (for instance, for a meeting fragment, it displays the its extractive summary), and clicking on it enables the document to be viewed in full (including playback for topic segments).

The *Mobile Meeting Assistant* is intended for a remote participant to a meeting, connected through a mobile phone to a smart meeting room. The assistant is aimed at increasing the engagement in the meeting of the remote participant, despite the fact that the video signal from the meeting is not available through the mobile phone (a common bandwidth limitation with mobile devices).

Therefore, the assistant takes advantage of the automatic recognition of the speaker's identity, of the visual focus of attention of the participants, and of the slide capture, to

[1]This deliverable, numbered D6.7 in the updated AMIDA Technical Annex (February 2008), was initially numbered D7.3, and pertained to WP7. The title and topics remain those defined in the initial TA.

display 2D and 3D simulations of the meeting scene on the mobile device, thus allowing the remote participant to better follow the flow of discussion in the smart meeting room, and to get a detailed, real-time view of the projected slides regardless of their original file format.

1.2 Overview of the Hub

The Hub is a client-server architecture for annotation exchange between 'consumers' and 'producers', which is extensively documented in AMIDA Deliverable 7.2 (AMIDA, 2007). The Hub consists of a server to which both consumers and producers connect, and a database to which all annotations produced by the producers are stored. The database can be populated with existing annotations by using a dedicated producer that reads tab-separated files. There is also a producer that reads the AMI Meeting Corpus (Carletta et al., 2006) in its native NXT format and populates the Hub directly from that.

The Hub integrates real-time data exchange with historical data retrieval. Producers can come online at any time, declare the data they will produce and start producing it. Because the underlying format in the Hub's database is simple (using timed triples), no database changes are required for new types of data. Consumers request data using queries which can constrain the meeting, speaker, producer, annotation type and value of the results. Each matching annotation is then passed to the consumer, historical matches first, then live data.

To make a realistic demonstration framework, we load up the Hub with the first three meetings of a four-meeting set from the AMI Meeting Corpus. To do this we use the converter mentioned above. The data for the fourth meeting in the set is passed to the Hub as if it were live, using the same converter but with an extra flag telling it to adjust the time at which each element is produced. Ideally we would pass no pre-computed information to the Hub for this fourth meeting, instead of computing everything on the fly. At present however, the results of automatic speech recognition are computed offline and passed "as live". For the Mobile Meeting Assistant demo, the visual focus of attention is also computed offline.

Within this framework, the modules in the two demonstrators can simply be seen as Hub producers and consumers (and sometimes both), with the top-level consumer being the user-interface. The Hub thus proves to fulfill its objectives as an architecture for real-time annotation exchange.

2 The AMIDA Automatic Content Linking Device

2.1 Objectives

Participants in a meeting often mention documents containing facts that are currently discussed, but only a few of them are at hand. Searches could be performed in an document management system for the right piece of information, but the participants in a meeting usually do not have the time to perform such operations frequently during a meeting. Even when browsing through the recording of a previous meeting, users do not have the time to search for additional information related to that meeting, though this would help them better understand or locate information in the meeting.

Therefore, a system that could relate ongoing discussions to potentially relevant documents or other pieces of information, i.e. an Automated Content Linking Device (CLD), would be very valuable in at least two application scenarios:

1. *'Just-in-time' retrieval:* participants to a meeting are constantly made suggestions about documents (including excerpts of previous meetings) that are potentially relevant to the ongoing discussion. Participants are free to ignore them, or to start using them to enhance the discussion, e.g. with figures, precise facts, or decisions that were made in previous meetings. Previous work in this direction are papers by Hart and Graham (1997); Budzik and Hammond (2000); Henziker et al. (2005).

2. *Document/speech alignment for meeting browsers:* users of a meeting archive can view the recordings of previous meetings augmented with related documents, regardless of whether the participants to the meeting referred to them explicitly or not. This is crucial for meetings whose main purpose is to discuss a long document, e.g. a report. Previous work in this direction are papers by Rhodes and Maes (2000); Franz and Milch (2002).

Conceptually, the content linking mechanism is the same in both cases – only the resources that are available and the constraints of real-time results are different. Therefore, in the current implementation of the AMIDA CLD, a meeting from the AMI/IM2 Meeting Corpus is replayed (here, ES2008d as a case study) and as it plays, the device indicates relevant documents and pseudo-documents (segments from previous meetings, i.e. ES2008a-c). The highlighting changes as the meeting proceeds to show appropriate identifiers for those past documents that are most relevant at each point. Hovering over a document identifier gives more information (for instance, for a meeting segment, the label or the extractive summary), and clicking on it enables the document to be viewed in full (including playback for topic segments).

The Automatic CLD demonstrates the AMIDA concept of a (potentially remote) meeting assistant, which is supported by user requirements, and whose interface is the result of principled design. The device draws together a number of key technologies developed in AMIDA, and proves that the Hub is an effective communication and integration device in realistic conditions.

2.2 Overall Architecture

The Automatic Content Linking device is designed to perform searches at regular intervals, in a database of documents that can be prepared before a meeting, including for instance reports, emails that were exchanged, minutes and presentations from past meetings, etc. Pseudo-documents such as fragments of previous meetings that are related to the current one are also included in the repository.

The search criterion is constructed based on the terms that were recognized automatically from the meeting discussion (some pre-specified terms can receive greater weight) and the results are presented as a list of document names ordered by relevance (which can be empty if no document matches enough the words that were recognized). A persistence mechanism helps documents that are often retrieved to remain some time at the top of the list. The user interface offers the participants quick access to the content of the documents that are retrieved, if they believe that they contain valuable information to a given topic in the meeting.

These functionalities are supported by a number of modules, the main ones being the Document Bank Creator, the Document Indexer, the Query Aggregator, and the User Interface. These components exchange data through the subscription-based client/server architecture of the Hub. The system's architecture is shown in Figure 1.

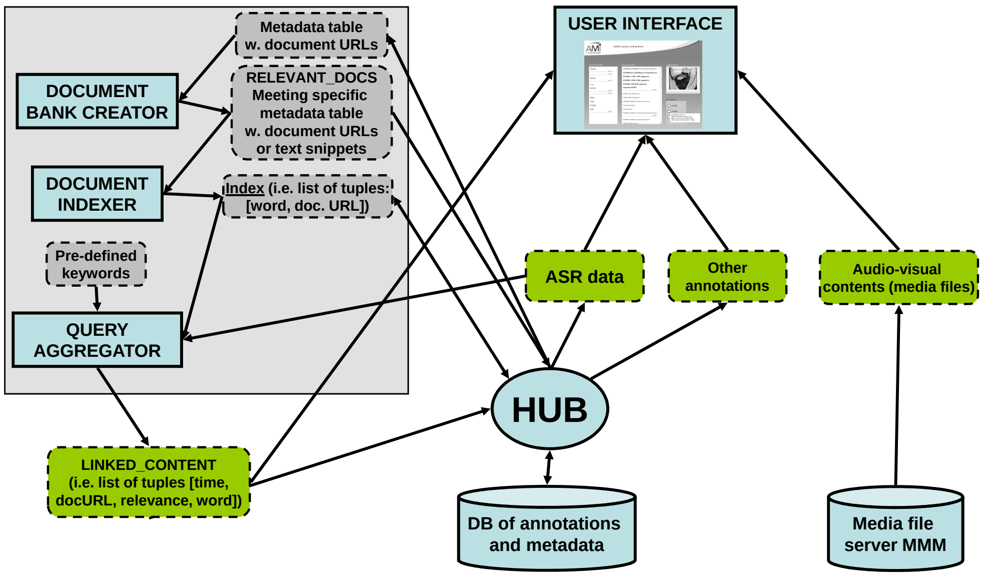

Metadata table w. document URLs
DOCUMENT BANK CREATOR
RELEVANT_DOCS Meeting specific metadata table w. document URLs or text snippets
DOCUMENT INDEXER
Index (i.e. list of tuples: [word, doc. URL])
Pre-defined keywords
QUERY AGGREGATOR
LINKED_CONTENT (i.e. list of tuples [time, docURL, relevance, word])
USER INTERFACE
ASR data
Other annotations
Audio-visual contents (media files)
HUB
DB of annotations and metadata
Media file server MMM

Figure 1: Architecture of the AMIDA Automatic Content Linking Demo.

The main components will be described in separate subsections below, but the following outline offers an overall perspective of their input and output data in order to facilitate the understanding of the data exchanges between modules.

- *Document Bank Creator* – documents are all reports, emails, minutes and presentations (slides) from past meetings, which are potentially relevant to the current meeting; pseudo-documents are fragments of previous meetings from a series.
  - IN: name of meeting (from user)
  - IN: metadata table with document URLs (from Hub's database)
  - OUT: metadata table with URLs or short text snippets of documents relevant to given meeting (to Hub's database)
- *Document Indexer* – uses Apache Lucene[2]
  - IN: metadata table produced by Document Bank Creator (directly via Hub or, first, given explicitly by an expert user)
  - IN (optional): list of particularly relevant keywords (from user)
  - OUT: index, i.e. machine-readable list of tuples (meeting, keyword, doc_type, pointer) (to Hub's database or, first, to a local file)
- *Query Aggregator* – performs searches using words and terms that are recognized automatically from the meeting discussion; a list of pre-specified terms which receive more weight when searching can be used; the Query Aggregator produces a list of document names, ordered by relevance, based on the search results and on a persistence model explained below.
  - IN: ASR from the Hub (inserted by Meeting Simulator or directly by an ASR producer such as Tractor)
  - IN (optional): other annotations (same source)
  - OUT (periodically): list of tuples (meeting, time, keyword, relevance, doc_type, pointer) (to Hub's database), i.e. Linked_Content annotation
- *User Interface* – displays results from Query Aggregator and offers quick access to TXT/HTML and source versions of documents.
  - IN: media files (either via URLs obtained from metadata tables in the Hub's database, or from "simulated" streaming managed by the Meeting Simulator)
  - IN: annotation files, especially Linked_Content (from the Hub's database or via the Hub directly from the Query Aggregator and other producers)
  - OUT: display meeting media, relevant documents and other annotations
  - OUT: interaction with documents

Several modules appear in Figure 1, but were not mentioned in the list above because they do not play a role in the version of the demo at the time of writing. These are the Overall Controller, the Meeting Simulator, and the Hub Monitor, whose intended role is described in more detail in Section 2.3.5 below.

[2]See `http://lucene.apache.org/` or for the Perl implementation used here, `http://search.cpan.org/dist/Plucene/`.

2.3 Components of the Automatic Content Linking Device

2.3.1 User Interface

We start the description with the *User Interface*, as this allows a better understanding of the functioning of the system. In theory, the only information given to the User Interface is the identifier of a meeting to display. This allows the interface to retrieve via the Hub (and its metadata database) the pointers to the related media, and the annotations that will be displayed, including the *Linked_Content* annotation produced by the Query Aggregator. The meeting could in principle be an ongoing or a completed one, though for demonstration purposes it is more convenient (for repeatability reasons) to use a completed one (here, ES2008d). The Interface should also allow the user to start playing the meeting. In practice, in the current implementation, a number of variables (mainly related to metadata) are hard-coded into the interface for meeting ES2008d.

Using the Linked_Content data structures – which basically contain meeting, time, keyword, relevance, document type, and URL to document – the User Interface displays for a given moment or place-in-transcript the *N* most relevant documents. It is possible to display only documents above a certain relevance threshold, but this filtering is more efficiently done by the Query Aggregator itself. The URLs allow the interface to show the list of the relevant documents and pseudo-documents, changing the highlighting of the most relevant ones as the meeting proceeds.

The interface also offers the users several possibilities for interaction with documents. These hover-over and clicking behaviours are different for each document type. For a meeting fragment, hovering over its label displaying its extractive summary obtained on-the-fly from the Hub, while clicking on the label displays the ASR transcript. For documents, clicking on their label displays their content in a raw-text window, from which a formatted version in HTML can be obtained. The source document can also be retrieved, but opening it with its dedicated program (often MS Word or Powerpoint) slows considerably the viewing process. These behaviors can easily be redefined by the User Interface (the only important information being the metadata that indicates the type of the document), and in the future, user studies should indicate the optimal behavior for rapid document exploration.

The User Interface is implemented using two components: the interface itself is in Flash, and a Java front-end ensures communication with the Hub as a consumer. A snapshot of the implemented interface is provided in Section 2.4 below, in Figure 3.

2.3.2 Document Bank Creator

The *Document Bank Creator* is run offline before a meeting to create the bank of documents and pseudo-documents that will be searched over during the meting. This is a preparation task, which either generates a metadata layer that indicates where documents are, or simply copies the documents to a separate folder, in preparation for the Document Indexer, which must be explicitly told which documents are potentially relevant to the current meeting. In a more user-friendly scenario, the module is told at the beginning of a meeting which project or series the meeting belongs to, and determines potentially relevant documents, which are sent to the indexer. The documents are referenced in the Hub's database (in the future based on metadata structures developed in the IM2.DMA project[3], and the result of the Document Bank Creator module (list of URLs to documents for a given meeting) can be viewed as a new metadata layer which can be added to the Hub's database.

The documents that are considered by the Document Bank Creator can be support documents, short fragments of previous meetings (whole meetings do not seem helpful as search results), slides, emails, personal web pages, etc. In particular, this requires breaking up the previous meetings into pseudo-documents corresponding to short segments; at their largest, these would be topic segments, but in the current implementation they are 30-second snippets of the meetings, which is a more appropriate length for retrieval (and ASR consultation or audio playback). These pseudo-documents are created from the ASR of the previous meetings.

The Document Bank Creator accepts heterogeneous file formats, and extracts text from them using calls to the proprietary software that created the files. The most frequent calls are to MS Word and MS Powerpoint, but MS Excel and HTML documents must also be converted to text in order to prepare them for indexing (in fact calls to MS Word is also used to extract text from HTML documents). Therefore, once the Document Bank Creator gathers the documents that are potentially related to a meeting, it starts a series of external calls to the above-mentioned software to derive text versions which will be used by the Document Indexer. In the process, the module also generates HTML versions of each document, which are easier and quicker to visualize by the user than the original MS Office versions.

2.3.3 Document Indexer

The *Document Indexer* uses the text version of the files associated to the current meeting by the Document Bank Creator to construct an index, i.e. a data structure that optimizes word-based search over the document set, which can become quite large over time, depending how many documents are attached to a meetings. The index can also be viewed as a new annotation layer, represented logically as a list of tuples: (meeting, keyword, doc_type, URL), where the URLs of the documents are used as unique identifiers.

One could index the documents for a number of prespecified keywords only, or increase the weight of such prespecified keywords[4]. However, current document indexers do a fine job by using all words as keywords, for instance Apache Lucene, which is used here in its Perl implementation called *Plucene*. Plucene constructs enhanced and optimized indices (using word stemmers and the TF-IDF weighting scheme (Salton and McGill, 1983)), and in the long-term, semantic analysis could be added to keywords in context to perform concept-based indexing.

[3]This is the Data Management and Access module of the IM2 Swiss NCCR, see `http://www.im2.ch`; the paper by Popescu-Belis and Estrella (2007) explains the metadata solution developed in IM2.DMA.

[4]Note also that each segment could be indexed based not just on the words spoken but also based on the words present on any slide put up soon before or during the segment, but this is the responsibility of the Indexer; or slides could be indexed independently, and it would be the user interface that, when a slide is retrieved, would play the corresponding part of the meeting.

As the index is a permanent layer of information concerning the documents related to a meeting, it could be stored in a declarative format in the Hub's main database, from where it could be retrieved at the beginning of the demo by the Query Aggregator, which is constantly using it. Alternatively, in the present implementation, the index is accessed directly by the Aggregator as a set of files.

2.3.4 Query Aggregator

The *Query Aggregator* periodically extracts from the speech of a given meeting a list of keywords that are mentioned (using the ASR or a manual transcript for development purposes). This aggregator gets the words from the Hub[5] and processes them in batches of fixed size. This can actually mean either at regular time intervals, or for a fixed number of words/utterances, e.g. every 30 seconds or 5 utterances or 50 words (the implemented version is the first one)[6].

So, the Query Aggregator is both a Hub data consumer and producer: it tracks the ASR as it comes through and periodically extracts keywords from the ASR; then, it combines them to build a query, and searches the index to retrieve relevant documents, which are sent as new annotations to the Hub – from where they can be used by the User Interface to display the document labels and give access to them.

It is possible to spot a number of pre-specified keywords as they come through the ASR, and to increase the importance of these keywords when doing the search, using Lucene's boosting mechanism (the weight of the boosting can be set too: here between 3 and 5 times the weight of non-boosted words). However, the aggregator works also without a list of boosted terms. A specific list was defined by the user-study group for the meeting series under study, and at present it contains the following words or expressions:

Keywords: 'fashionable', 'innovative', 'fancy', 'injury', 'cost', 'energy', 'component', 'case', 'chip', 'interface', 'button', 'L_C_D', 'lcd', 'material', 'latex', 'wood', 'titanium', 'easy', 'technical', 'working', 'look', 'meeting','design', 'project', 'remote', 'control', 'user', 'minutes', 'detailed', 'concept'.

In addition, the words from the ASR or transcript are filtered for stopwords, so that ideally only content words are used for search. The following list of stopwords is used:

Stopwords: 'uh-huh', 'can', 'just', 'would', 'it's', 'we're', 'I'm', 'we'll', 'Let's', 'I', 'the', 'it', 'we', 'that', 'like', 'of', 'to', 'yeah', 'a', 'in', 'and', 'you', 'your', 'for', 'is', 'on', 'by', 'be', 'with', 'do', 'will', 'how', 'are', 'what', 'from', 'or', 'as', 'this', 'have', 'um', 'so', 'okay', 'mm-hmm', 'gonna', 'at', 'there's', 'they're', 'but', '?', 'I'll', 'that's', 'mm', 'yep', 'ah', 'lemme', 'you're', 'you'll', 'didn't', 'don't', ''cause', 's', 've', 't', 'm', 'no', 'oh', 're',

[5]Words arrive gradually if the meeting is ongoing or simulated, or as a block if the meeting is a finished one, but from the point of view of the aggregator these two modes are similar.

[6]Strictly speaking, a function that asks the aggregator to find relevant documents on demand from a user interface could be useful, to allow users to update the list of relevant documents when they need it; implementing such a function would only change (temporarily) the period of the aggregator.

'if', 'wow', 'cause', 'hmm', '[gap]', 'kay', 'my', 'am', 'yes', 'wasn', 'kinda', 'll', 'alright', 'they', 'an', 'uh', 'was'.

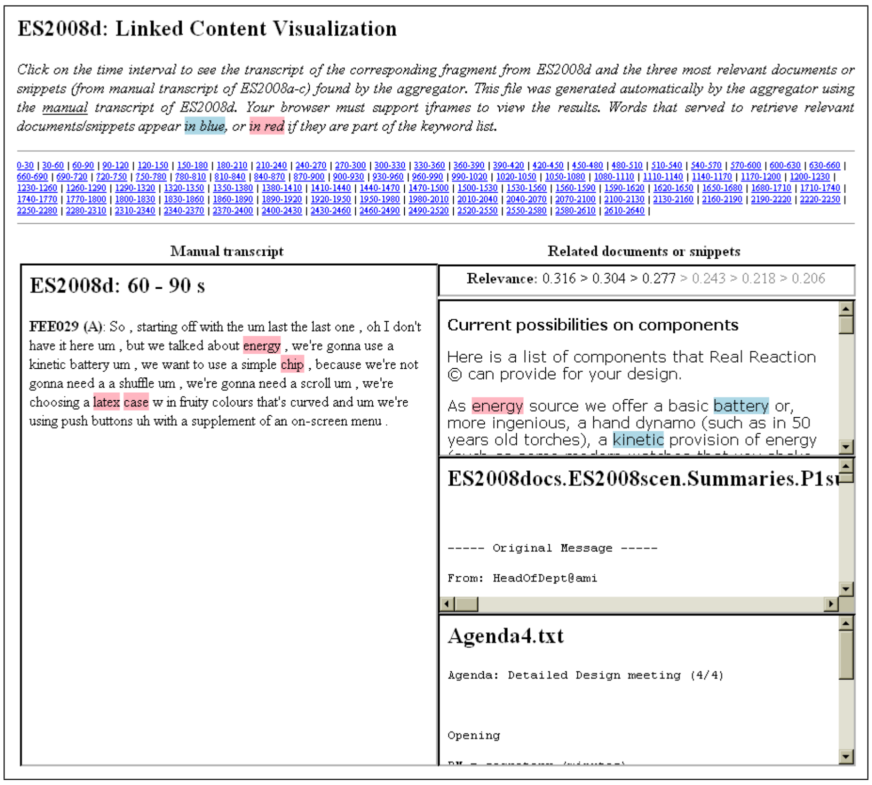
The Aggregator performs document search by matching the keywords from the ASR (from which stopwords were removed) with those from the index constructed above and returns the most relevant set of documents at that given moment, more specifically a list of tuples such as (meeting, time, keyword, relevance, doc_type, pointer). It is useful to include in this annotation the keywords that were matched (i.e. the ones that helped to retrieve the specific document) as well as a relevance score (preferably an absolute one, i.e. from 0 to 100%) to allow the interface to sort the relevant documents as needed.

This list is itself a new annotation layer, called Linked_Content, which can be sent via the Hub and stored in the Hub database, to which the User Interface must subscribe. The output can be described conceptually, for each document, using the following fields (they could be stored individually in the Hub as timed triples, but they are currently sent as XML into the Hub):

Document name: name of the document returned by the query.
Document id: some form of identifier of the document.
Document type: possible types of documents are 'seg' (a 15-20" segment or snippet of meeting), 'doc' (MSWord), 'ppt' (MS Powerpoint), 'xls' (MS Excel), 'txt' (in the future, more fine-grained types such as agenda, questionnaire, or minutes could be used.
URL: location of document (for meeting snippets, location of ASR transcript).
Query words: words of the query, separated by whitespaces, without the stopwords, but where keywords are boosted by marking them with ^*N* where *N* is a digit (3–6). This is useful to highlight the words in the documents.
Media URL: location of media file for meeting snippets.
Relevance: relevance score of document, based on the query results from Lucene search engine, filtered and modified as explained below.
Rank within result set: rank of document out of those returned by the query.
Related meeting: identifier of meeting from which relevant document retrieved.
Start time: start time of the relevant meeting segment (if type is snippet).
End time: end time of the relevant segment.
Meeting id: ID of the meeting from which the relevant document was retrieved.

To avoid inconsequent results from one time frame to another (due to the fact that words vary considerably from one frame to another) a persistence or "smoothing" mechanism was also defined. This mechanism modifies current relevance scores for each document returned by the Lucene search based on those from the previous time frame. The smoothing factor, called α, has the following impact on the updated relevance (noted $r'(t_n)$ where t_n is the current time frame and t_{n-1} the previous time frame): $r'(t_n) = r(t_n) + (\alpha * r'(t_{n-1}))$. A large value of α denotes a larger persistence – if $\alpha > 1$ there is an ever increasing $r(t_n)$) even if the document is no longer retrieved by Lucene – and in our experiments a typical value of $\alpha = 0.8$ was used[7].

[7]We have also experimented with a better controlled formula: $r'(t_n) = (r(t_n) + \alpha * r'(t_{n-1}))/(1 + \alpha)$ where $\alpha = 0$ means no influence, $\alpha = 1$ means arithmetic average of the two, $\alpha < 1$ that previous value has less influence than current, $\alpha > 1$ means more ..., $\alpha = \infty$ means the list remains unchanged. The problem with this formula is that the case when $r(t_{n-1})$ is close to zero is very different from the case where $r(t_{n-1}) = 0$.



ES2008d: Linked Content Visualization

Click on the time interval to see the transcript of the corresponding fragment from ES2008d and the three most relevant documents or snippets (from manual transcript of ES2008a-c) found by the aggregator. This file was generated automatically by the aggregator using the manual transcript of ES2008d. Your browser must support iframes to view the results. Words that served to retrieve relevant documents/snippets appear in blue, or in red if they are part of the keyword list.

0-30 | 30-60 | 60-90 | 90-120 | 120-150 | 150-180 | 180-210 | 210-240 | 240-270 | 270-300 | 300-330 | 330-360 | 360-390 | 390-420 | 420-450 | 450-480 | 480-510 | 510-540 | 540-570 | 570-600 | 600-630 | 630-660 | 660-690 | 690-720 | 720-750 | 750-780 | 780-810 | 810-840 | 840-870 | 870-900 | 900-930 | 930-960 | 960-990 | 990-1020 | 1020-1050 | 1050-1080 | 1080-1110 | 1110-1140 | 1140-1170 | 1170-1200 | 1200-1230 | 1230-1260 | 1260-1290 | 1290-1320 | 1320-1350 | 1350-1380 | 1380-1410 | 1410-1440 | 1440-1470 | 1470-1500 | 1500-1530 | 1530-1560 | 1560-1590 | 1590-1620 | 1620-1650 | 1650-1680 | 1680-1710 | 1710-1740 | 1740-1770 | 1770-1800 | 1800-1830 | 1830-1860 | 1860-1890 | 1890-1920 | 1920-1950 | 1950-1980 | 1980-2010 | 2010-2040 | 2040-2070 | 2070-2100 | 2100-2130 | 2130-2160 | 2160-2190 | 2190-2220 | 2220-2250 | 2250-2280 | 2280-2310 | 2310-2340 | 2340-2370 | 2370-2400 | 2400-2430 | 2430-2460 | 2460-2490 | 2490-2520 | 2520-2550 | 2550-2580 | 2580-2610 | 2610-2640 |

Manual transcript

ES2008d: 60 - 90 s

FEE029 (A): So , starting off with the um last the last one , oh I don't have it here um , but we talked about energy , we're gonna use a kinetic battery um , we want to use a simple chip , because we're not gonna need a a shuffle um , we're gonna need a scroll um , we're choosing a latex case w in fruity colours that's curved and um we're using push buttons uh with a supplement of an on-screen menu .

Related documents or snippets

Relevance: 0.316 > 0.304 > 0.277 > 0.243 > 0.218 > 0.206

Current possibilities on components

Here is a list of components that Real Reaction © can provide for your design.

As energy source we offer a basic battery or, more ingenious, a hand dynamo (such as in 50 years old torches), a kinetic provision of energy

ES2008docs.ES2008scen.Summaries.P1s

```
----- Original Message -----

From: HeadOfDept@ami
```

Agenda4.txt

```
Agenda: Detailed Design meeting (4/4)


Opening
```

Figure 2: HTML view of the offline output of the Query Aggregator.

A filtering mechanism deletes the least relevant of the documents sent to the User Interface. In any case, at most 6 documents are returned, but fewer documents may be returned if the relevance of the latter is small. This is set by two variables (in all of the above, 'relevance' is the updated relevance computed using the formula above):

min-absolute-relevance (typically 0.2): all documents with relevance below this value are deleted;

max-relative-decrease (typically 0.5): end list of results if document d_{k+1} score is lower than this proportion of the d_k score.

There is also an HTML version of the aggregator which generates static XML and HTML views for debugging the QA itself, or simply examining/evaluating the results. The HTML view, shown in Figure 2 displays on the left the list of words from the current time frame, and on the right up to six most relevant documents (in HTML version) with their computed weights. The keywords are highlighted in both meeting transcript and documents, and the words from the transcript are highlighted (in a different colour) in the documents when they are found.

2.3.5 Work in Progress: Meeting Simulator and Meeting Monitor

Two modules which were part of the original design of the ACLD have only been sketched in the first version, and need improvement in the future. The Meeting Simulator, which renders a past meeting as if it were ongoing, is at present implemented as a streamer of annotations from a tabular file into the Hub, but should in the future be able to transform annotations that are already in the Hub. An initial version of the Meeting Monitor, which shows the triples that are written to or read from the Hub, already exist but should be made more flexible in the future.

The *Meeting Simulator* simulates an ongoing meeting from data that corresponds to a completed meeting (here, ES2008d), in case the ACLD is used to view an archived meeting – as well as for demonstration purposes, as it will be unpractical to require people to hold a real meeting for each demo of the system! ASR and other annotations for the meeting must be streamed in real-time through the Hub, which implicitly controls timing of the system. Thanks to the Simulator, the User Interface and the Query Aggregator work in the same way whether the meeting is real-time or past: they receive the same date from the Hub, except that this can be produced by the Meeting Simulator or by real-time processors. So, the indexing/search components can be developed based on annotations/metadata of completed meetings: the content linking functionality is virtually the same whether the meeting is completed or not.

The Meeting Simulator should read annotations from some the Hub's database (or alternatively NXT files or other XML files), rendering them as what would come out of a set of live data producers, and pump them through the Hub. Examples of such annotations are: ASR (version to be specified – which can be reasonably expected from an AMIDA real-time system soon), as well as, and if required by the query aggregator, topic segments along with dialogue act segments and labels. The Simulator could play the meeting media files as well, e.g. by starting streaming processes towards the specific User Interface, because media files do not go through the Hub, or even give a synchronized presentation of audio and video in a dedicated window (e.g. using SMIL).

The *Hub Monitor* is a module consisting mainly of a graphical display that monitors Hub activity, i.e. the underlying transfer of data while the ACLD is running, so that the developers can understand what is going on. As new data producers and consumers are registered, they will be shown by the Monitor, and the messages that they produce will be seen going through the system (e.g. something in the style of the OAA monitor). The Monitor is in fact a part of the Hub which can be developed quite separately from the other modules described above. However, the User Interface and the Hub Monitor could be wrapped into a second, larger interface that displays the whole Automatic Content Linking Device, i.e. the meeting being simulated and the annotations that are going through the Hub.

2.4 Implementation and Results

The current version of the AMIDA Automatic Content Linking Device was successfully demonstrated at the February 7-8, 2008 AMIDA annual review and at the AMI Community of Interest Workshop, February 4-5, 2008. The current version follows the architecture shown in Figure 1 above, on page 7. A number of steps were taken for implementation of the architecture, which are described in this section: listing software prerequisites, definition of integration techniques, and design of application controls. The results observed when running the demo are described at the end of this section (Subsection 2.4.4).

2.4.1 Software Prerequisites

The following software components are prerequisites to the Content Linking Device[8], the main one being the Hub itself. The Hub requires a MySQL database management system (freeware), with a database called 'test', a schema loaded from the `test.db` file, and a user called 'annot'. Java JDK/SDK 1.5 or later should be available to allow compilation (making sure `java` and `javac` are on a visible path – check the PATH environment variable). To run the Query Aggregator, Perl and the Plucene module are required – preferably using Active State Perl for Windows and its Perl Package Manager. Another solution is to install Cygwin with its own Perl, and use Cpan to install Plucene, as this module has a lot of dependencies.

Compilation of all source files is centrally managed by a `build.xml` file in the top level directory, which requires the `Ant` software (again, included in the PATH). A number of variables can be set by modifying the initial lines of `build.xml`, which are self-explanatory:

```
<property name="host" value="localhost" />
<property name="dbName" value="test" />
<property name="user" value="annot" />
<property name="pass" value="*********" />
<property name="table" value="test" />
<property name="debug" value="false" />
<property name="port" value="4000" />
<property name="mysqlDefaultPort" value="3306" />
<property name="timeorderedoutput" value="true" />
<property name="logdir" value="./logs" />
<property name="linkedcontsource" value="online" />
```

The Query Aggregator's Java front-end to the Hub is automatically compiled when using the `Ant` tool on top level. The jar file containing the Hub classes is also automatically build by the 'ant' tool, as is the User Interface's Java front-end.

2.4.2 Integration Effort

The following steps were recommended by the AMIDA ACLD integrators to ensure a uniform integration procedure for all contributors:

1. Download and install your local copy of Hub & Middleware.
2. Create your own installations of the complete system and update them regularly from the CVS. It is possible to work with two instances, one for individual development and implementation purposes (used for implementing your module within the system environment, so this version might differ across partners, e.g. as it contains software not belonging to the system or not (yet) delivered, or works with older versions of system modules), and the other one exclusively for reference installation of

[8]Instructions to install them are available on the AMIDA project Wiki.

the overall system (which should be identical at all partners as it is used for system testing at under identical conditions).

3. Implement a basic version of your module that communicates effectively with the Hub & Middleware (is capable to produce and/or to consume simple example data entered in the Hub using the AmiDataStreamer for instance).
4. Deliver a basic module to CVS, including realistic example output data produced by it, to be used by other modules for manual testing of their I/O-behaviour (see for example the AMIDA wiki page with system traces).
5. *Repeat* the procedure – update your local system, improve/fix your module, deliver new version to CVS – *until system works as intended.*

The following rules were established for module developers concerning directory structuring, to improve integration. The overall system including all modules should appear into the toplevel system directory (onto the CVS), containing the following subdirectories:

- `bin/*`: general executables or startup scripts concerning the overall system
- `doc/*`: general system documentation
- `etc/*`: files necessary for running the overall system, e.g. configuration files
- `lib/*`: libraries of general (not module specific) nature
- `mod/<modulename>/*`: individual system modules (directory structure below)
- `opt/*`: optional components not needed at runtime, e.g. tools, visualizations, etc.

The directory structure being used internally for each module in `mod/<modulename>` is generally free. The recommendation for some subdirectories, whose use could simplify installation and integration for all partners, is the following:

- `mod/<modulename>/bin/*`: module specific startup scripts and executables
- `mod/<modulename>/data/*`: databases or data that are needed at runtime
- `mod/<modulename>/doc/*`: module specific documentation
- `mod/<modulename>/lib/*`: module specific libraries (if necessary)
- `mod/<modulename>/src/*`: module specific sources
- `mod/<modulename>/test/*`: example testdata for manual checking of correct processing

All system directories (including subdirectories) should be considered as *read-only* at runtime – otherwise each site could move quickly towards working with different systems, which in turn makes it difficult to debug the overall system, as it is impossible to reproduce identical system states at different locations anymore. This restriction only poses problems for writing log data (and possibly for configuration files) and it has been relaxed for these files. Moreover, ensuring that all folders are read-only at runtime makes it possible (at least in theory) to start the system from a CD/DVD.

Given that no module is allowed to write data into the system directory tree, one should create for this purpose a module-specific directory (e.g., `/tmp/<modulename>`) outside the system tree. In any case, *responsibility for the data format as well as for the actually produced data generally rests with the data producer.*

The integration of a module consists of the delivery (using `cvs commit`) of a module into the central CVS repository at UEDIN (disk/cvs/ami/Demos/ContentLinkingDemo_V0.1), under `mod/<modulename>` structured as described above. For each module, there should be a single command to start the module, usually an executable shell script saved in `mod/<modulename>/bin`. Before delivery, each module must have been successfully tested in interaction with the Hub Middleware as well as in conjunction with the latest committed updates from the CVS (therefore, before the delivery, one should perform first a `'cvs update -d'` command). If the answer to all the following questions is "yes", then the module may be committed to the CVS:

- Can the module be integrated with the Hub middleware successfully?
- Is the module version information and documentation up-to-date and complete?
- Does the module correctly process the available test data as input?

2.4.3 Controlling the Application

The demo runs on a Windows PC (other platforms will be considered later).

The following operations need to be performed, and initially they were done using `.bat` scripts and/or typed commands:

1. start MySQL and delete all content in the 'test' table within the 'test' database;
2. start the Hub server (use `mod/HubMiddleware/bin/startHubService.bat`);
3. populate the Hub with metadata and data for meetings *ES2008a*, *b* and *c* (use `mod/HubMiddleware/bin/populateHubForCLD.bat`);
4. start the Query Aggregator (use `mod/QueryAggregator/bin/startQueryAggregator.bat`);
5. start the Hub Monitor (use `mod/HubMonitor/monitor.bat`);
6. start the User Interface (use `mod/UserInterface/bin/startUIConsumer.bat` and then `amidaUI.exe`);
7. start streaming the words for meeting *ES2008d* (use `mod/HubMiddleware/bin/runWordStreamer.bat`).

However, a more convenient method for compiling and running the demo was defined, using the 'ant' command and the 'build.xml' file, which resides in the top-level directory on CVS. The following commands have to be run from this directory:

1. `ant runHubWithCleaning` = drops the content of the 'test' table, starts the Hub and populates the table again.
2. `ant runQueryAggregator` = starts the Query Aggregator.
3. (optional) `ant runHubMonitor` = starts the Hub Monitor.
4. `ant runUI` = starts the UIConsumer (frontend to the Hub) and the Flash GUI.
5. `ant runWordStreamer` = streams the words (ASR) for meeting ES2008d through the Hub.

There are also individual ant targets for dropping the table, running the Hub, compiling, and so on, as shown in the build.xml file. Calling ant runHubWithCleaning after a clean checkout from CVS will also compile all necessary modules.

An SQL solution to reset the Hub's table into the state that holds "after" meetings ES2008a, b and c, but "before" meeting ES2008d, has been developed, and the number of records a database restricted to this series should contain was determined independently by several developers (39322). The reset.sql script first removes all words inserted by the ASR module (in fact, at present, its simulated stream) regardless of their timing, and then all other annotations inserted after the last insertion pertaining to meeting ES2008c, which is the Extractive_Content. The embedded SELECT requires MySQL version 5.0 or higher.

```
-- Usage: mysql --user=annot -p test < reset.sql
--
DELETE FROM test WHERE object LIKE "ES2008d.word.%" ;
--
DELETE FROM test WHERE time >
  (SELECT time FROM
    (SELECT * FROM test) AS _test
  WHERE ((object = "consumer.Extractive_Consumer")
    AND (attribute = "unregisters"))
  ORDER BY time DESC
  LIMIT 1);
```

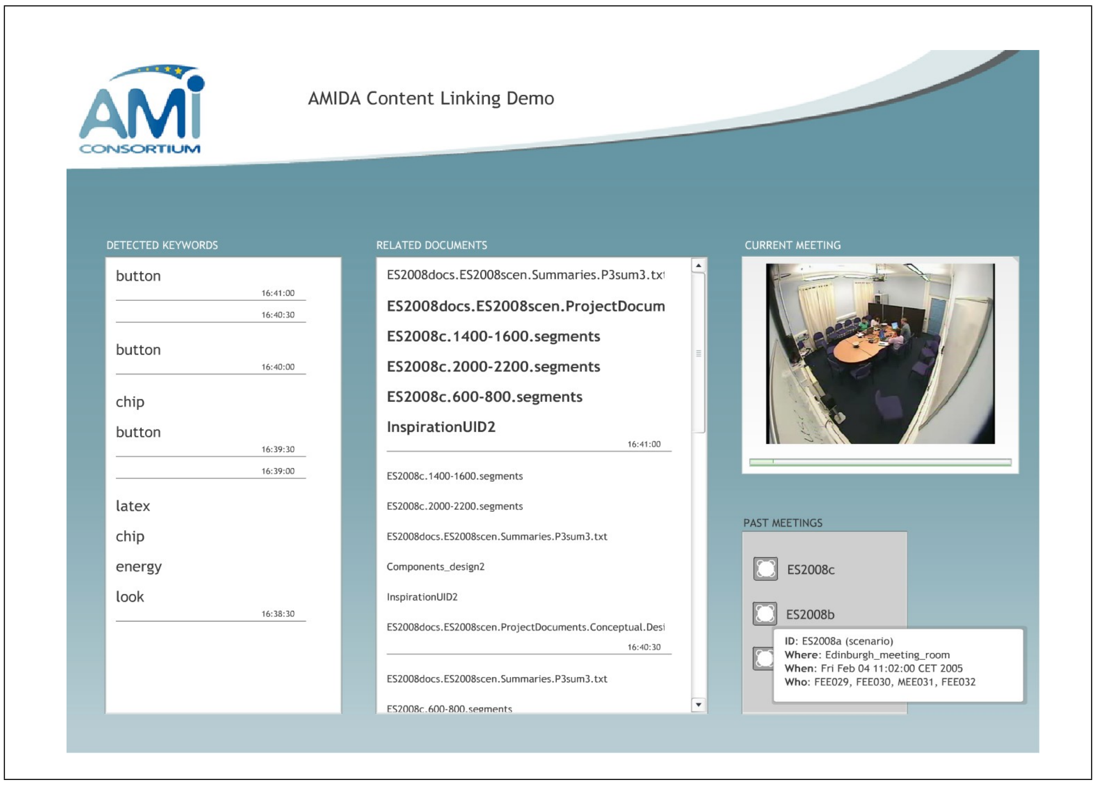
2.4.4 Results of the AMIDA ACLD

A snapshot of the resulting application is shown in Figure 3. The execution tests of the ACLD have been satisfactory: the communication between the modules using the Hub works smoothly, and the logs show proper connection, sending and receival of annotation triples. The documents that are retrieved contain the expected words and keywords (this is easier to check on the static HTML representation produced by the Query Aggregator and shown in Figure 2 above) and all the functionalities offered by the User Interface over these documents are available. The nature of Perl scripting makes it easy to change many of the parameters of the Query Aggregator, even while the system is running, which allows experimenting with various values of the persistence and filtering model, and with different lists of keywords and stopwords.

2.5 Evaluation, Feedback and Future Plans

The *performance evaluation* of the ACLD application is the topic of future work. One can of course test the performance of the retrieval system in terms of precision and recall, but this requires the definition of a ground truth document set for each time interval of a meeting, which is the main difficulty for such an evaluation. Three approaches of the ACLD evaluation problem are planned:

1. Setup an offline experiment where subjects are shown a meeting transcript (segmented into 30-second chunks) printed on paper, or alternatively a meeting record-



AMIDA Content Linking Demo

DETECTED KEYWORDS

button
16:41:00
16:40:30
button
16:40:00
chip
button
16:39:30
16:39:00
latex
chip
energy
look
16:38:30

RELATED DOCUMENTS

ES2008docs.ES2008scen.Summaries.P3sum3.txt
ES2008docs.ES2008scen.ProjectDocum
ES2008c.1400-1600.segments
ES2008c.2000-2200.segments
ES2008c.600-800.segments
InspirationUID2
16:41:00
ES2008c.1400-1600.segments
ES2008c.2000-2200.segments
ES2008docs.ES2008scen.Summaries.P3sum3.txt
Components_design2
InspirationUID2
ES2008docs.ES2008scen.ProjectDocuments.Conceptual.Desi
16:40:30
ES2008docs.ES2008scen.Summaries.P3sum3.txt
ES2008c.600-800.segments

CURRENT MEETING

PAST MEETINGS

ES2008c
ES2008b

ID: ES2008a (scenario)
Where: Edinburgh_meeting_room
When: Fri Feb 04 11:02:00 CET 2005
Who: FEE029, FEE030, MEE031, FEE032

Figure 3: Snapshot of AMIDA CLD's User Interface over meeting ES2008d. Left column shows recognized keywords (with timing), central column shows relevant documents (font size codes for importance), and right column shows a list of the previous meetings of the series, giving access to their contents (hovering over a label displays meeting metadata, as shown in the snapshot).

ing (audio/video) aligned with the transcript on a computer screen, and they are also given a set of documents (again, printed or on a computer screen). The task of the subjects is to associate to each chunk of the meeting zero, one or several documents that they believe are the most related to the given chunk. This allows us to construct ground truth data for the Content Linking Device, which can then be tested as many times as needed against this data.
The main challenge of this approach is to demonstrate acceptable inter-coder agreement on the definition of the "relevant" documents at each point in the meeting. To make the task tractable, it is likely that subjects would only be capable to keep in mind (hence potentially designate as relevant) only a subset of all available documents for a meeting (e.g. for ES2008d about 80 (short) documents), hence each subject should probably be given from the start only a small subset of documents for which he/she should be able to tell whether one of them is relevant at a given point in the meeting. Of course, this means that a lot of subjects are required: some need to work on the same sub-sets, to test for agreement, and the whole set of documents should be covered.

2. Conversely, it is possible to let users watch a meeting (with the possibility to pause) and ask them to rate the apparent relevance of each document returned by the

ACLD. Of course, this cannot measure the 'silence' of the ACLD, but only its 'noise' or purity.

3. Implement an experiment for *evaluation in use* by testing the ACLD on an ongoing meeting, and measuring how often the participants do consult the documents found by the ACLD. Several consultation modes could be defined, such as simply looking at the ACLD window, checking metadata for a document, or looking at the original version of the document. This type of evaluation appears to be very informative for an end-user system, but must be repeated every time the system is changed, which makes it very costly.

2.5.1 Feedback from Potential Users and Customers

Another type of evaluation is derived from the very positive reception of the ACLD, as well as the comments received at the AMIDA Community of Interest Workshop (Martigny, February 4–5, 2008). The ACLD was part of the interactive program (six "breakout sessions"), in which it received the visit of about twenty representatives of companies from the AMIDA COI[9], which are active in the field of meeting technology. The sessions lasted 30' each, starting with a presentation of the ACLD and followed by questions and feedback from the audience (notes were taken). In addition, the ACLD was shown at a regional technology transfer event later during the first day, providing additional feedback. The received feedback that can be grouped into three main categories, plus another one for various other remarks.

1. Graphical layout of the interface
   - the frame where the document names are displayed is too small with respect to the rest – a lot of space is lost in the interface – use a larger part of the screen for detecting documents;
   - display a larger overview of each document (e.g. as in Internet Explorer's "Quick Tabs" overview (Ctrl+Q)) in the main window, without keeping history data (i.e. past 30-second intervals);
   - avoid using too many mouse clicks: the content of documents should be more visible from the start;
   - colour code the nature of documents, and/or their relation to meetings;
   - make dates more visible.
2. Document repository
   - include documents that are known to users (from previous meetings) but also documents that are unknown to them: the value of the interface would then be to inform users about documents that they might not know of, not only the ones from previous meetings;

[9] See `http://www.amiproject.org/business-portal/about-ami/project-partners/community-of-interest/vendors`.

- include shared documents but also non-shared ones for individual users (e.g. my emails) – increase personalization, for individual use: each user could have their own document repository and their own list of keywords;
- include websites among documents, e.g. using a Google search (for instance, news sites, blogs, or websites of competitors) – without increasing too much the size of the repository;
- allow for various document categories: project-specific, company-specific, external, websites, etc.

3. Additional functionalities

   - after a meeting, email to all participants a list of (pointers to) the documents that were ”touched” (consulted) during the meeting
   - allow group use but also individual use during a meeting. For instance, allow each participant to use their own version, and record the interest of each of them in specific documents, then show to everyone information about the group use of the document repository (which ones are most clicked?);
   - implement relevance feedback: refine search technique based on user behavior (i.e. return more documents similar to those that are frequently consulted);
   - detect specifically the similarities with previous discussions and alert uses that they already had this discussion before;
   - represent the keywords as a cloud with emphasis varying with their frequency (see `http://www.quintura.com` for an example).

4. Varia

   - give the system more knowledge about the context of a meeting, and retrieve only documents related to that context (use word sense disambiguation over the query words) – improve accuracy of search;
   - both online and offline scenarios look interesting (online: meeting assistant; offline: meeting browser);
   - this application could lead to a cultural shift in the way meetings are organized: meetings could be divided into two short sessions, with an interval in the middle during which participants could consult the documents that were retrieved by the CLD;
   - frequent questions: who else is doing that?, has anyone else tried this?
   - infrequent question: how do you evaluate the quality of the result?

Finally, in a future version of the device, the architecture will be extended to include several other control modules, which will enable the user to select meetings and start/stop/-pause the demo in a simple way. This planned architecture is shown in Figure 4.

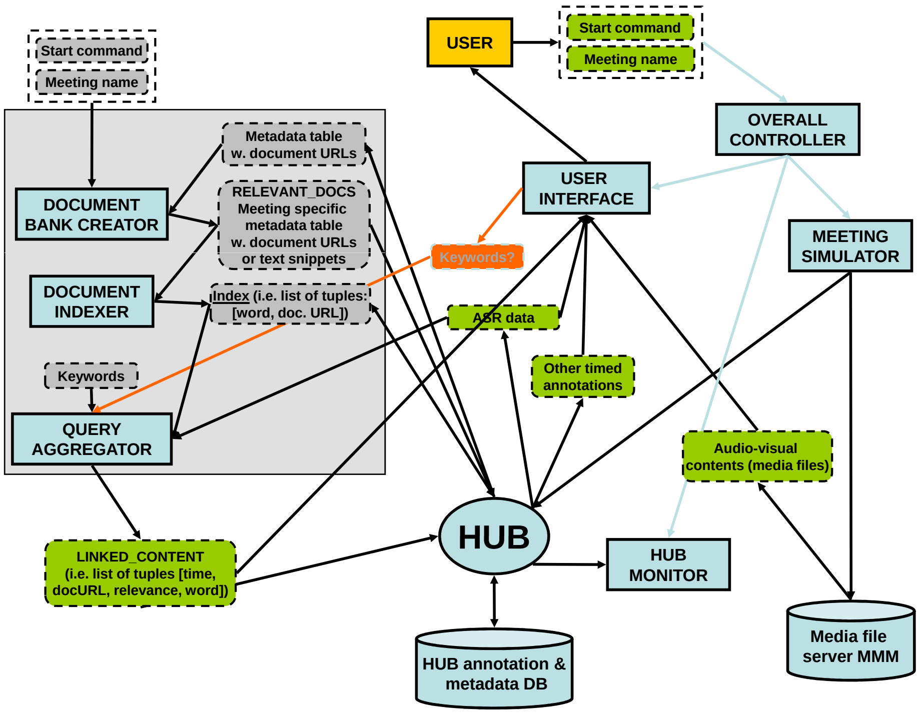

Start command
Meeting name
USER
Start command
Meeting name
OVERALL CONTROLLER
Metadata table w. document URLs
RELEVANT_DOCS Meeting specific metadata table w. document URLs or text snippets
DOCUMENT BANK CREATOR
USER INTERFACE
MEETING SIMULATOR
Keywords?
DOCUMENT INDEXER
Index (i.e. list of tuples: [word, doc. URL])
ASR data
Other timed annotations
Keywords
QUERY AGGREGATOR
Audio-visual contents (media files)
LINKED_CONTENT (i.e. list of tuples [time, docURL, relevance, word])
HUB
HUB MONITOR
HUB annotation & metadata DB
Media file server MMM

Figure 4: Future Architecture of the AMIDA Automatic Content Linking Demo.

3 The Mobile Meeting Assistant

The *Mobile Meeting Assistant (MMA)* is a graphical interface for accessing remote meetings in real time from mobile devices. For this version we focus on accessing annotated meetings of AMI corpus (Carletta et al., 2006). Two main user interfaces (2D and 3D view) are used to show the current communication situation in the meeting room. The real time aspect is based on streaming recorded meetings, because of lack of algorithms and of hardware that would be able to produce necessary annotations in real time. We state below the usage scenario and end-user requirements that we have set for a Mobile Meeting Assistant intended for a remote participant (Subsection 3.1). The design (3.2), implementation and results (3.3) are described, together with the results of a small-scale user study (3.4).

3.1 Requirements for a Mobile Meeting Assistant

A limitation of business meetings is the availability of some of the participants, e.g. due to unexpected events that prevent key persons from attending the meeting. An application that allows participating in meetings via a mobile device (a smart phone) is useful in this kind of situation. The task we investigate here is how to enable a remote user to better understand what is happening in the meeting room. Providing video stream from/to the meeting to/from the remote user is not feasible due to bandwidth constraints, and it might not also be the optimal solution, as compared to replacing the real video by a graphical representation. This might at first be seen as loosing an important part of the information, but a careful use of automatic description (annotations) of the meeting allows a Mobile Meeting Assistant to give the user *more* information than could actually be conveyed through a conventional video stream.

Indeed, even if the speaker is trying to be convincing using her words, the body language tells more to her partners in conversation (Kendon, 1981). Therefore, to help remote participants understand the maximum of the conversation, we have to express non-verbal communication in a graphical interface and (out of scope of this paper) how to detect such non-verbal means of communication. The key issue is how to represent a sufficient amount of communication signals in an intuitive interface. This version of the application deals with recorded or ongoing meetings.

As the MMA is intended for smart mobile phones, there are additional limitations such as: small size of the screen and varying light conditions, limited bandwidth, background noise and potential lack of attention. Fortunately, recent mobile phones cope already quite well with technical limitations, and computing power, bandwidth, graphic capacities and privacy concerns are gradually improving.

The MMA is a user-friendly interface that helps remote participants remain engaged in a meeting. Unlike a traditional telephone conversation, the mobile application helps remote users not only to hear other participants, but also to understand the non-verbal communication taking place. An MMA must also help remote users to access additional information about speakers, and about meeting documents such as presented slides[10]. Therefore, the following features that are particularly important for prospective users of an MMA:

[10]Conversely, the MMA could convey to the meeting room an avatar representation of the remote user,

- give access to basic channels of the meeting: sound, slides, documents, ASR/transcript;
- display addressee, i.e. allow the remote user to understand who is speaking to whom;
- notify the user that certain keywords have been pronounced;
- use personalized graphical representation of speakers;

3.1.1 Other Devices and Representations

Commercial video-conferencing equipment (e.g. Tandberg, `http://www.tandberg.com` or HP Halo, `http://www.hp.com/halo`) uses internet video streaming or broadcast and special hardware (e.g. the TotalView phone from Be Here). Video calls are available in many P2P communication applications and they are even already available for mobile phones in WCDMA/UMTS networks. But to our best knowledge, there is no wide spread working application that would use automatic graphical representation of a remote scene based on real-time events (excluded emoticons), as confirmed by a recent survey about the research in this area (Nijholt et al., 2006).

An original representation has been presented by Harry Drew in his Infospaces project (Drew and Donath, 2008). The Infospaces environment keeps track of all statements produced by each speaker, focusing on the agree/disagree dimension. This representation is probably too complex for the limited screen size of a mobile device, hence too confusing for the general user, but it is an example of an alternative graphical representation of a meeting flow (the Infoavatars proposed by the same authors shows how to represent the history of an interaction on a user's representation).

3.2 Design of the MMA

The design method that was followed is based on the six-step guidelines for building a user interface proposed by L. Chitarro (Chittaro, 2006).

Mapping. The meeting processing application outputs time-stamped annotations of the meeting in real time, and sends them to the Hub, from where the remote user's device (mobile phone) can retrieve them using a subscription mechanism. Each of the incoming annotations must then be mapped into a multimedia event, within a virtual space using simple avatars for participants, photographic representation of slides, red-color flashing highlighting for speech events, and green color arrows for the object of the visual focus of attention. We reserve sound and/or vibration events for further use, especially for keyword spotting, domain of interest warning, and "expected to speak" events (a notice to the remote participant that his people in the meeting room are expecting him to speak, i.e. give him the turn in conversation).

Selection The meeting processing tools provide, in principle, a wide range of abstracted information from the meeting, for both short time events (e.g. posture), as well as long term observations (e.g. automatic summary of a meeting). Moreover, multiple audio and video streams are available. It is, of course, virtually impossible and highly unpractical

so that the other participants also improve their understanding of his presence, as the avatar might express some of his communication states.

to fit all this information into the tiny screen of a mobile phone (usually 320x240 pixels). Therefore, given the goals of the MMM mentioned above, the following annotations were selected:

- head orientation and joint visual focus of attention (i.e. “who is looking to whom’), giving an idea of the dynamics of the meeting;
- automatic speech recognition, used to show mouth movement;
- speaker segmentation (i.e. “who speaks when”);
- clothes color, represented on the avatars[11].
- slide content and slide change.

In addition, the speech signal from the meeting room is conveyed to the remote user (a technical solution using VoIP is currently under implementation). The user study described in Subsection 3.4 below confirms that the selected features appear to be highly relevant to potential users.

To further reduce the amount of information displayed at the same time, we decided to show the focus of attention of only one meeting participant selected by the user at the same time. We experimented with automatic selection of the participant whose focus of attention to show but we have found out that the speaker changes sometimes so fast, that it was difficult to follow by the remote user.

Annotation producers that are represented graphically sometimes generate too many events to display them all, as that would produce a “jitter” on the screen that would make the graphic interface difficult to use. Therefore the amount of data sent to the Hub was reduced by filtering over the timestamps: for example, the standard output of the head orientation producer is 25 frames per second (fps), but we use only a 1 fps rate.

Presentation. The second design task was to place all selected visual items on the device’s screen. We propose two alternatives for displaying the meeting room: the 2D and the 3D view, shown respectively left and right in Figure 6. The 2D view is a top-view like representation, where each participant is represented by a simple avatar consisting in three basic shapes to represent the body and arms. The person or place in the focus of attention of a participant is marked by a light-green frame. The head orientation is represented by a sector starting from the avatar, which becomes red when the person is speaking.

The 3D view gives the user the possibility to change her view of the virtual representation of the meeting room, for instance by turning around the meeting table or by zooming on the slide-screen. The focus of attention is represented by a light-green arrow, while the speaking status is shown by a red moving lips.

The other steps in the design are **interactivity**, **human factors** and **selection of prototypes**. We want to engage the remote user in the meeting, i.e. not only watch it, but also to discuss, express her feelings and opinions. In the current version, the user interaction *with the interface* is limited to switching the interface, selecting the participant whose focus of attention will be shown and navigation in the 3D representation. Human factors such as perceptual and cognitive capabilities have guided the design as explained above; in addition, we placed text on surface with enough contrast; we doubled the thickness of focus

[11]As no automatic processor was available, therefore this metadata item was added manually. More personalized avatars could be used in the future.

of attention frame. Also the most important features (focus of attention and speaking/non speaking) are represented by signal colors (light green, red) and the color scheme stays the same in both representations. Finally, initial steps for the selection of the best solution were taken through the user study described below in Subsection 3.4.

3.3 Implementation of the MMA and Results

The three main parts of the presented system are: Smart Meeting Room (SMR) (Moore, 2002), the Hub (AMIDA, 2007), the media/document server and mobile phones with Mobile meeting application. The data flow is represented on Figure 5.

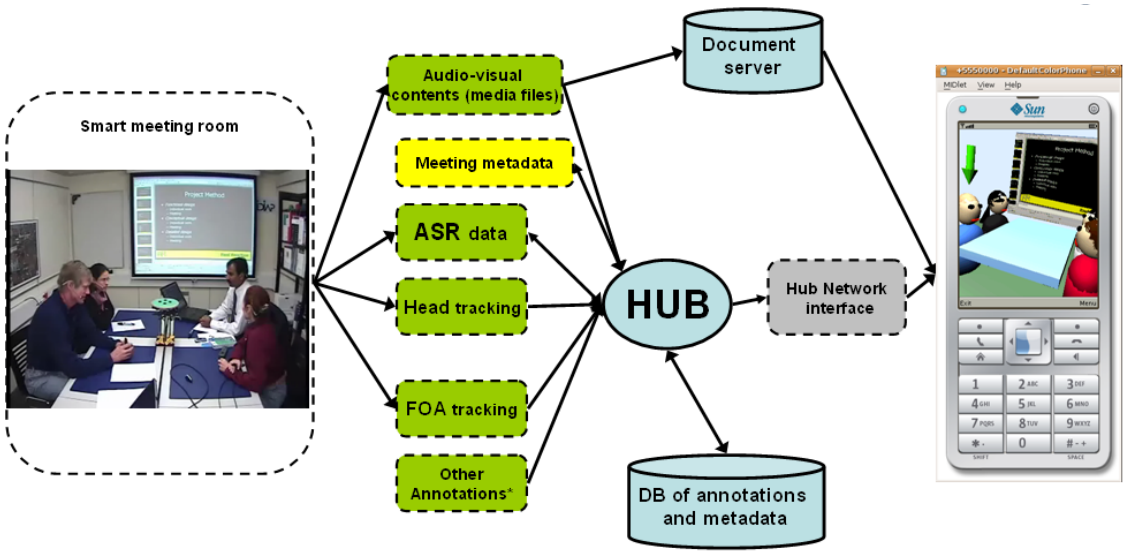


Figure 5: Architecture of the Mobile Meeting Assistant

The meeting takes place in an instrumented meeting room (SMR). Consequently the audio and video streams for each participant as well as for the overall scene are captured (and recorded). Automatic feature extraction software is used to produce annotations such as automatic speech recognition (ASR), speaker segmentation, head orientation, and most important here, the *visual focus of attention (VFOA)* (Ba and Odobez, 2006). The architecture supposes that these algorithms are running in real time and that resulting annotations circulate through the real-time data distribution entity, the Hub. However, as some of these algorithms do not run in real time at present, their output is simulated on past meetings from offline annotations.

We have implemented a Java 2 Mobile Edition interface that enables the communication with the Hub using an Internet connection. The Java application subscribes to the annotation flow from a particular meeting and is listening for updates. Upon reception of an annotation element, it renders this element directly to the GUI. If the annotation consists of an URI of a document (a slide capture for example), the application obtains the document from the document server, and displays it in the user interface.

Two types of interfaces were implemented. The actual 2D interface is represented on the left side of Figure 6, while one type of 3D interface is represented on the right side of the same figure.

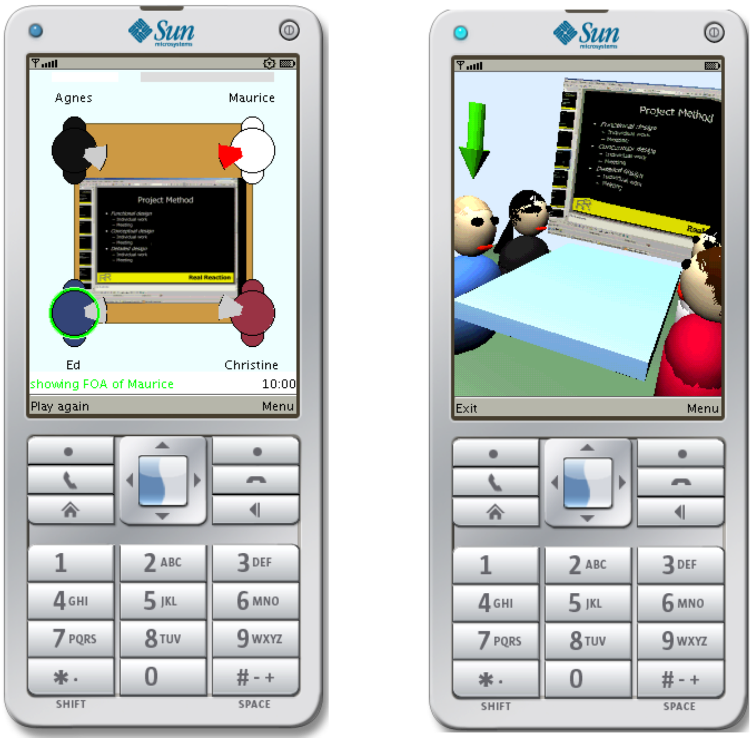


Figure 6: 2D and 3D interfaces displayed in a mobile phone emulator.

3.4 Evaluating the MMA through a Pilot User Study

A small-scale user study was performed with 13 subjects, all of whom use information technologies every day and have a university degree in computer science. The subjects were given a demo of the MMA application running on an emulator in real-time, with a video recording of the meeting playing on second computer, for a duration of 5 minutes (meeting IS1008a from the AMI Corpus). They had then the possibility to interact with the application, for instance to change the interface or the viewing angle, for a maximum duration of 5 minutes.

The subjects answered a questionnaire shortly after the demonstration and had a possibility to add personal comments. Some questions required them to rate various aspects of the MMA device, as well as possibilities for future implementations, and other questions dealt with their own needs for a remote meeting assistant. Numeric ratings are coded from 1 to 5, 1 being best and 5 worst.

The subjects judged the MMA very positively, as they liked the concept (1.5/5) and the present approach (1.9/5) – if a positive answer is counted if 1 or 2 was the response, 92% liked the general idea and 85% liked the current approach. They would use such an application "sometimes" (9 out of 13), mainly for design/technical meetings (11 out of 13) or business meetings (10 out of 13), but less for personal meetings (5 out of 13). They would mainly use the application while waiting at the train station or at the airport (10 out of 13), in the office or on a train/airplane (9 out of 13 both). The main limitations for use in such conditions is the available attention if the user must do something else (e.g. go to a gate or catch a train), the small size of the screen, and noise from the environment (7 out of 13 both).

In terms of the users' experience, they seem equally satisfied with the 2D and the 3D interfaces (2.3/5 and 2.2/5)[12]. The interface and color schemes are at the appropriate level of complexity (11 out of 13). The most appreciated information is "who is speaking when/to whom" (1.7/5), followed by the full-screen slide preview (2.0/5), the focus of attention (2.1/5) and head orientation (2.5/5). Possible features to be added in the future have been rated similarly: "you are expected to speak" alert seems the most desired one (2.0/5), followed by "enter/leave room" (2.2/5), use of personalized avatars (2.4/5), and speech transcript (2.5/5)[13].

Finally, most of the subjects would also use a desktop version of the MMA (11 out of 13), and some would even prefer it (8 out of 13), a fact that meets some of the explicit suggestions received from industrial partners.

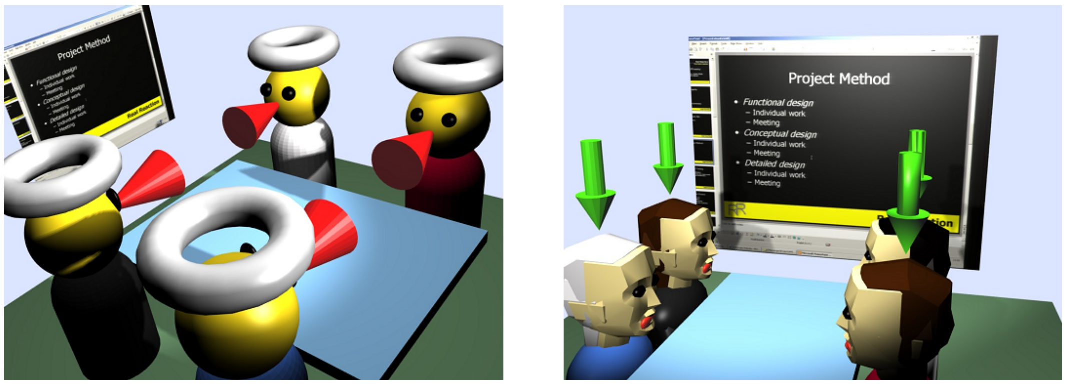


Figure 7: Prototypes of "funny" and "advanced" 3D interfaces.

3.4.1 User Study: Questions and Answers

We summarize below the questions that were asked in the user study, and the answers obtained from 13 subjects in [square brackets].

Q.1 Experience: please give us your opinion on the application that you just have tested. Rate from 1 (you like a lot) to 5 (you don't like at all) the following interfaces: 2D [AVG=2.3]; 3D standard [AVG=2.2]; 3D funny [AVG=2.3]; 3D advanced [AVG=2.7].

Q.2 Interface and colors used in the main demo were: too simple [8%] / just fine [84%] / too complicated [8%].

[12]We observed in fact a variety of acceptance for each interface when doing the study: while an important proportion of users prefer a representation which is close to reality – such as the 3D representation on the right of Figure 6 or even more realistic as in Figure 7 – while others prefer a simple 2D representation, or even funnier alternatives. Several users suggested the possibility to upload their own avatars for each participant.

[13]In other words, regarding the selected media channels and annotations, the user study showed that 77% of users would like to see who is speaking when and to whom, 69% of users were interested to know what participants are looking at (focus of attention). Another strongly requested feature was the full screen slide preview (by 69% of participants). Finally the expected to speak feature was appreciated by 73% of participants of the poll.

Q.3 Please rate following and planned features from 1 (you like a lot) to 5 (you don't like at all): who is speaking when/to whom [AVG=1.7]; entered room/left meeting alert [AVG=2.2]; possibility to create/upload personalized avatars [AVG=2.4]; head orientation [AVG=2.5]; focus of attention [AVG=2.1]; full-screen slide preview [AVG=2.0]; you are expected to speak! alert [AVG=2.0]; accurate representation of where people are sitting [AVG=2.8]; textual transcript of the meeting [AVG=2.5].

Q.4 Think about the possible use of this application in your perspective. What sort of meeting would you like to participate remotely in: business: yes [77%] / no [0%] / don't know [23%]; design/technical meeting: yes [85%] / no [15%] / don't know [0%]; personal (family, friends): yes [38%] / no [46%] / don't know [16%].

Q.5 Where would you use it? In the office: yes [69%] / no [23%] / don't know [8%]; On the train/airplane: yes [69%] / no [23%] / don't know [8%]; At the airport/train station: yes [84%] / no [16%] / don't know [0%]; In a car: yes [33%] / no [59%] / don't know [8%]; Other situations: yes [33%] / no [0%] / don't know [67%].

Q.6 What limitations do you see for use on the road? Select as many as you wish: environment too disturbing to be on meeting [7/13]; privacy concern [4/13]; screen/device too small [7/13]; time available [1/13]; attention available (driving the car, switching trains) [10/13].

Q.7 Overall rating. Rate from 1 (you like a lot) to 5 (you don't like at all) the following questions. How do you like the idea? [AVG=1.5] How do you like our current approach? [AVG=1.9] Is it easy to understand who is speaking? [AVG=2.2] I could imagine being more engaged in the meeting. [AVG=2.5]

Q.8 When ready and adapted to your needs would you be using it? Regularly [2/13] / sometimes [9/13] / never [2/13].

Q.9 Desktop version. Would you prefer a desktop version? yes [61%] / no [23%] / don't know [16%]. Would you also use a desktop version? yes [92%] / no [8%] / don't know [0%].

Q.10 Additional comments. Give us a personal feedback, hint or request a feature.

3.4.2 User Study and COI Workshop: Suggestions

From the user study questionnaire (Q10), we reformulate and synthesize from actual users' feedback the following ideas. They are grouped into two (inter-related) categories: evaluation of demonstrated system, and suggestion for the future.

Evaluation from user study:

1. The demo looks quite intuitive to use.
2. The demo is of great use for someone who is relatively new to the meeting participants, but when participants are familiar with each other, this graphical information looks superfluous, and speech could be enough.
3. 2D graphical conventions are not clear enough (ask a good graphic designer).
4. It is uncertain that 3D adds anything new, but visual complexity. (*Another user:* 3D interfaces look more like a gadget than a really useful feature.)

5. Subtle body language is not conveyed – reactions such as signs of disagreement and agreement.
6. The interfaces are too primitive to be convincing (realism is not required, but richness and clarity are).
7. The slides should be more readable (having an unreadable black board is more disturbing than helping).
8. The most useful informations (who is speaking and the slides) should remain the most visible compared to the other features.

Future developments suggested by the user study:

1. Give people in the meeting a means to see the presence of the remote user virtually (otherwise the remote user is only watcher/listener). To be active in the meeting, they need to be present virtually.
2. Record the meeting so that one can track back on interesting parts, and use speech transcription for indexing.
3. Try to spread this application widely, not only for business use but also for personal communication.
4. Solve also the challenges of a real phone (using data and voice at the same time).
5. Can speaker and VFOA annotations be done in real time? What if two persons speak simultaneously?

Feedback from COI workshop participants mainly concerns ideas for future improvements or developments. This feedback was obtained during the "breakout sessions", in which the MMM received the visit of about twenty representatives of companies from the AMIDA COI[14], which are active in the field of meeting technology. The sessions lasted 30' each, starting with a presentation of the ACLD and followed by questions and feedback from the audience. In addition, the system was demonstrated at a regional technology transfer event later during the first day, providing additional feedback.

1. Add more interfaces among which the user can choose, some of them possibly very abstract (position of participant corresponding to company hierarchy etc.). Let users choose their own avatars, possibly also used in other programs (e.g. `http://www.wee-mee.com`). Let users define their own meeting rooms (simple form / using image / select from a list).
2. Add access to information about other persons involved in the meeting.
3. Allow document sharing.
4. Integrate actual photos/camera streams.
5. Include visualization of longer-term meeting features (social network).
6. Keep the interface very simple.
7. The display of slides in real-time on the mobile phone could be turned very easily into a product.
8. The system could also be used for purely remote meetings, to create a feeling of meeting room.

[14]See `http://www.amiproject.org/business-portal/about-ami/project-partners/community-of-interest/vendors`.

9. Test it on real device.
10. Make a desktop version with more features.

The MMA appears thus quite intuitive to use, and seems especially useful to someone who is new to the meeting participants; however, 2D graphical conventions could be made clearer, and the 3D representation seems unnecessarily complex to some users. In both cases, slides should be made more visible, as well as the identity of the speakers. Ideally, more subtle body language, such as signs of disagreement and agreement, should be conveyed.

More realism was also required, e.g. by including actual photos of users, or at least letting them choose their own avatars. Accessing information about the other participants as well as meeting documents was another suggestion, while the slide capture itself appeared to be a good candidate for a commercial product. Of course, a realistic system running on a physical phone is a primary objective, bringing the audio stream to the user via the data stream (using VoIP), and synchronizing it with annotations and with the graphical representation of the meeting.

An important development will be the converse representation of the remote participant into the meeting room, because people in the meeting also need to improve their understanding of his/her presence beyond pure speech. At this point, the system could also be extended to support purely remote meetings, so that it creates the feeling of a meeting room using virtual reality.

3.5 Future Work

There are several development axes that we want to investigate further. First there is a need of improvement of the user interface so that it is easier to understand. The second step is to enable users to access additional information about participants and about meeting context in a convenient way. An important task is also to bring the audio stream to users via data stream and to play it locally (synchronized with the graphical representation of the meeting).

Another task will take into account the use of HTML as light-weight platform for remote meeting browsing as well as remote meeting participation for business people on the move. Persons will be able to hand-over a mobile phone session to a stationary device and vice versa in their hotel room, for instance.

Finally, the implementation of the remote participation in meetings is another goal of this project. This will include voice streaming and the control of the remote participant's virtual representation in the meeting room. These design steps should be backed up when possible by small scale user studies, and a large-scale user study will be performed before starting work on remote's participant interaction.

References

AMIDA (2007). Commercial component definition. Deliverable 7.2, AMIDA Integrated Project IST033812 (Augmented Multi-party Interaction with Distance Access).

Ba, S. and Odobez, J.-M. (2006). A study on visual focus of attention recognition from head pose in a meeting room. In Renals, S., Bengio, S., and Fiscus, J. G., editors, *Machine Learning for Multimodal Interaction III*, LNCS 4299, pages 75–87. Springer-Verlag, Berlin/Heidelberg.

Budzik, J. and Hammond, K. J. (2000). User interactions with everyday applications as context for just-in-time information access. In *International Conference on Intelligent User Interfaces*.

Carletta, J., Ashby, S., Bourban, S., Flynn, M., Guillemot, M., Hain, T., Kadlec, J., Karaiskos, V., Kraaij, W., Kronenthal, M., Lathoud, G., Lincoln, M., Lisowska, A., McCowan, I., Post, W., Reidsma, D., and Wellner, P. (2006). The ami meeting corpus: A pre-announcement. In Renals, S. and Bengio, S., editors, *Machine Learning for Multimodal Interaction II*, LNCS 3869, pages 28–39. Springer-Verlag, Berlin/Heidelberg.

Chittaro, L. (2006). Visualizing information on mobile devices. *Computer*, 39(3):40–45.

Drew, H. and Donath, J. (2008). Information spaces - building meeting rooms in virtual environments. In *CHI 2008 (25th SIGCHI Conference on Human Factors in Computing Systems)*, Florence, Italy.

Franz, A. and Milch, B. (2002). Searching the web by voice. In *Coling 2002 (19th International Conference on Computational Linguistics)*, volume 2, pages 11–15, Taipei, Taiwan.

Hart, P. E. and Graham, J. (1997). Query-free information retrieval. *IEEE Expert: Intelligent Systems and Their Applications*, 12(5):32–37.

Henziker, M., Chang, B.-W., Milch, B., and Brin, S. (2005). Query-free news search. *World Wide Web: Internet and Web Information Systems*, 8:101–126.

Kendon, A. (1981). *Nonverbal communication, interaction, and gesture: selections from Semiotica*. Mouton, Berlin.

Moore, D. J. (2002). The IDIAP Smart Meeting Room. Communication 02-07, IDIAP Research Institute.

Nijholt, A., Rienks, R., Zwiers, J., and Reidsma, D. (2006). Online and off-line visualization of meeting information and meeting support. *The Visual Computer*, 22(12):965–976.

Popescu-Belis, A. and Estrella, P. (2007). Generating usable formats for metadata and annotations in a large meeting corpus. In *ACL 2007 (45th International Conference of the Association for Computational Linguistics) Interactive Poster and Demonstration Sessions*, pages 93–96, Prague, Czech Republic.

Rhodes, B. J. and Maes, P. (2000). Just-in-time information retrieval agents. *IBM Systems Journal*, 39(3-4):685–704.

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, USA.