NIELS OLE BERNSEN, LAILA DYBKJÆR
AND MYKOLA KOLODNYTSKY

AN INTERFACE FOR ANNOTATING NATURAL INTERACTIVITY

Natural Interactive Systems Laboratory, Denmark

Abstract. The need for annotated corpora in the field of natural interactivity and its sub-disciplines is steadily increasing. Annotated corpora are being used to, e.g., develop and evaluate interactive speech systems, produce animated faces, and develop gesture recognition systems. The more natural the system must behave, the stronger are the demands on the quality and sophistication of the underlying corpus annotation. Thus, good tools are in high demand in order to facilitate advanced corpus annotation processes. In this chapter we describe ongoing work towards building a general-purpose tool for coding natural interactive communicative behaviour. We briefly present a review of existing natural interactivity coding tools followed by the common NITE project coding tool specification. Based on the specification, we describe our work on the interface to the NITE WorkBench for Windows, or NWB.

1. INTRODUCTION

Annotation of spoken dialogue data has emerged as an important field of research during the past 10-15 years. A key factor driving this development, although by no means the only factor involved, is the need for annotated data for the development and evaluation of interactive speech systems, such as spoken language dialogue systems and spoken translation systems. As the sophistication of interactive speech systems increased, so did the need to better understand spoken interaction. Spoken language is the core communication modality in standard *situated* communication, having developed to efficiently serve human-human communication in shared space, time, and situation, physical and otherwise. The tremendously rich speech signal reveals as much about the speaker's personality and mental states as it informs and directs the interlocutor(s). One way of describing and analysing the expressive richness of spoken dialogue contributions is in terms of a potentially very large number of *levels of analysis,* such as prosody, syntax, semantics, co-reference, speech acts, speaking style, discourse structure, spoken turn-taking cues, etc. From this point of view, the issue of spoken dialogue data annotation is one of mastering as many analytical levels as possible through the development of appropriate coding schemes for those levels. Moreover, those levels are interrelated in many ways, such as when prosodic stress acts as a cue for lexical disambiguation.

Even if we are still far from having scientifically sound, i.e. complete and semantically unambiguous, coding schemes for all levels of analysis of spoken dialogue, researchers and technology developers are moving beyond spoken human-

T.H.E. Editor(s) (ed.), Book title, 1—6.

system dialogue towards the long-term goal of achieving *natural interaction* between humans and machines. In natural human-system interaction, humans exchange information with machines in the same ways in which they exchange information with one another [Bernsen 2001]. In fact, common situated human-human communication is naturally interactive, participants exchanging information not only through speech but also through facial expression, gaze, gesture, head and body movement and posture, object manipulation including the handling of objects which themselves express information, such as texts, images, etc. Animated graphical interface agents capable of some amount of spoken dialogue, humanoid robots with similar capabilities, and combined speech and gesture input understanding systems all illustrate the emerging trend towards the development of increasingly natural human-system interaction [Cassell et al. 2000].

Putting the trends described above together, it seems clear that current needs for properly annotated data and scientifically sound coding schemes at all levels of analysis of human communication as well as cross-level and cross-modality, are larger than ever before. Arguably, a key factor in accelerating the development of high-quality annotated data as well as of new coding schemes, is the availability of appropriate coding tools. Good coding tools make data annotation and analysis faster and more efficient, and the sheer complexity of some of the coding schemes we need for coding natural interactive behaviour makes it difficult to even contemplate developing them without having an appropriate coding tool at hand. Without such tools, far fewer and far smaller annotated data resources will be created, and new, much needed coding schemes will be developed at a far slower pace or will not be developed at all. For instance, it is not known how human speech and speech prosody, facial expression, gaze, eyebrow movement, head movement, etc. manage to express, in a completely coordinated way as produced (mostly) unconsciously by the human communicator, the particular command focus of the utterance “You MUST go home now”. The problem is not just that scientifically sound coding schemes are missing for some of the levels of analysis involved, such as hand gesture. Even if the relevant coding schemes were available, it is hard to see how it would be possible to explore the complex regularities involved in the example without using a coding tool which enables cross-level and cross-modality coding for all the levels of analysis involved. Such a tool does not exist today.

This chapter describes ongoing work towards building a user-friendly general-purpose tool for coding natural interactive communicative behaviour. The perspective adopted is a partial one, as we will focus eventually on our own work on developing the user interface for the tool. Before describing this work, we describe the state of the art in coding tools for natural interactive communication (Section 2). We then (Section 3) present the European collaborative project NITE in which the work is being carried out. NITE has three development strands going on in parallel, one of which is the development of the NITE WorkBench (NWB) described in this chapter. The strands all aim to create natural interactivity coding tools, or toolsets. Section 4 discusses the different user groups who might want to use a future general-purpose tool. Section 5 presents a high-level requirements specification for a general-purpose tool. Based on those requirements, Section 6 presents requirements to the tool's annotation interface. Section 7 presents the current state of the NWB interface. Concluding the chapter, Section 8 discusses work still to be done.

2. TODAY'S NATURAL INTERACTIVITY CODING TOOLS

This section discusses current needs for natural interactivity data annotation and annotation tools (Section 2.1) and reviews some of the most prominent tools and tool projects in the field (Section 2.2).

2.1 Current needs

As argued in Section 1, current needs for annotated natural interactivity data span academic research on all aspects of natural interactive communication behaviour, including, e.g., human communicative behaviour, prosody, linguistics, psychology, anthropology, disappearing languages and cultures, and human factors, research prototype development of interactive systems which include increasing amounts of natural interactivity, and the emerging commercial development of limited-capability natural interactive systems. Based on those needs, annotated natural interactivity corpora are being used for a range of purposes, including information gathering, coding scheme research, component training, component evaluation, systems design and development, multimodal human-computer interfaces, talking heads, embodied agent design, animation, audio-visual speech recognition, automatic person identification, language learning, and lie detection. The dialogues that people may want to annotate span two-party human-human dialogue, multi-party human-human dialogue, such as small and large group discussions, dialogue among adults and among children, dialogue between people from the same or different languages and cultures, and human-computer dialogue.

It is important to note that those needs and purposes have emerged gradually over time and in a highly distributed fashion. No single person, research unit, or company have them all. Rather, the general trend has been that each group collected its own special-purpose data resources, whereupon those resources were coded using equally special-purpose, home-grown annotation tools. In the field of spoken dialogue corpus annotation, level-specific coding tools gradually emerged – for morphosyntactic annotation, co-reference annotation, dialogue acts annotation etc., as described in the MATE (Multi-level Annotation Tools Engineering) project report on the state of the art in spoken dialogue annotation tools [Isard et al. 1998]. All of those tools were either completely level-specific or very limited as regards their multi-level coding capabilities. To our knowledge, the MATE Workbench which appeared in 2000 is still the only fully multi-level and cross-level spoken language dialogue coding tool around. However, this tool still has important limitations, such as being fragile and lacking an appropriate user interface for the average user.

One reason why we do not have a general-purpose coding tool yet has to do with standard data formats for annotated corpora. Without emerging standards to aim for, tool developers risk betting on the wrong horse. It is only recently that XML (eXtensible Markup Language) has emerged as a possible standard for representation of annotated data. A second reason why it is far from easy to develop a general-purpose tool is the lack of a general markup framework. The tool should allow its users to enter their own coding schemes in order to be able to code any set of phenomena in natural interactive communication. This requires that the tool embodies a general-purpose coding framework based on which the user can be told which information must be included in order to create a valid coding scheme which the system can cope with. Such a framework does not fully exist yet.

2.2 Existing annotation tools

Considering the state of the art in natural interactivity coding tools, we find a variety of home-grown, limited-functionality, special-purpose, and level-specific coding tools from among which somewhat more general tools are beginning to emerge. The ISLE NIMM Working Group report on natural interactivity and multimodal corpus annotation tools [Dybkjær et al. 2001a] describes twelve tools and projects which support annotation and analysis of (spoken) dialogue, facial expression, gaze, gesture, and/or bodily posture, etc., and possibly cross-level or cross-modality issues as well. Figure 1 shows the reviewed tools and tool projects. In Figure 1, *Tool* is the name of the tool or project reviewed. *NIMM (Natural Interactivity and Multimodality) aspects addressed* are the NIMM aspects which a particular tool is explicitly claimed by its developers to support. *Brief tool/project description* is a brief description of the tool or project, including a web address.

Tool: Anvil. *NIMM aspects addressed:* Speech and gesture. *Brief tool description:* Annotation of video and language data. A Java-based tool for annotating digital video files. See www.dfki.de/~kipp/anvil

Tool: ATLAS. *NIMM aspects addressed:* No tool was available at the time of review. jATLAS has been added since then. *Brief project description:* Architecture and Tools for Linguistic Analysis Systems. See www.itl.nist.gov/iaui/894.01/atlas

Tool: CLAN. *NIMM aspects addressed:* Text, speech and gesture. *Brief tool description:* Computerised Language Analysis. A program designed specifically for analysing data transcribed in the format of the Child Language Data Exchange System (CHILDES). Transcriptions can be linked to audio or video files. See childes.psy.cmu.edu

Tool: CSLU Toolkit. *NIMM aspects addressed:* Speech, TTS and facial expression. *Brief tool description:* Center for Spoken Language Understanding Toolkit. A suite of tools based on the CSLUsh programming environment and including an annotation tool called OGIsable. It allows the user to attach properties to a text before it is spoken, e.g. to synthesise facial expression synchronised with speech output. See cslu.cse.ogi.edu/toolkit/

Tool: MATE Workbench. *NIMM aspects addressed:* Speech and text. *Brief tool description:* Multi-level Annotation Tools Engineering. A Java-based tool in support of multi-level annotation of spoken dialogue corpora and information extraction from annotated corpora. See mate.nis.sdu.dk

Tool: MPI tools: CAVA and EUDICO/Computer Assisted Video Analysis and European Distributed Corpora. *NIMM aspects addressed:* Speech and gesture. *Brief tool description:* Both tools support annotation of audio-visual files and information extraction. See www.mpi.nl/world/tg/CAVA/CAVA.html, www.mpi.nl/world/tg/lapp/eudico/eudico.html

Tool: MultiTool. *NIMM aspects addressed:* Speech and gesture. *Brief tool description:* MultiTool was developed in a project on a Platform for Multimodal Spoken Language Corpora. A Java-based tool in support of the creation and use of multimodal spoken language corpora (audio and video). See www.ling.gu.se/multitool

Tool: The Observer. *NIMM aspects addressed:* Gesture and facial expression. *Brief tool description:* A commercial system for the collection, analysis, presentation, and management of video data. It can be used to record activities, postures, movements, positions, facial expressions, social interactions or any other aspect of human or animal behaviour as time series of tagged data. See www.noldus.com/products/index.html?observer/

Tool: Signstream. *NIMM aspects addressed:* Speech, gesture and facial expression. *Brief tool description:* Signstream was developed as part of the American Sign Language Linguistic Research Project. A database tool for analysis of linguistic data captured on video. See web.bu.edu/asllrp/SignStream

Tool: SmartKom. *NIMM aspects addressed:* No tool available. *Brief project description:* This large-scale project aims to merge spoken dialogue-based communication with a mixture of graphical user interfaces and gesture and mimetic interaction. SmartKom uses tools developed elsewhere: in Verbmobil for audio annotation, Anvil for mimics and gesture coding. See www.smartkom.org

Tool: SyncWriter. *NIMM aspects addressed:* Speech and gesture. *Brief tool description:* A commercial tool for transcription and annotation of synchronous "events" such as speech and video data. See www.sign-lang.uni-hamburg.de/software/software.html

Tool: TalkBank. *NIMM aspects addressed:* Speech (Transcriber), text, speech and gesture (CLAN), see above. *Brief project description:* This project aims to provide standards and tools for creating, searching, and publishing primary materials via networked computers. Transcriber and CLAN had been incorporated as part of the project at the time of review. AGTK: Annotation Graph Toolkit has been added since then. See www.talkbank.org

Figure 1. Overview of 12 natural interactivity and multimodality coding tools and projects. Adapted from Dybkjær et al. 2001a.

A couple of tools had not been implemented at the time of the review. However, the tool concepts presented by the projects aiming to develop the tools were found sufficiently interesting for including a project description in the ISLE NIMM survey, e.g. because the project has standardisation among its goals. Two tools are commercial, i.e. The Observer and SyncWriter. The rest are research tools (or projects). The MATE Workbench only supports spoken dialogue and text annotation. This tool was included in the survey because of its advanced capabilities for multi-level and cross-level annotation, which may point the way towards building a general-purpose natural interactivity coding tool. The CLSU Toolkit coding tool is for output generation only. Finally, so far, at least, the SmartKom project is a user rather than a provider of NIMM coding tools.

Modalities addressed by the tools

Speech annotation is addressed by all the tools in Figure 1 but one (The Observer), i.e. by nine tools. Gesture annotation is addressed by seven tools. Facial expression annotation is addressed by three tools only. One tool focuses on speech annotation only (Transcriber (TalkBank)). One tool focuses on speech and text annotation (MATE Workbench). One tool handles speech, text and gesture annotation (CLAN). Four tools address speech and gesture annotation (Anvil, the MPI tools, MultiTool and SyncWriter). One tool focuses on speech, gesture and facial expression annotation (SignStream). One tool handles speech, facial expression and text-to-speech annotation (CSLU Toolkit). One tool addresses gesture and facial expression annotation (The Observer).

Among the tools reviewed, the MATE workbench is the most advanced tool as regards markup of spoken dialogue. It comes with a number of example coding schemes for different annotation levels, such as dialogue acts and co-reference. Most of the other tools are capable of addressing spoken dialogue annotation in a multimodal context. As regards their capabilities for spoken dialogue annotation, these tools either do not go beyond the transcription level or they offer, at most, single-level annotation of, e.g., dialogue acts (Anvil). Annotation is done either according to a built-in annotation scheme or, better, according to an annotation scheme which can be modified by someone with the required skills in, e.g., XML.

Several of the reviewed gesture annotation tools could probably, with more or less effort, be extended to handle markup of facial expression. When it is not mentioned in Figure 1 that a gesture annotation tool supports facial expression annotation, this is typically because the tool does not include a coding scheme for facial annotation. It should be noted that if the coding scheme is hard-coded into a tool, it is not necessarily easy to add new coding schemes to the tool. Most of the tools for gesture annotation have a number of basic functionalities in common, including the possibility of viewing video-recorded gesture, adding markup according to some built-in coding scheme, synchronising video and annotation, time-aligning coding files and raw data, and extracting coding file information. Considering the actual implementations, one of the most advanced and stable tools for gesture annotation would seem to be The Observer.

The Observer provides support for annotation of gesture as well as facial expression because it is quite easy to add new annotation schemes by using the interface offered for this purpose. To add new coding schemes, however, one must comply with the general markup framework provided by The Observer, which imposes important limitations on the structure of the coding schemes that can be added. Only two other tools (SignStream and the CSLU Toolkit) claim to support annotation of facial expression. For the CSLU Toolkit, the annotation support is intended for output generation of an animated speaking face.

Programming languages

Java, Tcl/Tk, C and C++ are the preferred programming languages for the reviewed tools. XML is used in many cases for coding file representation. The majority of the tools run on a Windows platform. Many of the tools (at least the Java-based ones) should also run under Linux and Unix (but not on MacOS). A couple of tools run on a Macintosh platform only. In nearly all cases, an executable version is freely available. In four cases, i.e. the CSLU Toolkit, the MATE Workbench, MultiTool and Transcriber (TalkBank), the source code is freely available.

3. THE NITE PROJECT

NITE (Natural Interactivity Tools Engineering, nite.nis.sdu.dk) belongs to a series of three projects which pursue the objective of creating a firm basis for progress in natural interactivity coding. NITE's direct predecessor is the MATE (Multi-level Annotation Tools Engineering, mate.nis.sdu.dk) project which involved many of the NITE partners. Running in parallel with NITE, the joint EU/US project ISLE (International Standards for Language Engineering) includes a Working Group on Natural Interactivity and Multimodality (NIMM, isle.nis.sdu.dk) whose European partners have produced a series of state-of-the-art reports on natural interactivity data resources [Knudsen et al. 2002a], natural interactivity coding schemes [Knudsen et al. 2002b], and natural interactivity coding tools [Dybkjær et al. 2001a], respectively. In addition, the European ISLE NIMM group has produced an initial specification of a general-purpose natural interactivity coding tool [Dybkjær et al. 2001b] which has served as a launch-point for ongoing work in NITE. NITE began its work in April 2001 and has a duration of two years. The partners are: NISLab (Odense, Denmark), DFE (Barcelona, Spain), DFKI (Saarbrücken, Germany), HCRC (Edinburgh, UK), ILC (Pisa, Italy), IMS (Stuttgart, Germany), and Noldus (Wageningen, The Netherlands).

NITE is developing several versions of a workbench, or an integrated toolset, for annotating and analysing full natural interactive communication among humans and between humans and systems. In many ways, NITE pursues objectives similar to those of MATE. The crucial difference is that NITE goes beyond spoken dialogue to address full natural interactivity data annotation and analysis. Thus, NITE develops a natural interactivity markup framework; identifies or develops several natural interactivity best practice coding schemes to be described following the markup framework; and builds general-purpose tools which include those coding schemes and support the addition of new ones within the general boundaries of the markup framework.

The NITE consortium follows a three-tiered approach. Based on common requirements specifications, the commercial partner Noldus works towards meeting some of the core NITE requirements in future releases of the company's widely used The Observer software package. Based on the same specifications, the NITE academic partners are developing two different open source versions of the NITE toolset. The version described in this chapter is the NITE WorkBench for Windows, or NWB, which implements the NITE specifications in a Windows environment familiar to all or most users. The second version is the NITE XML Toolkit, or NXT, which is platform independent and requires XML skills of its users [Soria et al. 2002].

As will be discussed in the next section, developers of a general-purpose tool for coding and analysing natural interactivity data face a large variety of user needs from a highly diverse user population. In view of the composition of this user population, we believe that an essential condition for building a tool which could be successfully used by the majority of those working in the field, is to provide the tool with an easy-to-use, user-friendly interface.

4. NITE TARGET USER GROUPS

Users of natural interactivity annotation tools may be divided into three groups, as follows.

The first user group includes people who need a tool which allows them to do their tasks of annotation, information extraction, and analysis without bothering about the internal design, data representation formats, and workings of the tool. Typically, these users are experts in their area, such as the understanding of the integration of speech and facial expression in human communication, and they consider the annotation tool simply as a vehicle for making their work more efficient and its results more useful and more widely available. Considering the many different disciplines whose practitioners are potential users of a general-purpose tool, it seems likely that this user group is the larger by far compared to the two following user groups.

The second user group includes users who have become so familiar with data coding formalisms, such as SGML or XML, or who find that existing editors are not sufficient for their purposes, that they feel most comfortable if they can edit the coded data directly. Thus, if, e.g., XML is being used for internal data representation, they want to have access to the XML files, possibly via an editor, even if the internal data representation is meant for the computer rather than for the user.

The third user group includes users who have the programming skills and the motivation to add new tool functionalities to an existing tool provided that the tool makes this possible. To accommodate these users, and given the fact that no natural interactivity coding tool will ever include all conceivable useful functionality, the best option is to equip a general-purpose toolset with an open architecture which enables the addition of new functionality.

NITE aims to support all three user groups through three different strands of development. NXT will support the second and third user groups. NXT is a stylesheet engine for working with complexly structured data expressed in "stand-off" form, with which users with standard XML technical skills can extract data for import into other packages, specify data transductions into other forms including ones appropriate for tools such as Anvil, The Observer, and NWB, and build end-user data displays and coding interfaces. The first and third user groups will be supported by NWB's graphical user interface and Windows environment. The first user group will be supported as well by the commercial tool (The Observer) by Noldus which will provide a limited part of the functionality provided by NWB and NXT.

In Section 5 we present the general NITE annotation tool requirements. Sections 6 and 7 focus on the NITE WorkBench and on the first-mentioned user group above, since these are the users who need a familiar and easy-to-use visual interface. NXT and The Observer are described in [Soria et al. 2002].

5. GENERAL TOOL REQUIREMENTS

Before proceeding to describe current work on the NITE WorkBench's user interface, we briefly present the NITE consortium's *core functionality* philosophy. According to this line of thinking, current tool developments in the field of natural interactivity should be observant of the fact that multiple parallel activities are already going on world-wide: in tools for orthographic and phonetic transcription, statistics packages, data representation formats, metadata standards, etc. There is no reason for a NITE tool to replicate existing tools or functionalities when these are already satisfactory. Rather, NITE should focus on the core functionalities involved in providing tools for full natural interactivity data annotation and analysis. If this task can be solved to the satisfaction of the intended users, those users can be relied upon to use already existing tools for the many other things they have to do when working with their data and which are not being provided by the NITE tools. What this requires is for the NITE tools to incorporate, to the extent possible, emerging standards for data representation, such as XML, enable data import from, and export to, already existing tools, such as advanced statistics packages, etc.

On this background it appears that a general tool in support of natural interactivity data transcription, annotation, information extraction, and analysis should satisfy at least the following requirements [Dybkjær et al. 2001b].

1. A flexible and open architecture which allows for easy addition of new tool components (a modular workbench) by the third user group described in Section 4. A general-purpose tool is not likely to cover the needs of all target users. Thus it is important that users can make their own additions to the tool and that APIs are made available which support the addition of new tool components.

2. Separation of user interface from application logic and internal data representation. The internal data representation should be separated from the user interface via an intermediate logical layer so that the former two layers can be modified separately. Good GUI software practice prescribes a three-layered structure as a minimum: the user interface, an intermediate (logical) layer, and an internal data representation layer which is hidden to standard users. Often these layers will be further subdivided. The layered structure needs not mean anything for end-users but it means a lot for developers who want to make changes either to the interface or to the internal representation.
3. Annotation (including transcription) support at different analytical levels, for different modalities, and for annotating relationships across levels and modalities. The launch version of the toolset should include a number of best practice coding schemes for annotation at different levels, for different modalities, and for annotating interaction across levels and modalities. These coding schemes should be described in a way which makes them comprehensible and easy to use. MATE proposed a standard markup framework which facilitates uniform and comprehensive description of coding schemes across annotation levels [Dybkjær et al. 1998]. The framework has been found useful for describing spoken dialogue coding schemes at multiple levels. Ongoing work in NITE aims to extend the framework to other areas of natural interactivity data coding.
4. Today, annotation at certain levels, such as (morpho-)syntactic annotation, can be done semi-automatically or automatically. To the extent that it is possible to (semi-) automate natural interactivity data annotation processes, this should be supported by the toolset. Similarly, to the extent that it is possible to (semi-) automate data analysis, this should be supported by the toolset. Automation should be supported in two ways: (i) via the possibility to add (through an API) additional components for automatic annotation and analysis, and (ii) via the use, as far as possible, of standard(ised) data formats allowing easy import and export of annotations for subsequent processing by other tools.
5. Powerful functionality for query, retrieval, and extraction of data from annotated corpora; a minimum of tools for data analysis, possibly including simple statistical tools. Powerful query and information retrieval is needed for the user to exploit the annotated data for an open-ended range of purposes.
6. Adequate support for viewing and listening to raw data. The toolset should allow users to play videos and listen to sound files or parts thereof. Also, display of sound waves should be included. There should be adequate support for navigating and searching raw data according to time information.
7. Adequate visual presentation of annotated data. The toolset should enable visualisation of timeline information and synchronised views of different layers of annotation and different modalities. The toolset should come with a set of predefined but flexible and versatile visualisations.
8. Easy-to-use interface. In general, the tool interface should support the user as much as possible, be intuitive, and as far as possible be based on interface

standards which the user can be expected to be familiar with. The query tool interface, for instance, should make it easy to specify even fairly complex query expressions, and results must be presented to the user in a sensible, intuitive and easy-to-use manner.

9. Support for easy addition and use of new coding schemes (cf. (3) above) and for defining new visualisations of annotated data, such as presenting annotations based on new coding schemes.
10. Possibility of importing and re-using existing data resources via conversion tools. Today, there is wide-spread use of proprietary systems and formats for natural interactivity data coding. This constitutes a major obstacle to creating a standard coding tool which still allows users to exploit the data resources they have built using other tools and in whatever format they have found appropriate. The only practical solution would seem to be to offer a way of adding tools for converting from the user's data format to the format used by the NITE system. It should be easy to write converters and add them since it is not practically feasible to incorporate every conceivable converter from the outset.
11. Possibility of exporting, by means of conversion tools, coded data resources for further processing by external tools.
12. Most importantly, perhaps, the tool must be robust, stable and work in real time even with relatively large data resources and complex coding tasks.

As a general note to the requirements above, it should be remarked that there is an important degree of interchangeability among the requirements concerning tool components creation and addition (1) and conversion tools (10, 11). For instance, if an adequate tool already exists for transcribing part of the resources to be handled by the NITE tool, it may be considered to create a conversion tool for importing this transcribed data instead of creating a transcription tool for the data. Or, if an adequate tool already exists for the statistical analysis of annotated resources created using the NITE tool, it may be considered to create a conversion tool for exporting those resources rather than creating a statistical analysis tool in NITE. It is probably futile to try to replace existing work practices for all user needs, first because that would spread the NITE coding effort very thinly indeed, secondly because it is hard to predict what the complete set of needs will be, and finally because people often like the tools with which they are familiar already.

6. ANNOTATION USER INTERFACE REQUIREMENTS

At the time of writing, the NITE tools are being developed in distributed fashion at the partner's sites. We are about one year into the project and roughly half-way to completion of the NITE WorkBench. In the following, we focus on NWB's annotation user interface which is the most developed part of the interface for standard users. In addition, we describe a first draft of part of NWB's coding scheme definition interface.

Based on the user needs expressed in the tools reviewed in Section 2, the NITE core functionality philosophy and general tool requirements in Section 5, and taking into account that our focal user group is those who want a tool in order to carry out their annotation and analysis tasks efficiently and without bothering about programming and internal tool data representations (Section 4), the following would seem to be the core requirements to NWB's audio-visual interface. The tool should:

1. support working with annotation projects, including meta-data;
2. enable flexible control of raw data files (audio and/or video);
3. support annotation of natural interactive communication at any analytical level and across levels and modalities through the use of existing or new coding schemes;
4. enable users to specify new coding schemes either by editing existing coding schemes or by adding new ones from scratch;
5. enable information extraction and analysis of annotated data.

With these basic requirements in mind, which have also been used to structure the tools comparison in Figure 2, the first step in the design of a visual interface for the NWB tool has been to make a series of design decisions concerning the general layout of the visual interface to ensure that those requirements can be met in a coherent and user-friendly way. In essence, the resulting requirements specification constitutes a more comprehensive, and at the same time more detailed, version of the list in Figure 2. Thus, eventually, the NITE tools will jointly tick √by all entries in Figure 2 and, in addition, enable full natural interactivity coding and some amount of file import-export. Figure 2 compares key functionality of four tools: Anvil, the MATE Workbench, The Observer, and Transcriber. These tools are of particular interest to the NITE project which: builds on the MATE workbench, includes developers of The Observer, collaborates with the site developing Anvil, and considers making a link to Transcriber for orthographic transcription. Moreover, each tool has a range of useful functionalities. They each cover some limited part of natural interactivity coding only but they are among the more advanced state-of-the art tools. For more details on the MATE Workbench, Anvil and The Observer, see [Soria et al. 2002]. For Transcriber see http://www.etca.fr/CTA/gip/Projets/-Transcriber/.

To facilitate tool access and operation by standard users (Section 4), we want the interface layout to be similar to what users are likely to be familiar with from other programs. This will ease the learning curve for novice tool users as regards basic layout and functionality. We have thus opted for a common top-screen menu line that provides access to all other system functionality, cf. points 1-5 above, including basic options such as copy, paste, save and print. Some options will always be available. Other options are specifically related to one of the requirements above and can only be selected when the user is actually doing the task addressed in the requirement.

So far, we have developed the tool's interface based on analysis of the first three requirements above, i.e. those addressing annotation support. The NITE visual interface aims to support annotation of any kind of phenomena involved in natural interactive communication. To this end, the following requirements to the annotation user interface have been specified.

Functionality	*Tools*			
	A	*M*	*O*	*T*
1. Working with an annotation project:				
create a project which includes standard metadata;	√	√	√	√
link to audio and/or video data files;	*	–	*	*
open/save a project;	√	√	√	√
print components of a project;	–	–	–	–
store and access results of information extraction analysis.	–	–	√	–
2. Controlling raw data (i.e. audio and/or video files):				
visualise and play video file;	√	–	√	–
listen to and graphically represent audio data;	*	*	–	√
open several video data windows at the same time.	–	–	–	–
3. Enabling users to specify coding schemes:				
create new annotation schemes by using, e.g., XML;	*	*	–	–
create new annotation schemes via a graphical user interface.	–	–	√	–
4. Supporting annotation via built-in coding schemes:				
annotation of gesture and facial expression;	√	–	–	–
orthographic transcription;	–	–	–	√
annotation of spoken dialogue at multiple levels;	–	√	–	–
add free-form comments;	*	*	√	√
visualise the result of the coding;	√	√	√	√
customise the visualisation of the annotation.	*	*	*	*
5. Enabling information extraction and analysis of annotated data:				
extract arbitrary parts of data;	–	*	√	–
support statistical analysis.	–	*	√	–

Figure 2. Comparison of functionality offered by Anvil (A), the MATE Workbench (M), The Observer (O) and Transcriber (T). “ √” means that the functionality is fully implemented, “” that it is partially implemented, and “–“ that it is (close to) absent. “Partially implemented" means that a tool does not fully support a certain functionality but that the functionality can be used if, e.g., the user is willing to code in XML. The “*” under Observer, first row under point 4, indicates that The Observer does not really have built-in coding schemes.*

1. *Raw data perception and inspection.* Raw data must be made perceptually accessible to the user at will. The user must be able to look at the video whenever necessary and switch it off at will. The user must be able to listen to the audio track whenever necessary and switch it off at will. The audio track and the video track must be controllable independently of one another. The same applies to the viewing of other kinds of raw data, such as log files and graphical representations of acoustic information (the latter is considered raw data for present purposes even if this might be contended). Acoustic raw data,

video raw data, and graphical representations of acoustic information should all come with a visible time-line. It must be possible to navigate back and forth in the raw data based on the timeline. It should be possible to open several video raw data windows at the same time, for instance in order to inspect complex long-range dependencies, such as cause-and-effect or co-reference.

2 *Data annotation.* Appropriate facilities must be present for annotating any aspect of the audio/video, including free-form comments on what goes on in the audio/video as well as use of standard annotation schemes for spoken dialogue, facial expression, emotion, gaze, gestures of all kinds, lip movements, bodily posture, actions, etc. Annotation is here also understood to include orthographic transcription of spoken dialogue. Phonetic transcription remains an open issue in NITE.

3. *One annotation at-a-time.* We do not expect serious users to annotate according to several annotation schemes simultaneously. This means that the structure of one palette/one annotation scheme/one layer of annotation active for annotation would be a valid one. Palettes are described in Section 7. However, since we want to support the discovery of new regularities which may well be cross-level or cross-modality, such as turn-taking cues as expressed in all natural interactive communication modalities, it should be possible to visibly link several annotation levels during annotation.

4. *Resolution levels.* Given the fact that the phenomena to be annotated in trying to understand natural interactivity may not only be quite complex per time unit, as it were, such as when a long series of facial tags are required for characterising a facial expression at a given point in time, but also temporally extended and cross-correlated to an indeterminate extent both synchronously and diachronously, it must be possible to view transcriptions and annotations at different levels of resolution. User viewing needs are likely to range from viewing few-seconds-duration cross-level, cross-modality annotations close-up to viewing minutes-long stretches of transcription and annotation birds-eye. This imposes additional requirements onto the representation of levels of annotation as well as within-level and cross-level links between annotated phenomena. For instance, an antagonistic turn-taking episode may well have a duration of, say, 20 seconds. If the annotator is coding the episode cross-level and cross-modality as represented by orthographically transcribed speech, dialogue acts, facial expression and gesture, all of these codings must be visibly present on the screen and it should be possible to view the entire episode birds-eye to verify that all relevant relationships have been marked up.

5. *Editors.* It should be possible to open simple editing windows at any time. These windows may be used for, e.g., inserting additional observations related to the annotation process, or doing preliminary experiments with new coding schemes which have not yet been specified to work with the tool. It seems quite likely that new coding schemes will often be developed in this way. The annotator starts by informally marking up phenomena, or correlations between phenomena, in a free-form editor. Only when the emerging coding scheme has

reached a state where it makes sense to specify it as a tag set will the annotator include it into the NITE WorkBench as a formally specified coding scheme.

6. *Tag palettes.* On-screen palettes must be able to refer to (or mark up) single-level phenomena as well as cross-level links. User-defined keyboard short-cuts corresponding to the tags on the palette must also be enabled.
7. *Within-level tagging.* It should be possible to clearly label time-aligned entities during annotation. Labelling of entities, if not done free-form, could be done by selecting from a palette elsewhere on the screen. The palette would contain all possible tags belonging to a particular coding scheme. It should be possible to link the following temporal entities to the common timeline: points in time (no duration), timed entities of any length, such as sub-words, filled or unfilled pauses, words, phrases, communicative acts, as well as longer-duration communication units of any kind. It should be possible to visually link to the common timeline the following types of within-level relationship: long-range dependencies, such as co-references, cause and effect, and event conditions.
8. *Cross-level, cross-modality tagging.* It should be possible to visibly link annotations at different levels and of different modalities, such as linking facial expression annotation to transcription, or linking gesture annotation to speech act annotation. It should be possible to tag these links. This will enable researchers to establish highly innovative coding schemes for coordinated cross-level, cross-modality clusters of communication phenomena.
9. *Number of displayed annotation levels.* It must be possible to display up to ten annotation levels simultaneously on the screen. Given the fact that displaying as many as ten annotation levels may be a relatively rare occurrence, allowance could be made for screen extensions to make this possible, for instance through scrolling. However, it must be possible to inspect a reasonable number of annotation levels on the screen without scrolling.
10. *Perceptible time-alignment.* Annotation levels (including orthographic transcription) should visibly share a common timeline. This means that, independently of the particular details of the way in which those annotation levels are being presented, the common timeline corresponding to the audio and/or video and/or graphical representation of acoustic events, must visibly “run through” all annotations.
11. *Display management during coding.* Suppose, for instance, that the coding at some analytical level is being displayed as a time-aligned “band” running across the screen, and suppose that tagged phenomena at that level are being visibly linked to tagged phenomena at the same level or at a different level which is being displayed as well. In such cases of visibly linked tags, it should be possible to reorganise individual links as well as all links between two different levels of annotation. It should be possible to re-order annotation representations on the screen. Re-ordering may affect cross-level links. Links should be preserved but a certain clutter may be unavoidable.

7. THE AUDIO-VISUAL ANNOTATION INTERFACE

Based on the user group in focus in this chapter, i.e. the first user group from Section 4, and the interface requirements presented in Section 6, the NITE WorkBench will have a uniform visual interface which offers the kinds of customisation known from many other programs. As the requirements suggest, the interface will otherwise have to be a rather complex one because it has to cater for the presentation of raw data, multi-level and cross-level annotation, free-style comment, annotation analysis, annotation comparison, search and inspection of search results, as well as basic interface operations such as file saving, printing, deletion, overview, duplication, import and export. In the following, we first briefly describe and illustrate the data structure model underlying the visual interface (7.1), and then present the first version of the annotation interface (7.2) and a draft of part of the coding scheme interface (7.3).

7.1 Data structure model

The overall functionalities to be supported by the interface, as described in Section 6, include annotation of natural interactive communication at any analytical level (points 1-3 in the five-point list in Section 6), specification of new coding schemes (point 4), and information extraction and analysis of annotated data (point 5).

To support these activities at the interface level, an appropriate data structure model is needed. The idea is to use “project files” as organisers. A project file basically consists of pointers to a set of raw data files, e.g. video or audio standard format files, such as *.mpg, *.avi, or *.wav, as well as meta-data descriptions of the raw data, and pointers to all coding files drawing on that raw data, i.e. transcriptions and other annotation files, and a meta-data description per coding file. A meta-data description of raw data will list all the raw data files belonging to the data set and include information about when the data was recorded, by whom it was recorded, who can be seen/heard in the recordings, etc. A meta-data description for a coding file will contain, e.g., information about which coding scheme was used, who annotated the data, when annotation took place, and pointers to raw data.

The data structure model arrived at in NITE thus consists of the following five major structures:

1. project file;
2. raw data and its meta-data description;
3. coding files and their meta-data descriptions;
4. coding schemes; and
5. display objects for visualisation and interaction.

The display objects enable the user to visualise and perform operations on the data structures referred to in points 1-4.

During annotation, the user draws on all five major data structure components. Manipulation is supported by the display objects. The files directly involved are a coding file and its meta-data description, and possibly also the project file which may have to be updated with a pointer to a new coding file. Annotation draws directly on raw data, on a coding scheme, and possibly on other coding files such as a transcription. When defining or revising a coding scheme, the display objects are being used for manipulation and a coding scheme file is being used as well. Finally, if the user's activity is information extraction, coding schemes are not being used but all the other four major data structures may be involved. The project file will tell which files are relevant, and the relevant raw data, coding files and their meta-data are the potential objects of the queries made. The display objects support the making of queries at the interface level.

The actual implementation of the data structures may be done in several different ways. The candidates for data representation which have been considered in NITE are XML files and a database solution (DBMS or Data Base Management System). Comparison with other annotation tools shows that both solutions have been used. For instance, The Observer uses the relational DBMS approach and provides good support for information extraction and analysis of annotated data. Other tools, such as Anvil, MATE, and Transcriber use XML both to store data and to supply the annotator with data visualisations, e.g. via style sheets. Both representation techniques have advantages and disadvantages. For instance, it is likely that search in large corpora will be faster when data are stored in a database rather than in XML files. On the other hand, XML is becoming a standard and facilitates exchange of data across platforms. For the NITE WorkBench, we have adopted the DBMS approach as complemented with XML conversion and export/import capabilities. NXT, on the other hand, follows the XML approach.

7.2 Annotation interface

So far, only the annotation part of the NITE WorkBench has been designed in detail at the interface level. The first version of the annotation interface is described and illustrated in the following.

To provide the user with a uniform way of doing multi-level and cross-level annotation of natural interactivity data as well as to enable several ways of displaying annotated files, the user interface for data annotation is composed of the following five main components:

1. the *main window* which includes the main menu, its title, etc. (Figure 3);
2. the main window *toolbar* which includes the changeable (content-sensitive) set of buttons (Figure 3);
3. a variable number of *panels* each of which enables annotation of a particular class of phenomena and shows the annotated phenomena. Between 1 and 10 panels may be shown at the same time (Figure 4), cf. Section 6 Point 9;
4. a variable number of *raw data windows* displaying the different types of raw data which are being annotated (Figure 4), cf. Section 6 Point 1;
5. the common *control board* for controlling the active raw data window(s) (Figure 5), cf. Section 6 Point 1.

In addition, numerous palettes (dialogue boxes) with built-in controls are provided or can be defined by the user in order to work with different coding schemes, for instance in order to insert or delete tags, visualise tags, etc. (Figure 5), cf. Section 6 points 2, 6, 7, 8, and 11.

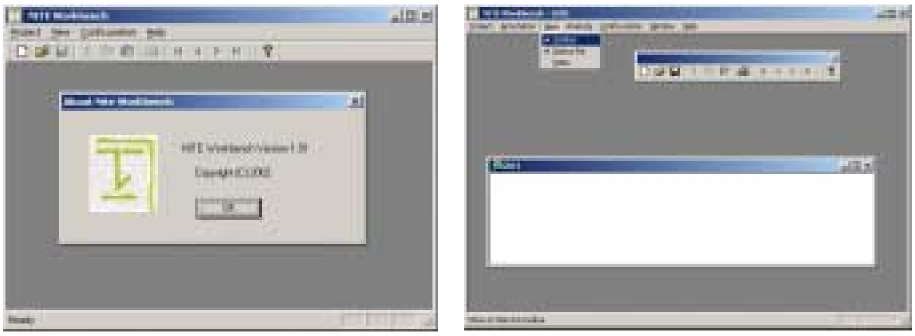


Figure 3. NITE Workbench main window with toolbar.

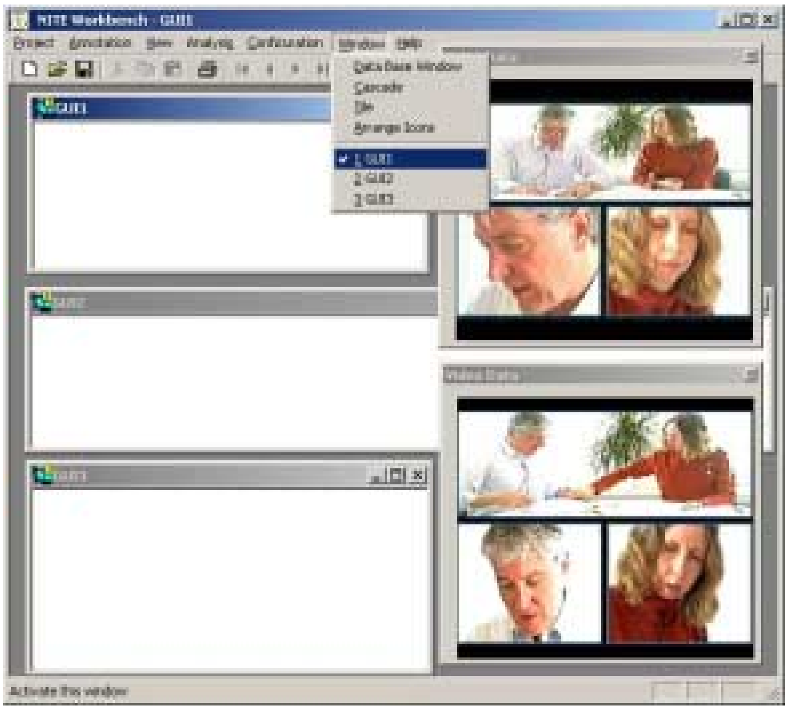


Figure 4. Raw data windows showing the NITE floor plan corpus with its three camera angles. It is possible to have multiple displays of the same raw data and annotate on different timelines.

Points 4 (resolution levels) and 5 (editors) in Section 6 are not illustrated in the figures in this section. Points 3 (one annotation at a time) and 10 (perceptible time-alignment) are illustrated below. Resolution levels can be set by allowing the user to zoom in and out of the present timeline resolution view as well as choosing among different types of visualisation, i.e. musical score format and ordinary running text format. An editor is an, initially empty, window which allows the user to enter and edit pure text and save it in a file. If feasible, it should also be possible to link between comments in the editor window and the corresponding time-aligned tags in the annotation file.

Based on the interface concept outlined above, the following two steps will be needed to accomplish an annotation:

1. *select* a class of phenomena to annotate by selecting a particular coding scheme (Figure 6). This will cause the list of tags belonging to the selected coding scheme to appear on the screen as a coding palette together with a panel in which to insert the annotations (Figure 5), cf. Section 6 points 2, 3 and 6;
2. *edit* (insert/delete) time markers on the time-line in the appropriate annotation panel. Markers will visualise the tags according to the chosen tag set on the annotation palette (Figure 7), cf. Section 6 points 6, 7, 8, 10 and 11.

The result may look as outlined in Figure 8. The described approach allows us to provide a uniform style of work with the annotation tool. For any level of annotation and any coding scheme, the user will perform the same set of actions: choose a class of phenomena (or a coding scheme), choose the appropriate button (the appropriate tag) from the coding palette, and insert the marker for the tag onto the time-line on the panel.

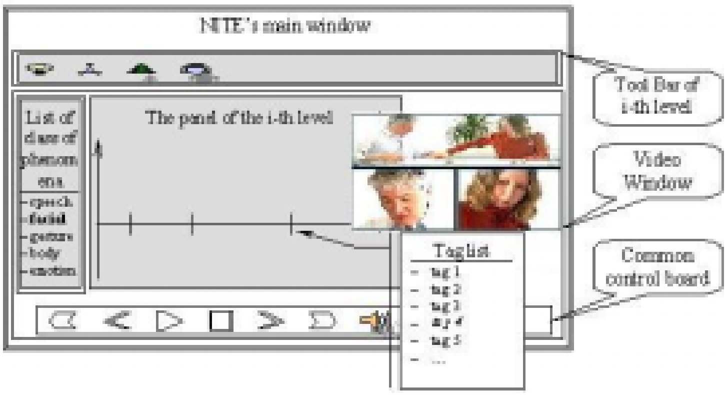


Figure 5. Raw data windows, control board, tag palette, and timeline (the horizontal line with inserted tags shown as small vertical lines).

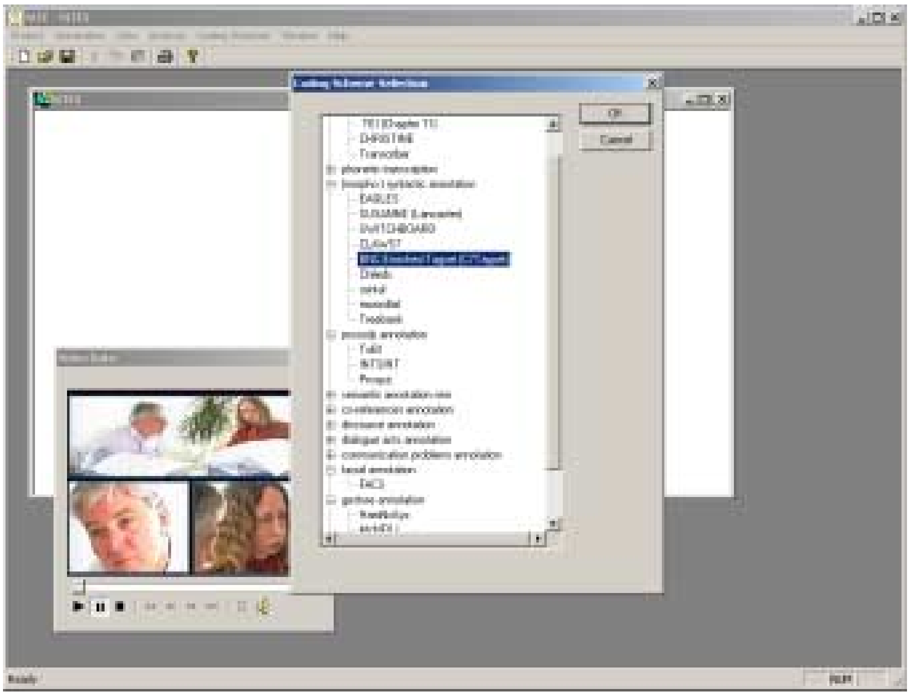


Figure 6. Selecting a coding scheme.

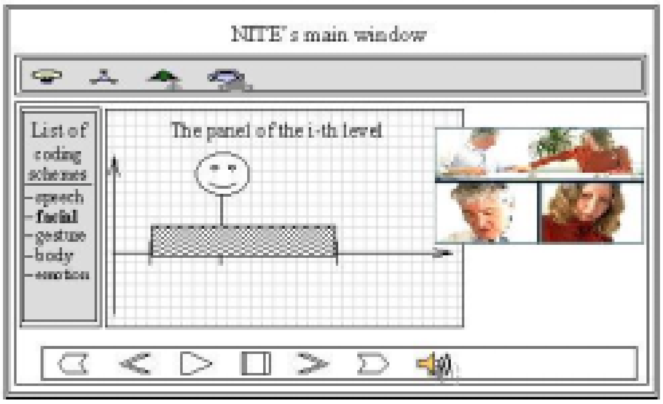


Figure 7. Tag visualisation shown on the timeline which runs from left to right.

7.3 Coding scheme interface

This section briefly describes part of the coding scheme interface. The purpose of this interface is to enable users to easily define, or enter into the NITE WorkBench, new coding schemes either by editing existing coding schemes or by adding new ones from scratch. Once a coding scheme has been entered, it can be selected for annotation use as shown in Figure 6.

We do not have a complete interface yet for defining new coding schemes. However, the underlying mechanisms have been implemented together with part of an early version of the user interface. Coding schemes are entered into, stored in, and retrieved from an Access database. What this means is that users can: generate their own specification of the tags they are going to use during annotation, including pictures, icons, shortcut keys, etc., so that they can create exactly the palette(s) they want to use during annotation; verify the correctness of their tag set entries by producing database reports; specify the tag set semantics for each tag in the database; etc.

The NITE WorkBench database already includes, among others, the HamNoSys sign language coding scheme and the FACS facial expression coding scheme [Knudsen et al. 2002b]. Each of these coding schemes include a tag set consisting of labelled images that visualise gestures and facial expressions, respectively. Figure 9 shows a FACS picture tag which has been entered into the database. Figure 10 shows one view of the database report which enables verification of correct tag entry.

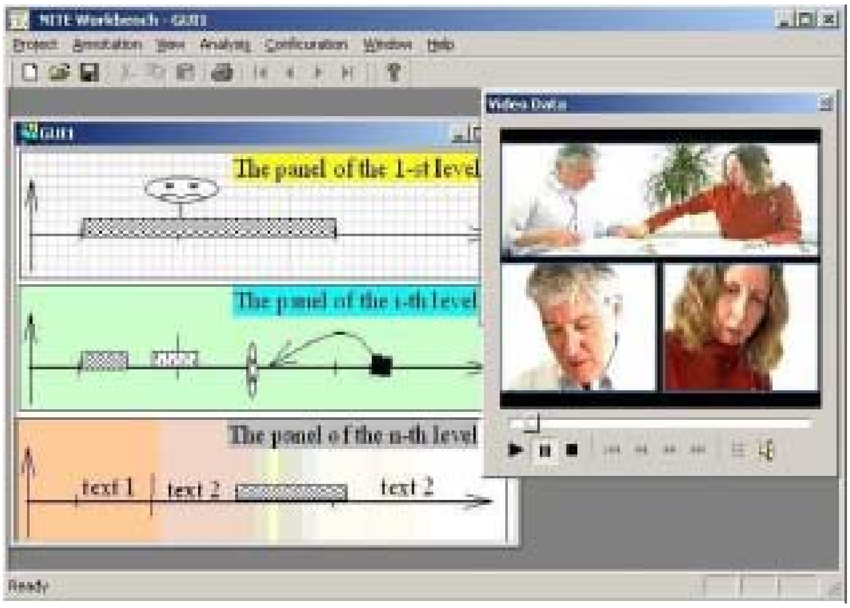
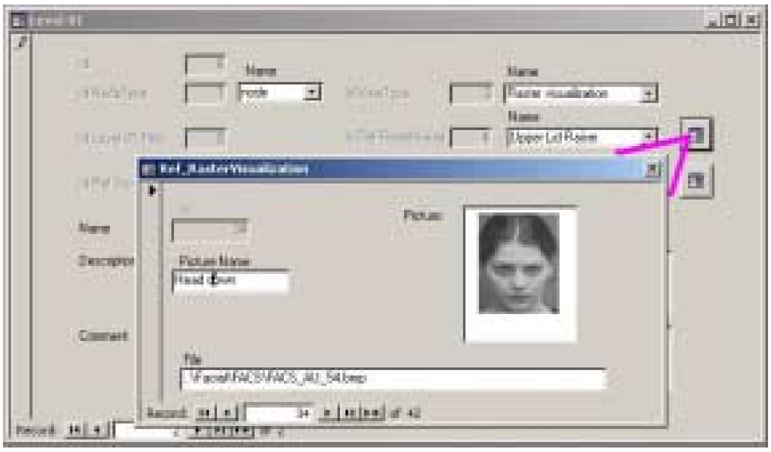
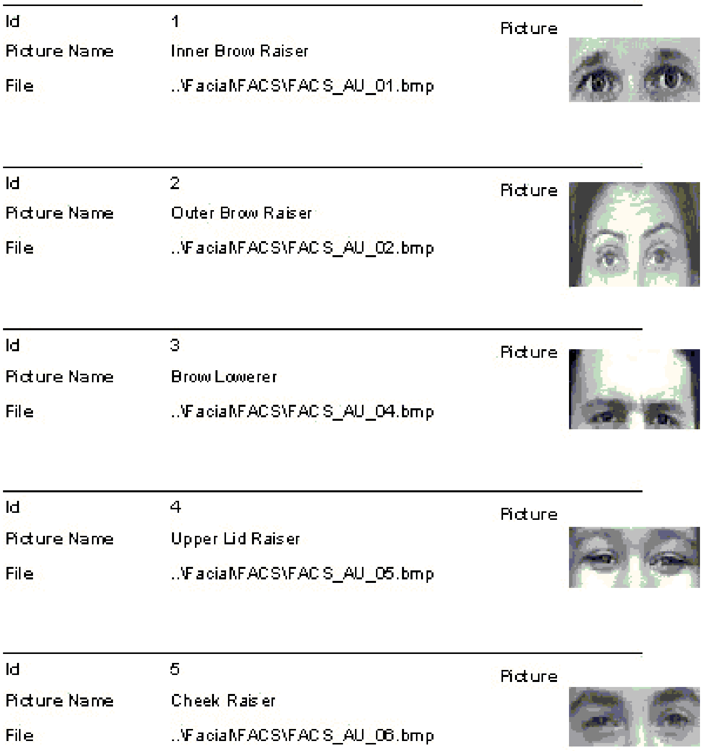

NITE Workbench - GUI1
Project Annotation View Analysis Configuration Window Help
GUI1
The panel of the 1-st level
The panel of the i-th level
The panel of the n-th level
text 1
text 2
text 2
Video Data
Ready

Figure 8. Tags visualised on the time line.



Name
node
Name
Raster visualization
Name
Upper Lid Raiser
Ref_RasterVisualization
Name
Description
Comment
Picture Name
Head down
Picture
File
\Facial\FACS\FACS_AU_54.bmp
Record: of 42

Figure 9. A FACS image tag.



Id	1	Picture	
Picture Name	Inner Brow Raiser		
File	..\Facial\FACS\FACS_AU_01.bmp		
Id	2	Picture	
Picture Name	Outer Brow Raiser		
File	..\Facial\FACS\FACS_AU_02.bmp		
Id	3	Picture	
Picture Name	Brow Lowerer		
File	..\Facial\FACS\FACS_AU_04.bmp		
Id	4	Picture	
Picture Name	Upper Lid Raiser		
File	..\Facial\FACS\FACS_AU_05.bmp		
Id	5	Picture	
Picture Name	Cheek Raiser		
File	..\Facial\FACS\FACS_AU_06.bmp		

Figure 10. Part of the database tag set report for the FACS coding scheme.

8. CONCLUSION AND FUTURE WORK

The NITE Workbench annotation user interface is currently being implemented in C++. We are in the process of enabling actual annotation as well as developing and implementing the user interfaces for adding new coding schemes and performing information extraction and analysis. Although first versions of the various panels and windows for annotation have been implemented, some of the underlying functionality is still missing and will have to be provided before a user can actually annotate a corpus using the tool. In parallel, work is going on towards specifying the NITE markup framework, including the NITE metadata representations, as well as identifying a core set of import/export facilities from and to tools of high relevance to NITE WorkBench users.

ACKNOWLEDGEMENTS

We gratefully acknowledge the support of the NITE project by the European Commission's Human Language Technologies (HLT) Programme. We would also like to thank all partners in NITE for their collaboration.

REFERENCES

Bernsen, N. O.: Natural human-human-system interaction. In Earnshaw, R., Guedj, R., van Dam, A., and Vince, J. (Eds.): Frontiers of Human-Centred Computing, On-Line Communities and Virtual Environments. Berlin: Springer Verlag 2001, Chapter 24, 347-363.

Cassell, J., Sullivan, J., Prevost, S. and Churchill, E. (Eds.): Embodied Conversational Agents. Cambridge, MA: MIT Press, 2000.

Dybkjær, L., Berman, S., Bernsen, N. O., Carletta, J., Heid, U. and Llisterri, J.: Requirements Specification for a Tool in Support of Annotation of Natural Interaction and Multimodal Data. ISLE Deliverable D11.2, 2001b.

Dybkjær, L., Berman, S., Kipp, M., Olsen, M. W., Pirrelli, V., Reithinger, N. and Soria, C.: Survey of Existing Tools, Standards and User Needs for Annotation of Natural Interaction and Multimodal Data. ISLE Deliverable D11.1, 2001a. View or download from isle.nis.sdu.dk

Dybkjær, L., Bernsen, N. O., Dybkjær, H., McKelvie, D. and Mengel, A.: The MATE Markup Framework. MATE Deliverable D1.2. Odense University, Denmark, 1998.

Isard, A., McKelvie, D., Cappelli, B., Dybkjær, L., Evert, S., Fitschen, A., Heid, U., Kipp, M., Klein, M., Mengel, A., Møller, M. B. and Reithinger, N.: Specification of workbench architecture. MATE Deliverable D3.1, August 1998. View or download from mate.nis.sdu.dk

Knudsen, M. W., Martin, J.-C., Dybkjær, L., Ayuso, M. J. M, N., Bernsen, N. O., Carletta, J., Kita, S., Heid, U., Llisterri, J., Pelachaud, C., Poggi, I., Reithinger, N., van ElsWijk, G. and Wittenburg, P.: Survey of Multimodal Annotation Schemes and Best Practice. ISLE Deliverable D9.1, 2002b. View or download from isle.nis.sdu.dk

Knudsen, M. W., Martin, J.-C., Dybkjær, L., Berman, S., Bernsen, N. O., Choukri, K., Heid, U., Mapelli, V., Pelachaud, C., Poggi, I., van ElsWijk, G. and Wittenburg, P.: Survey of NIMM Data Resources, Current and Future User Profiles, Markets and User Needs for NIMM Resources. ISLE Deliverable D8.1, 2002a. View or download from isle.nis.sdu.dk

Soria, C., Bernsen, N. O., Cadée, N., Carletta, J., Dybkjær, L., Evert, S., Heid, U., Isard, A., Kolodnytsky, M., Lauer, C., Lezius, W., Noldus, L. P. J. J., Pirrelli, V., Reithinger, N. and Vogele, A.: Advanced Tools for the Study of Natural Interactivity. Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02), 2002, 357-363.

Websites
ISLE: http://isle.nis.sdu.dk
MATE: http://mate.nis.sdu.dk
NITE: http://nite.nis.sdu.dk