Advanced Tools for the Study of Natural Interactivity

Claudia Soria[*], Niels Ole Bernsen[†], Niels Cadée[°], Jean Carletta[§], Laila Dybkjær[†], Stefan Evert[¥], Ulrich Heid[¥], Amy Isard[§], Mykola Kolodnytsky[†], Christoph Lauer[+], Wolfgang Lezius[¥], Lucas P.J.J. Noldus[°], Vito Pirrelli[*], Norbert Reithinger[+], Andreas Vögele[¥]

[*]Istituto di Linguistica Computazionale - CNR
Via Moruzzi 1, 56124 Pisa, Italy
{claudia.soria, vito.pirrelli}@ilc.cnr.it

[†]NISLab, Odense University
Science Park 10, DK-5230 Odense M, Denmark
{nob, laila, mykola}@nis.sdu.dk

[°]Noldus Information Technology b.v.
Costerweg 5, P.O. Box 268
6700 AG Wageningen, The Netherlands
{n.cadee, l.noldus}@noldus.nl

[§]University of Edinburgh, HCRC
2, Buccleuch Place
Edinburgh EH8 9LW, UK
{jeanc, amyi}@cogsci.ed.ac.uk

[∞]Institut für Maschinelle Sprachverarbeitung (IMS)
Azenbergstraße 12, D-70174 Stuttgart, Germany
{evert, heid, lezius, voegelas}@ims.uni-stuttgart.de

[+]Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany
{clauer, bert}@dfki.de

Abstract

The NITE European project aims at building an integrated best practice workbench for multi-level, cross-level and cross-modality annotation, retrieval and exploitation of multi-party natural interactive human-human and human-machine dialogue data. In this paper we intend to broach the general lines of software development envisaged in NITE, the four prototypes we intend to make available to the scientific community at large and our approach to usability evaluation of the prototypes. Under the aegis of LREC 2002 we plan to encourage conference participants to take active part in usability evaluation and provide early feedback to our software design choices.

1. Introduction

The main objective of the NITE European project (Natural Interactivity Tools Engineering http://nite.nis.sdu.dk) is to build an integrated best practice workbench for multi-level, cross-level and cross-modality annotation, retrieval and exploitation of multi-party natural interactive human-human and human-machine dialogue data. Under the aegis of LREC-2002, the NITE Consortium proposes to organize a demo session where conference participants are invited to test and evaluate four software prototypes for the analysis/annotation of multimodal interaction between two or more persons, or between persons and systems, through speech, gestures and facial expressions.

The present paper illustrates the software being developed by NITE to improve upon current support for working with corpora of recorded audio and visual data from human-human and human-system interaction. This includes the transcription, annotation, coding, and analysis of this data, where the interaction can be purely spoken or involve other communicative modalities, such as gesture. ISLE deliverable D11.2 (Dybkjær et al., 2001) argues that the main shortcomings of current software provision in this area fall in three areas: i) support for the process of adding annotation and structured coding to a corpus according to a defined scheme, where "structured" means that tags refer not just to timespans but can relate to each other; ii) support for the management of projects, especially the storage of metadata to encourage data reuse and information about the meaning and source of structured coding; and iii) support for the development and analysis of new forms of structured coding. For these software functions, users would of course like to have software that is stable, covers the complete range of their needs, is easy to use, and can be adapted to new tasks. The software being developed in the NITE project is designed to address exactly these requirements.

Section 2 discusses and delimits the scope of the planned developments in terms of the intended users and functionalities. It also points out the major considerations that have led us to a particular development strategy. Because there are a number of existing tools that go part of the way towards providing the required functionalities, starting from scratch on one tool to serve all purposes would be counter-productive. Section 3 describes the tools already available to the user community, focusing in particular on those already available within the NITE Consortium. These background tools are to be seen as a starting point for the development of more advanced NITE prototypes, which are described in Section 4. Each prototype is described in terms of its extra functionalities and compared with the functionalities provided by the background tools of Section 3. In our view of things, the lines of development of the four NITE prototypes have the potential of addressing the user needs identified in section 2. Finally, Section 5 describes the evaluation methodological framework under which we intend to gather input from prospective users.

2. Intended Users and Functionalities

Human interaction is of both academic interest and economic and social importance. We expect NITE technology to be of interest to industrial sectors that can benefit from understanding how humans interact with each other and with machines. The industrial sectors at which NITE technology is targeted are spoken dialogue systems and multimodal human-computer interfaces, animation, communication technologies, and language documentation, among the others. Finally, simply documenting different languages and different kinds of interaction, including their gestures, is a goal of some potential users.

In order to address the needs of such a wide community, a number of functions is required. Users must be able to create video and audio recordings of interactions; transcribe them, orthographically and phonetically; time-align the transcription to the signal; design and use multiple kinds of structured coding applicable on either transcriptions or other representations of speech such as waveforms; analyse reliability of the codings; apply automatic coding processes to the data, and hand-correct the results; display the data in different ways, as they explore the relationships contained within it; extract and index arbitrary parts of data; build statistical descriptions of data; and analyze them using inferential statistics.

Furthermore, users must be assisted in managing their data — e.g. by expressing metadata about the conditions under which recordings were made, about the transcription conventions adopted and, their level of granularity and reliability; by providing information about the codings available, together with an indication of who or what coded the recorded data and whar portion of recorded data was selected for coding. All this information is taken to be critical not only to ensure consistency and reliability of coding through a project lifespan, but also to promote coded data exchange and future reusability. The scientific community at large is becoming increasingly aware of the importance of metadata information. Although there already exist some tools that help users to manage this information semi-automatically (ISLE D11.2, Dybkjær et al., 2001), NITE will put considerable effort into the provision and parametrisation of some missing functions for data management.

3. What the User Community Already Has

The best way to characterize existing support is that the core functionalities for hand code transcriptions, coding display, indexing, and extracting data are not yet provided, while the peripheral functions of transcribing, performing basic or advanced statistical anlyses, automatically coding data are.

Transcription is supported via packages such as Transcriber[1]. Some users employ xwaves for this purpose, and others use standard word processing, sometimes aligning words with signal after the fact in xwaves. Descriptive and inferential statistics can be performed in, for instance, SPSS and Microsoft Excel, as long as it is possible to export the data to be analysed in some kind of simple tabular format. Tools for automatic data coding, such as part-of-speech tagging, already exist. It would be time-consuming to reimplement these functionalities. Instead, we intend to make it possible to link our software with existing tools by having them share a common data format: XML. Many tools, such as Transcriber, Microsoft Word, and some part-of-speech taggers, are already starting to employ XML as a native data format or to include export options. The filters for extracting data from XML into tabular formats are quite simple, as are those for uptranslating text-based transcriptions.

This leaves the core functionality of hand-coding, including project management, the coding of structured information which links existing codes in complex ways, configurable data display, indexing based on queries matching structural and temporal constraints on the data, and extracting subsets of the data based on the same sorts of queries. There are three existing tools that address this core functionality, but each fail to provide the full support needed. They are MATE, Anvil, and The Observer. The following sections describe the main features and shortcomings of each of them.

3.1. MATE

The MATE workbench[2] is a software tool for the display and annotation of XML encoded speech or text copora. The workbench allows a user to display and edit existing corpora, add new levels of annotation, perform queries over part or all of a corpus, and display or output the results. The format of the display and editing actions are set up using rule-based stylesheets based on a pre-standard version of the XSLT transformation language. The workbench provides a number of pre-defined stylesheets for use with particular annotation schemes, but its major strength lies in the fact that the stylesheet language is sufficiently high-level for writing stylesheets to be significantly easier than writing an editor from scratch. Any successful corpus project team is likely to have someone who understands the rudiments of XML, since otherwise they will have difficulty preparing their data for input into existing tools, analysing the data, and so on. In the MATE concept, this person has all of the necessary skills to specify tailored interfaces. Queries can also be performed to select a subset of the corpus for further processing within the workbench or for output to external tools. The workbench has in-built support for standoff annotation in which annotations are not all stored in one document but are linked by means of pointers. This allows the editing of one level of annotation without

[1] See http://www.etca.fr/CTA/gip/Projets/Transcriber/

[2] The MATE workbench is freely available and the code can be downloaded under a GNU public licence from http://mate.nis.sdu.dk/.

disturbing other levels and also makes it possible to have multiple annotation tiers with overlapping branches. The MATE workbench has been used successfully on several corpora for coding phenomena such as dialogue structure, tutoring strategy, and the use of metonymy, but there are a number of areas in which improvement is necessary.

The support for using the raw speech data while annotating is very limited — it is possible to play a section of speech, but it is only possible to display a waveform of the entire speech file, and the waveform display is not properly integrated with the rest of the workbench. There is also no support for spectrograms. The MATE workbench has no video capability at all, and would need to be extended to permit the annotation of video resources. As is common with the first implementation of a radically new idea, the workbench is under-documented and not as robust as would be required for widespread use. Neither the query language nor the display objects have been optimised, and it is necessary to have a detailed knowledge of the complex (and mostly undocumented) interactions between them in order to write an efficient stylesheet. The language in which graphical user interface actions are specified needs to be both simplified and extended, and it must be possible to load more XML material into the workbench at one time. The workbench is also too slow in practice for many common coding tasks.

3.2. Anvil

Anvil[3] is a research tool for the analysis of digitized audiovisual data (see Kipp, 2001). It allows the user to code human behavior and other visually accessible information in temporal alignment with speech and other auditory signals. Anvil was written as part of a PhD project on nonverbal communication at the University of the Saarland with support from the Deutsche Forschungsgemeinschaft (DFG) and builds on experiences with mass corpus annotation of dialogue acts within the speech-to-speech machine translation project *Verbmobil*. It has been actively used to encode video samples of a German TV show with nonverbal communication events, mostly gestures, and linguistic information.

In Anvil, for each type of behaviour (for instance, hand gesture or bodily posture) the user first defines an *annotation scheme* representing the range of behaviours that can occur. Then the user can use the software to divide the video into *behavioural units*, each represented by a code, for each type of behaviour. During coding and subsequent display of the data, codes are shown in layers, with one layer for each annotation scheme. These layers are displayed one below the other, running from left to right, just as a musical score (or “Partitur”, in German) shows instrumental parts in parallel. Behavioural units are depicted as boxes (rectangles) whose left and right borders correspond to their start and end points on a common time axis, the width thus being the duration of the element. Adding labels and colours to these bars allows intuitive comprehension of such a behavioural “Partitur” where temporal relationships between the layers, categories and durations can be captured at a glance. Anvil aims to make coding as intuitive and fast as possible, and to give the one most informative view of the resulting data. Although for the most part one can consider coding for the different types of behaviour to be independent even though they are temporally synchronized, it is possible to link behavioural units across the annotation layers into more complex structures. Anvil's control and data files are all in XML, the W3C standard markup language. This means that Anvil users can exploit the many tools that exist for the manipulation and transformation of XML files, especially when they wish to carry out data analysis.

Anvil is written completely in Java, making use of a recent extension called Java Media Framework (JMF); it works with the common video formats that are supported within JMF, such as AVI and QuickTime.

Anvil has some very attractive features, such as its Partitur-based visual representation, which make it an attractive choice of tool for work in natural interactivity. However, Anvil is also a very basic software package and needs further work before it will properly support users. Currently, for instance, it lacks on-line help, keyboard shortcuts for performing annotation, and the ability to add free structure annotations or “comments” to the data, even though all of these things are important during video coding. Some users may have multiple video tracks, but Anvil is limited to displaying one. Anvil is also currently intended to support only the video coding process and no related functions. Users must directly edit an XML file in order to configure the annotation schemes to be applied. It has no search, import, or export capabilities; instead, the user is expected to work directly with the XML which Anvil reads and produces. Although Anvil allows some linking across annotation layers, the basic expectation is of independence, with the output XML conforming to one rigid, non-tailorable structure.

Despite its being an attractive starting point for subsequent development, Anvil is not open source software and hence cannot directly be used as a starting point of development in the framework of the NITE project.

3.3. The Observer

The Observer (Noldus et al., 2000) is a professional tool for coding and analysis of video data (stored on tape or in digital media files) of any kind of behavioural process. It is a commercial package that works on Windows platforms (Windows 95, 98, NT, 2000 and XP). Installation is easy and straightforward. The user can design configurations for coding, and keep a library of configurations for re-use. However, the richly structured and inter-linked coding schemes typical for the NITE project are not supported in the current version.

Annotations can be coded from the keyboard in real-time after some practice. An overview window with all the codes in the configuration can be displayed during coding. The event log, which is a table with time running vertical, keeps track of the time and all different behavioural classes (or tracks). The event log is linked to the video time, which allows for quick searching and reviewing. It takes some time to learn to master these options. A horizontal timeline like in the ANVIL program might be a more intuitive display of multiple tracks. Text transcription is possible, but text is treated as free comments on time intervals. For NITE, more structure is needed for speech annotation, for example specifying

[3] Anvil is freely downloadable for research purposes from http://www.dfki.de/~kipp/anvil.

words within phrases, and phrases within sentences. It would be nice to be able to do video annotation, graphically marking up a specific part of the video image where a certain behaviour occurs.

The Observer data files contain the date and time of scoring, and every code is stored with a time stamp. The names of the people who did the coding can be stored as independent variables. There is a need for more elaborate project administration containing, for example, which MPEG files still need to be encoded, and which annotations have been checked for accuracy. XML import/export is not currently supported.

The Observer has several data analysis functions. Most of these require some training and practice before they can be used. Below we list each function with its basic functionality as well as needs for enhancement as identified by the NITE partners, and what is currently under development as a result.

- The time-event view shows either a table or a plot. The table gives a basic table-like overview of the annotation. The plot looks like the ANVIL horizontal timeline display format, with one line for each behavioural class (or track). However, there is no display of spoken text, sound waveforms or other signals, and it is not possible to create links between different tracks.
- Reliability analysis lets you compare coding of the same video data by two different people, by calculating the Kappa statistic and confusion matrices. However, you cannot compare more than two people's coding simultaneously.
- Elementary statistics provides frequencies and distributions for all codes. It is not possible to display graphical plots, but data can be exported to Excel or graphics programs.
- Lag sequential analysis can be used to analyze the frequency of transitions between behaviours, and the probabilities of those transitions. Transition sequences from up to nine behaviours before a specific behaviour, to nine behaviours after it, can be analysed.

For the NITE project, the main shortcoming of The Observer is that speech annotation and analysis have not yet been implemented. Support for complex coding schemes, with many cross-linked tracks that can be linked and grouped, is also needed. The most basic improvement to The Observer would be XML import/export functionality, which enables communication with text transcription programs and other linguistic annotation tools. Our developers are currently working on this. More NITE requirements are in our development plans for the next generation of The Observer program.

4. Nite Prototypes

4.1. The Observer

The Observer 4, the latest release, will be demonstrated. Compared to previous editions, The Observer 4 features improved usability, especially for design of the configuration or coding scheme. Data selection has been completely redesigned, and allows for the most complex filtering of annotation results. For example, one can define time intervals of variable length based on actual scored events, to answer questions like 'How often did Peter grin between the time when John entered the room, and the time when John left the room again?' Finally, The Observer 4 has an intuitive new layout that shows projects and their content in a tree view.

A digital video file of a group discussion will be used as an example throughout the demonstration. All parts of the program will be demonstrated:

- The concepts of configuration design will be explained, by showing the possibilities for structuring a coding scheme.
- The method of annotation is shown, with all the options for scoring, varying the video play speed and searching functions. In addition, it will be demonstrated how to build a video clip with highlights from the annotation data, based on annotated events.
- Examples of simple and more complicated queries of annotations will be given using the extensive options for data selection.
- Each of the analysis functions will be demonstrated with an example: time-event tables and plots, elementary statistics, reliability analysis and lag-sequential analysis.
- Finally, it will be shown how to export raw data and analysis results to graphics and statistics programs for further treatment and hypothesis testing.

During the demonstration, all ideas and comments on implementing language support in The Observer are most welcome. We are currently working on requirements for a new generation of software, with support for annotation and analysis of speech, among many other improvements and new features for other fields of research.

4.2. The Standard Display and Coding Interface

Although the MATE concept facilitates tailored interface design, it is inconvenient to have to write stylesheets before anything can be done with some data, especially since many end users do not personally have the technical skills required. Therefore, it has been decided to equip the NITE workbench with a familiar – albeit fairly complex – visual user interface which will enable ordinary users who are not skilled in any particular programming language to easily (i) add their own coding scheme, (ii) annotate a corpus using a coding scheme, and (iii) analyse and retrieve information from annotated corpora. So far, our development focus has been on facilitating annotation (ii). In the following we describe the visual annotation interface which we expect to demonstrate in late spring 2002.

The visual interface consists of the following five main components:

1. the *main window* which contains the main menu, the title, etc.;
2. the main window *toolbar* which contains the changeable (contents-sensitive) set of buttons;
3. a changeable amount of *panels* of the *i*-th class of phenomena to be annotated – 1 up to 10 panels;
4. the *raw data windows* displaying the different types of raw data – video, audio;
5. the common *control board* for controlling the active raw data window.

Figure 1, which includes the above five points, provides an idea of what the visual interface will look like.

In addition to the five main components, numerous palettes (dialogue boxes) each including several controls will be provided for the user to work with different coding schemes, inserting/deleteting tags, to visualising tags, etc.

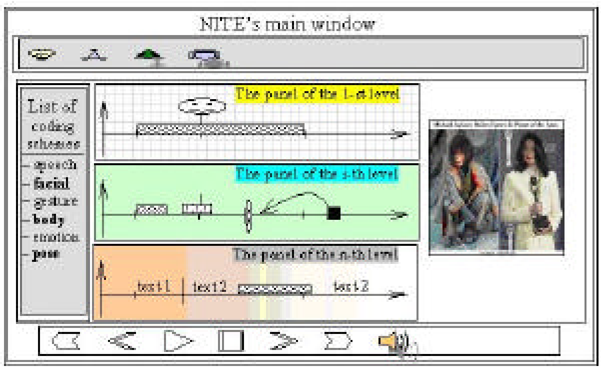

Figure 1. The NITE visual interface illustrated

To perform the actual annotation, the user will basically have to go through the following three steps:

1. *select* a class of phenomena to annotate using a particular coding scheme;
2. *edit* (insert/delete) a time marker on the time-line of the appropriate annotation panel, i.e. the one related to the selected coding scheme;
3. *visualise* the tags.

Markup of the time-line of the appropriate annotation panel is performed in a two-step process:

1. insert the marker of an appropriate tag onto the time-line;
2. visualise the tags, having chosen an adequate style of visualisation from the pre-defined set of options (Figure 1).

This approach allows a style of work with the annotation tool which is uniform in the following sense: for any level of annotation and any coding scheme, the user performs the same set of actions: choosing from the panel a class of phenomena or a coding scheme, choosing the appropriate button (the appropriate tag) from the coding palette, inserting the marker of the tag onto the time-line on the panel and, finally, choosing the style of graphical visualisation of the tags on the time-line.

The visual interface is being implemented in C++ and works on a Windows platform. More details on the visual interface can be found in (Bernsen, Dybkjær, and Kolodnytsky, 2002).

4.3. Adding External Functionality: Plug-ins for Signal Processing and Annotation

To annotate multi-modal data it is not sufficient to only add textual information. The annotator should also have access to analysis tools, and should be able to insert mark-up information directly in the multi-modal signal data. In NITE we envision the realization through plug-ins: the workbench has an interface to add external modules for e.g. signal analysis or markup directly in the video stream.
As prototypical test environment we defined a plug-in interface in Anvil and added two modules for spectral analysis and video markup.
Sonogram transforms time-domain based audio signals into the frequency domain using different methods like FFT or wavelet transformations. The most common audio and video file formats are supported. The two dimensional frequency presentations can be adjusted by changing the processing parameters (see Figure 2). It id also possible to show three-dimensional frequency plots. Signal plots can be stored as "Scalable Vector Graphics" (SVG) and bitmaps.

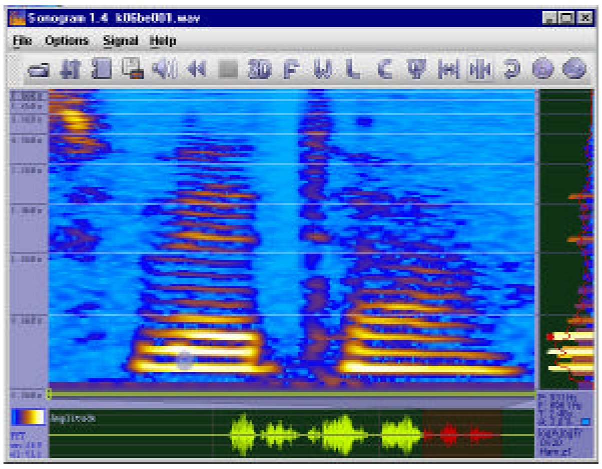

Figure 2: The Sonogram plug-in

The second plug-in currently realized in Anvil is used to insert mark-up information directly into the video data. The user can link each annotation tag with a tag in the video stream (see Figure 3). A gesture's main stroke is marked with a highlighted rectangle.

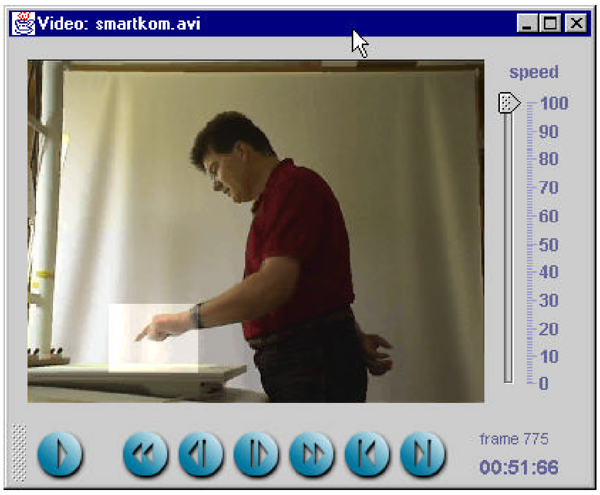

Figure 3: The video annotation plug-in

While in Anvil only one mark-up can be linked to the video stream, in NITE we will extend this functionality. The user can mark interesting events in the video, e.g. gesture movements, with tags for start and end points and interesting events in between. The annotation plug-in will compute one tag for the annotated movement so that a gesture tag in the textual annotation can be linked to the whole movement.

4.4. Generating Specialist Interfaces from Declarative Specifications

Our last demonstration shows a partial reworking of the MATE concept. Our goal is an engine that will create specialist user interfaces for specific corpora and specific tasks from a declarative specification of the interface's form and behaviour. Although MATE was successful in demonstrating this concept, it must be considered a prototype rather than an end-user system because it is slow and buggy, and the terms of the declarative specification relating to behaviour were given insufficient consideration. We are improving upon MATE in three ways. First, we are improving performance by employing what is now standard XML technology and considering possible efficiencies in the implementation. MATE had to define and implement its own processing techniques, and, as a radically new idea, had little time for optimization. Since then, XML standards have been developed for stylesheets and stand-off annotation, with professionally developed library implementations freely available. Second, we are adjusting our data model and the query language that accompanies it to make them clearer and better suited for multimodal data. Third, we are re-designing the declarative specification language used for defining an interface to make it more usable and less reliant on implementation details.

At this stage in the project, we will be demonstrating the use of standard XML technology to build a data display from a declarative specification of the display. As part of demonstrating this very early prototype, we will explain the underlying concept from the vantage points of the software designer, the interface designer, and the end user. Our end goal is a declarative specification format that admits sufficient flexibility for interface designers to define good interfaces quickly, and an engine that makes usable interfaces from these specifications. (Carletta, McKelvie, and Isard, 2002) argues that this flexibility is an important advance in functionality and gives a more complete description of the basic concept underlying the engine.

5. Evaluation Procedure

One of our aims in demonstrating the NITE prototypes is to gather feedback from prospective users about the tools' usability, performance, and concept. For this reason, we intend to pair each hands-on demo session with an evaluation session during which users will be asked to go through a questionnaire essentially aimed at evaluating the usability of the different prototypes. The framework in which we intend to perform evaluation is better known as usability testing, and will make use of the two well-known techniques of *assessment tests* and *cognitive walkthrough* (see Nielsen, 1994).

By means of assessment tests users are involved in a quantitative and qualitative examination of a partially working design, in order to determine areas of difficulty before they become hard to change.

So-called cognitive walkthrough is a more expert-oriented review, which makes use of "task scenarios" to guide evaluators in their analysis of an interface.

The feedback gathered will inform subsequent following development. We aim at involving small groups of users in parallel.

Usability will be measured along the major parameters sketchily illustrated in Table 1. Evaluators should note, however, that the demonstrated tools are still in a prototype version, and hence cannot be treated as final products. It is also worth emphasizing that the different parameters will have to be differently weighted according to the different tools being evaluated.

Learnability	How easy is it for new users to find features and do common tasks? Can the user get a job done the first time they sit down at a computer?
Efficiency	Does the new product reduce the time to complete a task? Does the product match their existing work flow or obstruct it?
Memorability	How often do users need to re-learn a feature? How often do they need to consult a manual for a feature they have used before? Are the operations of the product tangible to the user?
Errors	How often do users make mistakes?
Satisfaction	Did the user like the product? Was it a good first impression? Is the product adding or decreasing stress on users?
Productivity	Do they get more done before or after implementation of the product? Can their existing work flow be improved and augmented to increase how much a worker can do?
Training time	Does the product require training?
Data input speed and interpretation of data	Can the structure of the product increase how fast you can enter information? How fast can you understand what the computer is displaying for you? Are you missing important information?
Technical support needed	A technical support call means the product failed for a user in one of the above categories. How can this be reduced?
Maintenance costs	Is the product durable? Can it be made more stable? Can it be more simple? Can product updates be easier and faster?

Table 1: Usability parameters

It must be noted, indeed, that the four demonstrated tools require a slightly different evaluation methodology, depending on the different features and functionalities, but also on the underlying developmental goals. For instance, in at least three cases the NITE demonstrations concentrate on the form and behaviour of relatively static (but configurable) end user interfaces for data display, manipulation, and analysis. For these tools, the usability testing or "cognitive walk-through" is most appropriate. On the other hand, evaluating the demonstration version of the interface design engine is more difficult. The demonstration does not show all of the functionality that the user community needs at this point. Here, what should be evaluated is the basic concept. The demonstration will be a success if evaluators judge that the sort of flexibility that the engine accommodates is a useful advance on current functionality and that potential users will be able to configure the technology as they require using the building blocks that we are providing. Evaluators for this purpose must be somewhat more technically-minded than the end users required for the other demonstrations. In this case, thus, evaluation will take the form of in-depth group discussion with a small number of evaluators to pinpoint potential problems with the design; these interviews will inform downstream development. In a similar way, evaluation of the two modules for spectral analysis and video markup not only requires users' judgement about general usability, but also experts' comments about whether or not it actually complies with its underlying software concept as a plug-in component. For this reason, the task scenarios that will be used during the evaluation sessions will be differentiated according to the different prototypes to be tested.

6. Conclusions

The NITE project aims at building an integrated best practice workbench for multi-level, cross-level and cross-modality annotation, retrieval and exploitation of natural interactive behavioral data. The purpose of this demo is not only to describe to the user community the software design that is under development in the project, but also to gather early advice and input from the user community about the prototypes' usability and concept. Such an input will be helpful for user-centered development both during and after the project, and will likely highlight interesting areas of further improvement.

7. Acknowledgements

We gratefully acknowledge the support of the NITE project by theEuropean Commission's Human Language Technologies (HLT) Programme.

8. References

Bernsen, N. O., Dybkjær, L. and M. Kolodnytsky (2002). The NITE Workbench - A Tool for Annotation of Natural Interactivity and Multimodal Data. To appear in Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas, May 2002.

Carletta, J., McKelvie D. and A. Isard (2002, to appear). Supporting linguistic annotation using XML and stylesheets. In G. Sampson and D. McCarthy (Eds.), *Readings in Corpus Linguistics*. London and NY: Continuum International.

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

Kipp, M. (2001). Anvil - A Generic Annotation Tool for Multimodal Dialogue. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, pp. 1367-1370.

Nielsen, J. (1994). *Usability Engineering*. Morgan Kauffmann.

Noldus, L.P.J.J., Trienes, R.J.H., Hendriksen, A.H.M., Jansen H., and R.G. Jansen (2000). The Observer Video-Pro: new software for the collection, management, and presentation of time-structured data from videotapes and digital media files. *Behavior Research Methods, Instruments & Computers*, 32:197-206.