| | |
|---|---|
| **Project ref. no.** | IST-1999-10647 |
| **Project title** | ISLE Natural Interactivity and Multimodality Working Group |

| | |
|---|---|
| **Deliverable status** | Public |
| **Contractual date of delivery** | 31 October 2000 |
| **Actual date of delivery** | July 2001 |
| **Deliverable number** | D11.2 |
| **Deliverable title** | Requirements Specification for a Tool in Support of Annotation of Natural Interaction and Multimodal Data. |
| **Type** | Report |
| **Status & version** | Final |
| **Number of pages** | 30 |
| **WP contributing to the deliverable** | WP11 |
| **WP / Task responsible** | Laila Dybkjær, NISLab |
| **Author(s)** | Laila Dybkjær, Stephen Berman, Niels Ole Bernsen, Jean Carletta, Ulrich Heid and Joaquim Llisterri |
| **EC Project Officer** | Brian Macklin |
| **Keywords** | Tools specification, functionality, interface, architecture and platform, natural interactivity and multimodality data, annotation support |
| **Abstract (for dissemination)** | This report discusses overall functionality, interface, architecture and platform requirements to a toolset in support of transcription, annotation, information extraction and analysis of natural and multimodal interaction (NIMM) data. NIMM data are corpora of recorded audio and/or visual data from natural and/or multimodal human-human and/or human-system communicative interaction. The report builds on input from ISLE deliverable D11.1 (Dybkjær et al. 2001), the authors' joint experience from developing the MATE workbench (http://mate.nis.sdu.dk), and ongoing work in ISLE WP8 and WP9 on natural and multimodal interaction data resources and coding schemes, respectively. |


# ISLE Natural Interactivity and Multimodality Working Group Deliverable D11.2

# Requirements Specification for a Tool in Support of Annotation of Natural Interaction and Multimodal Data

**July 2001**

## Authors

Laila Dybkjær[1], Stephen Berman[2], Niels Ole Bernsen[1],
Jean Carletta [3], Ulrich Heid[2], Joaquim LListerri[4]

1: NISLab, Odense University, Denmark. 2: IMS, Stuttgart University, Germany.
3: HCRC, Edinburgh, UK. 4: DFE, Barcelona, Spain.

# Section responsibilities

| | | |
|---|---|---|
| Section 1: | Introduction | Niels Ole Bernsen and Laila Dybkjær |
| Section 2: | Functionality | Jean Carletta |
| Section 2.1.2: | Transcription of prosody | Joaquim Llisterri, María Machuca, and Mònica Estruch |
| Section 2.3: | Information query and retrieval | Stephen Berman and Ulrich Heid |
| Section 3: | Interface | Laila Dybkjær |
| Section 4: | Ramifications of requirements | Jean Carletta |
| Section 5: | Final discussion | Laila Dybkjær |

# Contents

# 1 Introduction

This report discusses overall functionality, interface, architecture and platform requirements to a toolset in support of transcription, annotation, information extraction and analysis of natural and multimodal interaction (NIMM) data. NIMM data are corpora of recorded audio and/or visual data from natural and/or multimodal human-human and/or human-system communicative interaction. The report builds on input from ISLE deliverable D11.1 (Dybkjær et al. 2001), the authors' joint experience from developing the MATE workbench (http://mate.nis.sdu.dk), and ongoing work in ISLE WP8 and WP9 on natural and multimodal interaction data resources and coding schemes, respectively. The report will serve as input to the NITE project (http://nite.nis.sdu.dk) which started in April 2001.

This chapter discusses and delimits the scope of the toolset in terms of general user needs (Section 1.1), target users (Section 1.2), and domain and task coverage (Section 1.3). The delimited scope provides the basis for identifying existing tools as point of departure or input providers for the toolset to be specified (Section 1.4). The subsequent chapters provide an overall specification of a toolset based on the tool scope definition of Chapter 1. Chapter 2 specifies the required functionality. Chapter 3 specifies requirements to the interface. Chapter 4 addresses architecture and platform requirements. Chapter 5 concludes the report.

## 1.1 General user needs

The "users" addressed in this section include *standard users*, i.e. those who want to use the tool for NIMM data annotation and data handling, as well as *tool users*, i.e. those who are capable of modifying the tool and adding functionality to it (see also Section 1.2).

Throughout the report we make an overall distinction between *raw data* (graphical and/or acoustic) and (textual, symbolic) *annotated data*. Annotated data include transcriptions, annotated transcriptions (the transcribed data bundled together with or linked to their annotations), and annotated raw data (the annotations being text, symbols, etc. and the data being the raw data referred (or linked) to. There is no such thing as a "transcription" of, e.g., video images but it is common practice to write annotations which refer to the raw data images. In fact, this same generic kind of annotation (i.e. annotation of raw data) may well be done to the spoken part of a dialogue (i.e. without an intermediary transcription). See also below.

ISLE deliverable D11.1 (Dybkjær et al. 2001) reviews a range of existing and planned tools in support of NIMM data annotation and data handling, compares the tools, and draws conclusions on user needs and standardisation efforts. Given the fact that the reviewed tools can safely be viewed as expressing at least some of the actual user needs, it appears that a general tool in support of NIMM data transcription, annotation, information extraction and analysis should satisfy at least the following requirements.

1. A flexible and open architecture which allows for easy addition of new tool components (a modular workbench). A general tool produced by a research project or a company 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. This requirement is mainly of importance to the third user group mentioned in Section 1.2 below.
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 need not mean anything for end-users but it means a lot for developers who want to change either the interface or the internal representation.
3. Transcription support and annotation support at different levels of abstraction, 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 transcription and annotation

at different levels of abstraction, 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. The MATE project has 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 several different levels. However, it remains to be seen to what extent the framework will work for other areas of NIMM data coding.

4. Already 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 NIMM annotation processes, this should be supported by the toolset. Similarly, to the extent that it is possible to (semi-) automate NIMM 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 data analysis, and (ii) via the use, as far as possible, of standard(ised) data formats allowing easy importing and exporting of (automatically created) annotations.
5. Powerful functionality for query, retrieval and extraction of data from annotated corpora; tools for data analysis, possibly including 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/searching raw data according to time information. It should be considered whether more advanced raw data query and extraction functionality could be included in the toolset.
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 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 (e.g., presenting annotations based on new coding schemes).
10. Possibility of importing and thus reusing existing data resources via conversion tools. Today, there is wide-spread use of proprietary systems and formats for NIMM data coding. This constitutes a major obstacle to creating a standard NIMM 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 system specified here. 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 toolset under specification, 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 toolset under specification, it may be considered to create a conversion tool for exporting the resources

instead of creating a statistical analysis tool for those resources. It would be futile to try to replace existing work practice for all of the functions, first because that spreads coding effort very thin indeed, second because it is hard to predict what the complete set of needs really will be, and finally because people often like the tools they know and will resist the whole set of tools if they believe that they are meant to replace those which they know already.

The general requirements stated above are insufficient for fixing the scope of the toolset. Before generating a more detailed requirements specification, we need to consider who the target users are, what is meant by NIMM data annotation support, and which domain and task coverage the tool should have.

## 1.2 Target users

The NIMM data coding toolset specified in this report is intended to be general-purpose. Thus it can be expected that the tool will be of potential interest to people with different professional backgrounds working with annotation in many different areas, from several different perspectives, and with an unlimited number of different annotation purposes.

It is not within the scope of ISLE NIMM to collect input on the needs and desiderata of user groups collectively representing all or most areas for which the toolset may be of interest. Rather, we will consider the needs expressed through the tools reviewed in D11.1 and the user groups and areas for which these tools are intended. We will, moreover, draw on input from the user groups and areas represented by the ISLE NIMM participants themselves. The ISLE NIMM participants work in many different fields, including spoken and written dialogue systems development and evaluation, development and evaluation of multimodal systems, embodied agent design, face and gesture annotation, and research in prosody, linguistics, psychology, anthropology, human behaviour and human factors.

It is the authors' experience that, basically, users of annotation tools may be divided into three groups as described below. Needs for, and consequences of, supporting each and all of these groups should be thoroughly considered and the choices made should be reflected in the requirements specifications.

*The first user group* includes people who need a tool which allows them to do their tasks of annotation, extraction, and analysis without bothering about the internal design, representations and workings of the tool. Typically, these users are experts in their area, such as the understanding of speech and facial expression integration in human communication, and they regard the annotation tool simply as a vehicle for making their work more efficient and its results more useful and more widely available.

*The second user group* includes users who have become so used to data coding formalisms, such as SGML or XML, or who experience that existing editors are not good enough so that they feel most comfortable if they can edit the coded data directly. Thus, if XML is used for internal data representation they want to have access to the XML files, possibly via an editor. The internal data representation is meant for the computer rather than for the user, and it should be considered whether direct access to the internal data representation should be supported and what the possible consequences are.

*The third user group* includes users who have the skills and the motivation to add new functionalities to an existing tool provided that the tool makes this possible. Since no NIMM data coding toolset will ever include all conceivable useful functionality, the best option is to equip a general-purpose NIMM data coding toolset with an open architecture which accommodates the addition of new functionality.

## 1.3 Domain and task coverage

The toolset under specification should support natural interactivity and multimodal (NIMM) data transcription, annotation, and information extraction and analysis. Given its general-purpose nature, the toolset should be useful for carrying out these tasks for purposes of research, development and education.

Before proceeding to discuss the domain and task coverage of the toolset in more detail, some definitions are needed in order to avoid the conceptual and terminological confusion which still reigns in the emerging field of natural and multimodal interaction. A *modality* is a particular way of presenting information in some medium. A *medium* is a physical substrate for information presentation, such as light/graphics which is being perceived visually, or sound/acoustics which is being perceived auditively. Obviously, there are many different ways of representing information in a particular medium. This is why we need to distinguish between different modalities presented in the same medium. For instance, spoken language is a modality expressed in the acoustic medium, whereas written language and facial expression are modalities expressed in the medium of light/graphics. In the form of lip movements or textual transcription, spoken language may also be expressed in the medium of light/graphics. Thus, the same modality can be represented (more or less adequately) in different media. During human-human-system interaction (see below), a modality may be used as an *input* modality (from a human to the system or to other humans) or as an *output* modality (from the system to humans or, rarely today, to another system), or both. *Multimodal* representations are representations which can be decomposed into two or more *unimodal* modalities. For more details on these basic concepts in Modality Theory, see (Bernsen 2001). *Interaction* refers to communication or information exchange between humans, possibly mediated by a computer system, or between humans and computer systems. The term *natural* qualifies the interaction and refers to the ways in which humans normally exchange information with one another.

The rationale behind the choice of modalities which the toolset should be able to handle, is the following. The modalities of information representation which the toolset must be able to handle include multi-party speech, text, facial expression, gesture and bodily posture. Together, these modalities cover most of the modalities used by humans during communication. To be sure, humans also sometimes touch during communication, the touch (at least, sometimes) having some communicative meaning. However, there are no plans to include the medium of touch/haptics in the toolset, except for touch as visually represented. For the purpose of the tool, we view touch-as-visually-represented as an extension of gesture. Visually represented touch, thus conceived, includes important acts of communication, such as the manipulation of artefacts, some of which are themselves carriers of information, such as text, maps or images. In future, we can expect an increase in the number of input/output modalities of information representation which will be used for exchanging information with computer and telecom systems, enabling more natural interaction than has been possible so far. Similarly, we can expect continuous development towards ultimate natural interactivity in which the system is capable of understanding and generating information on-line in the same ways humans do, i.e. through speech, facial expression, gesture, bodily posture and artefact manipulation. Such interactive systems will increasingly be designed to allow multi-user, as opposed to single-user, interaction, and systems will increasingly be designed to not only enable (multi-)user interaction with the system itself but also human-human interaction through, or mediated by, the system. In conclusion, the NIMM coding toolset under specification should be able to support core future developments towards ultimate natural human-human-system interaction.

In addition to the above clarifications of concepts and terminology, and of the intended scope of the toolset in terms of modalities and media, an important claim underlying the specification of the tool is the following. Among the different input/output modalities involved in natural interaction, some are more fundamental than others in an important sense. These modalities include speech, full sign languages for the hard-of-hearing and the deaf, and touch languages for the blind-and-deaf. These languages are all used in *situated communication*, possibly mediated by computer systems, and they are all *maximally expressive* modalities. Something which comes close to these languages, is the language of lip reading. Facial expression alone, including non-professional lip reading, as well as (non-full-sign-language) gesture alone, gaze alone, or bodily posture alone, on the other hand, are *communicative accompaniments* to the basic languages of situated communication. They are not in themselves as expressive as are the basic languages of situated communication but they do add important information in situated communication.

The claim, now, is that it follows that the toolset must have an important component which represents the *textual transcription* of what is being communicated in those basic languages of situated communication. The component may not be used by all users for all annotation purposes, hence it

should be possible to annotate, e.g., raw video data with respect to gesture-only, eye-brow movement-only, or facial emotion-only. However, these annotations are not transcriptions of the basic communication but interpretations of accompanying communicative behaviour. Thus, if the annotation aims to embed an interpretation of communicative accompaniment modalities in the context of situated communication, the annotation must link the accompanying facial expressions, gestures or bodily postures to the core transcription of the basic language of situated communication used. In an extended sense, the same notion of communicative accompaniment modalities applies to the physical/environmental *setting* in which the communication takes place. Thus, the toolset under specification should support, at least, textual transcription of spoken language because it provides the basic context for the interpretation of accompanying communicative modalities. Therefore, the toolset must impose a default order of representation of situated communication consisting of (i) raw data linked to (ii) textual transcription of the language of situated communication used linked to (iii) cross-level and cross-modality annotations. Other more specialised transcriptions, such as sign language, may be addressed in separate projects and linked to the toolset we are considering.

Beyond what has already been said in Section 1.1, it seems hard to derive a multitude of specification requirements from the intended goals of the tool, i.e. to support tasks of transcription, annotation, and information extraction and analysis for purposes of research, development and education. However, the research and development support goals clearly imply that the toolset should be able to handle reasonably large corpora without problems, e.g. as regards speed and robustness. The educational support goal implies ease-of-use because many of the students learning data annotation can be expected to belong to the first user group described in Section 1.2. The same applies to many researchers who may want to use the tool. The educational support goal also implies the usefulness of furbishing the launch version of the tool with a number of illustrative best practice coding schemes and useful data visualisations. In addition, the research support goal, in particular, implies the need for powerful data query and analysis mechanisms, transcription support and annotation support at different levels of abstraction, for different modalities, and for annotating interaction across levels and modalities, and easy addition of components and visualisations. The research and development support goals imply the need for adding conversion tools and new tool components.

## 1.4 Tool platform

Based on the analysis of existing tools, standards considerations and user needs in (Dybkjær et al. 2001), we have selected a tool basis which will form our point of departure for the specification of a (new/enhanced) toolset for annotation of natural interaction and multimodal data in the following chapters. Of the tools reviewed in (Dybkjær et al. 2001), the MATE workbench is by far the most advanced as regards markup of spoken dialogue. It can in principle handle any annotation level and comes with a number of example coding schemes for different annotation levels, such as dialogue acts and coreference. The other reviewed tools for handling speech data either do not go beyond the transcription level or enable annotation at a single, higher level of abstraction, such as dialogue acts, according to a built-in annotation scheme.

We believe that the MATE workbench would form a good point of departure for a NIMM toolset. It has various shortcomings as mentioned in the following chapters. Some of these have been worked on in the ISLE project, cf. Chapter 5, to bring the MATE workbench into a better position to serve as a point of departure for a NIMM toolset.

Among the NIMM annotation tools reviewed in (Dybkjær et al. 2001) the Noldus Observer and the Anvil tool are considered the most interesting tools to study in detail to get further information on NIMM functionality requirements. This is currently being done in-depth in the NITE project (http://nite.nis.sdu.dk). Also Eudico is considered an interesting tool. However, so far it has not arrived at a stage where it has been possible to use it for actual annotation. Input for this report concerning functionality and interface requirements derived from the mentioned tools is based on the survey report (Dybkjær et al. 2001) and not on the in-depth study of each tool which is being undertaken in the NITE project. This report is intended to form the overall background for the more detailed specification in NITE.

The following three chapters discuss relevant toolset functionality, interface, and architecture and platform issues.

# 2 Functionality

In this section, we describe the functionality required by NIMM data annotators. This is not to suggest, however, that a single tool should provide all of the functionality required. There are already a number of tools around in various states of progress or completion, each of which cater for some of the needs of NIMM data annotators. One should avoid duplicating the effort already put into the creation and maintenance of useful NIMM data annotation tools. Moreover, trying to address all user needs in the field would run a significant risk of never producing the kind of tool which the field strongly needs. In addition, users tend to favour tools which they already know and trust. Rather, developers of a new tool should (i) focus on those user needs which are currently least well served and (ii) make sure that the new tool can do the tasks it is required to do really well.

Below, after presenting the functionality required by NIMM data annotators, we summarise the tool functionality which is already available as well as the gaps which still exist.

## 2.1 Transcription

There are two common annotations of spoken audio signals which are so basic that they go by the name of transcription: orthographic transcription and prosodic transcription. These together represent the core transcription needs for NIMM users. In both cases, transcription gives a basic representation of the signal data upon which further description can be built. Although it is also possible to have transcription from video (for instance, of sign languages), the way to support this process in NIMM tools is much more akin to support for the video annotation process than to more conventional audio transcription. For this reason, transcription of sign language is considered just to be one case in section 2.2.

### 2.1.1 Orthographic transcription of spoken language

Some suggestions for how to represent the orthography of spoken language have been developed by e.g. EAGLES and the TEI (Sperberg-McQueen and Burnard 1994). These coding schemes may be suitable for the majority of NIMM data users or those starting out their exploration of a data set. Thus, the functionality required might be seen as that needed to produce compliant transcription. This includes not just the ability to key in words and to insert codes (such as those indicating a cough or a laugh), but also the ability to play and replay specified parts of the audio signal until the transcription is correct (possibly at a different speed from the original) with as little disruption to the speed of transcription as possible. In cases of poor signal quality or multiple speakers, it may also be necessary to review video signals intensively in this way.

In addition, there are NIMM data resource providers who have valid reasons for producing non-compliant transcriptions. Transcriptions made by conversational analysts, for instance, rely on a set of more fine-grained distinctions than the ones usually provided in data destined for use in computational linguistics. Transcription support should include support for the definition of new transcription conventions and formats, and the use of these conventions, as well as the use of existing conventions.

For some data resources, depending on the intended primary application for the data, spell checking can be an issue. For instance, if the data is intended to be used for training a speech recogniser, it can be useful to normalise spellings in a way which may be harmful for research into dialectal variation. NIMM data users should have integrated access to spell-checking during the transcription process, either through spell checking as transcription is performed or via an extra pass in the transcription process. Note that whatever spell checking process is used, it must be possible to supply special-purpose lexicons to suit the particular needs of individual data resource providers, including special-purpose codes reflecting the events particular to the data collection context.

In addition to tools supporting transcription itself, it should be possible to attach information to a data resource about the transcription processes and conventions used, both as a form of project documentation and as an aid for others in determining whether or not the data is re-usable for their purpose (metadata).

For certain kinds of NIMM resources a textual version of the spoken utterances may be available electronically. The NIMM tool should then allow interaction with an external text/speed alignment tool via export and (re-)import. Such aligners exist, e.g. for English and German.

### 2.1.2 Transcription of prosody

The amount of information included in a prosodic transcription as well as the reliability of this information will determine the overall quality of the transcription. For this reason, it is essential to make correct decisions on the linguistic features that should be represented. Similarly, the analysis of the data will be highly influenced by the adoption of a specific coding scheme.

This sub-section reviews the linguistic phenomena that could be annotated in a prosodic transcription of spoken language. Earlier reviews of prosodic transcription and annotation schemes include (Grønnum Thorsen 1987, Gibbon 1990, Llisterri 1994, Quazza and Garrido 1998). The present proposal, taking into account previous work in this area, is based on general principles explained below.

First of all, a prosodic transcription system should, in principle, express all and only the linguistically relevant suprasegmental phenomena. It should be independent of the transcribers and should also be language independent, since cross-language comparisons have to be possible using the same system (Williams 1996). Moreover, proposed labels should serve not only for all languages, but also for all types of spoken language, from planned discourse to spontaneous speech.

Along these lines a list of prosodic units is proposed so that researchers can choose according to the aims of the transcription, the purpose of the research, the speaking style considered and the specificity of the corpus. The list includes *segment*, i.e. each of the vowels or consonants that compose a syllable, *syllable*, i.e. the unit in which stress is placed (Quilis 1981, 1993, among others) and the domain for the pitch level, *stress group* (or "tonic group"), i.e. the segment of an utterance with a stress and with a coherent meaning that cannot be divided into smaller units (Navarro 1944; Fant 1984) (other definitions exist as well, cf. (López Gonzalo 1993, Thorsen 1978), *intonation group* (also called "melodic unit", "sense-group", "breath-group", "tone-group", "tone-unit", "phonological clause" or "intonational phrase"), i.e. the shortest possible segment with meaning and with a specific melodic contour, cf. (Navarro 1944), *phonic group* (also called "breath group"), i.e. the part of an utterance comprised of the interval between two empty – i.e. silent – pauses, cf. (Navarro 1944), *sentence*, i.e. the highest unit of prosodic analysis and the domain of intonational phenomena, and *paragraph* which is by certain authors also considered the domain of intonational phenomena. Although some units may seem quite similar, such as the "breath group" and the "intonation group", they may be adequate for different purposes; for example, the "intonation group" as defined here is far more adequate to the transcription of dialogues than the "breath group" which implies the presence of an initial and a final empty pause at the boundaries of the unit.

We distinguish between prosodic units, prosodic phenomena, phonetic correlates of prosodic phenomena, and linguistic phenomena. Prosodic units are defined and presented hierarchically, from minimal units – segments – to larger ones – paragraphs. The prosodic phenomena to be annotated have several acoustic-phonetic correlates. Those correlates can be observed in the speech signal – either in a time-domain representation, such as the speech waveform, or in a time-frequency representation, such as the fundamental frequency contour – and are the equivalents of acoustic cues – e.g. bursts, transitions, formant frequencies, duration, etc. - for segmental elements. They are used to define prosodic units and phenomena to be annotated. Prosodic phenomena include (i) intonation with *pitch accent* represented either by a single or double tone, *boundary tones* which are found at phrase boundaries, *phrase accents* which are used to show the shape of the melodic contour from the last (nuclear) accent to the phrase boundary, *downstep* which is a phonological effect controlled by the speaker and consisting of a progressive lowering of the H tones, always referring to the preceding one, and *upstep* which readjusts a tone whose value is 0 to have the none-zero value of the preceding tone, and (ii) stress (Abercrombie 1967, Crystal 1969, Lehiste and Peterson 1961) with *lexical stress* which is the stress placed on a syllable of a word, *primary stress* which is the most prominent word stress, *secondary stress* which is a less prominent stress that marks a specific rhythm in the utterance,

*sentence stress* which is the most prominent stress in a sentence and gives the rhythm to a language, and *emphatic stress* which is used to emphasise a specific syllable or word in a sentence.

Prosodic units and phonetic correlates of prosodic phenomena should be considered as two levels in a transcription. In this way we can observe the relation between these levels.

As far as linguistic phenomena are concerned, only those which are related to prosodic phenomena have been considered. A typical example would be vocals (i.e. vocalized but not necessarily lexical phenomena (Sperberg-McQueen and Burnard 1994)) meaning an assertion or a question depending on the direction of the pitch movement (Leech et al. 1995). Generally speaking, the behaviour of these linguistic phenomena distinguishes among different speaking styles. Linguistic phenomena should be considered in a transcription as a further level. Linguistic phenomena to be annotated in spoken language (Edwards 1993, Sperberg-McQueen and Burnard 1994, Llisterri 1996, Gibbon et al. 1998, Leech et al. 1995) include *turns* where F0 movements, lengthening of the last part of the discourse or the presence of filled pauses are indicators of a turn change or maintenance, *focus* which can be determined by the word order within a fragment or can be realised by means of duration, loudness and F0 changes, *hesitation* which is an unnatural interruption of the speech flow without any following correction of the lexical material and which can be related to pauses (silent or filled) or to a lengthening of the last part of a word, *topic* (new/old) concerns the relationship between sentence accent and the distribution of given/new – or old/new – information (Terken and Nooteboom 1987, van Donselaar 1991), *intensionality*, i.e. the behaviour of the speaker in a given situation, *modality* which is related to the linguistic function of a sentence - a question, assertion, exclamation, etc. – and has its main prosodic correlate in F0 movements, *speech acts*, i.e. the analysis of a discourse in terms of initiatives (actions) and responses (reactions), *overlaps* which appear when two or more speakers communicate at the same time, and *vocals* which are semi-lexical (e.g. “er”, “ah”, “mm”) or non-lexical (e.g. cough, sob, laugh) items which fulfil a communicative function.

According to our proposal, the annotation of prosody should have three levels where the transcriber can take into account prosodic units, phonetic correlates and linguistic phenomena. The importance of these levels is the same, so there is no hierarchy between them. Furthermore, the transcriber can choose the levels he wants to transcribe if he is not interested in all three of them.

## 2.2 Annotation

The simplest kind of annotation which a NIMM toolset might support is the addition of free text comments which are attached to certain time points or time spans aligned to the audio or video signal. This kind of annotation will usually suffice at the beginning of exploratory annotation processes, when a set of codes is still being developed, and in fact several tools in common use in the humanities offer this as their main functionality, perhaps with some additional post-structuring and tag name consolidation.

The next requirement for NIMM annotation arises from the fact that often users wish to annotate more than one kind of phenomenon on the same data set, keeping those kinds of annotation separate. A simple example is classifying both facial expressions and gestures on the same resource. Depending on the complexity of the coding schemes, coders might wish to perform both classifications at the same time or perform the two classifications in different passes over the data. Whichever method is used, NIMM tools must be able to maintain different layers of tags for different kinds of annotation.

Although simple tags which relate only to time points or time spans on a signal are sufficient for many purposes, many annotations in this field require more structure than timestamping provides. To give an example, it is important for the production of realistic avatars that the relationship between deictic expressions (such as "this") and the accompanying gestures (typically, pointing) is well understood. However, the accompanying gestures can occur temporally before, after, or during the deictic expression. It is impossible to determine the correct gesture to accompany a deictic expression simply based on timing, since the lag between speech and gesture in some cases may be bigger than the lag between gestures in others. Coders have no choice but to annotate the relationship between these phenomena structurally, rather than simply in temporal terms.

In other cases, structural relationships might possibly be derivable from timing information with fuzzy matching. For instance, to find the parse of a sentence where tags are only related to signals by timestamps, it may be possible to reconstruct the hierarchical structure by piecing together sequences of tags known to be syntactic categories allowing for fuzzy matching of timepoints and arranging these sequences in order by decreasing size of syntactic unit. However, these relations are primarily structural, not temporal. Treating them as temporal obscures the structure and makes it more difficult to create sensible graphical user interfaces for annotating the raw data. In addition, representing only timing information where structure is sometimes indicated can confuse the user into thinking that where tags are simultaneous, there must be a structural relationship. This is not the case; two phenomena can occur at the same time purely coincidentally, and sometimes determining whether there is a relationship between two phenomena is an empirical matter.

Multiple data streams, e.g. video recordings, might require multiple timelines (e.g. video-telephone conversation where both speakers are recorded separately). However, this will greatly complicate both the query language and its implementation.

An issue to consider is also the annotation of dynamic communicative events. An event may stretch across several frames and follow a trajectory across the image. For some purposes it might be desirable to annotate this.

NIMM annotations often relate to just one participant's behaviour, but this is not true for all of the annotations. For instance, dialogue games, a form of higher level discourse structure, reflect common sequences of utterance functions drawing from two people within a dialogue. This means that it must be possible with NIMM tools not only to indicate annotations on one participant's behaviour at a time, but that some annotations may refer to multiple data streams.

One issue of great importance for some NIMM users is data reliability - that is, how well two different humans performing the same annotation agree with each other about the occurrence of phenomena and their classification. Because of this, it should be possible within NIMM tools to maintain annotations of the same type made by two or more people on the same data, and to allow both, to keep such annotations separate or, whenever necessary, to get a merged picture.

Finally, it is becoming possible to code many features interesting to NIMM users automatically from video using pattern recognition. For instance, some sources claim (Cohn et al. 1999) that large-scale application of Ekman's FACS coding for facial expressions is obsolete, since, given some training data, the categories can be learned by neural networks. Where possible, NIMM tools should facilitate automatic coding. For very popular schemes, this may take the form of programs which perform the coding needed; for more esoteric schemes, it might be a workbench for constructing such tools. In some cases, automatic annotation can be useful, but not good enough on its own without human intervention. Correcting automatic annotation is simply another variant of the hand-annotation process, but one that should be supported.

## 2.3 Information query and retrieval

Since WP11 is concerned with NIMM annotation tools, this domain - i.e., annotation of NIMM corpus data - constitutes the principal focus of the requirements for query and retrieval functionality. Nevertheless, it is worth bearing in mind that other properties of corpora and of querying in general can play a role in determining this functionality, by complementing and interacting with the functionality specific to (NIMM) corpus annotation. These include the following:

- The raw corpus (object) data, i.e., irrespective of annotations or the representation of the data (i.e. their encoding and storage).
- The representation (encoding) of the corpus (both metadata, object data and annotations).
- The accessibility of the corpus (e.g., local or remote).
- General query and retrieval functionality, irrespective of corpus properties.

Before turning to query and retrieval functionality for annotated NIMM corpus data, we briefly consider issues relevant to functionality for these other domains. Some of these may be presupposed

by the NIMM-specific functionality, while others may be considered for added implementation, weighing benefits against costs.

### 2.3.1 Query and retrieval functionality for non-NIMM-specific domains

#### *2.3.1.1 Raw corpus (object) data*

The simplest example here is a text corpus, and the simplest functionality is text string search. This functionality is likely to be subsumed by more powerful devices implemented for corpus annotation, such as the use of regular expressions. In multimodal corpora, relevant functionality in this domain could include searching for particular properties of video images or acoustic signals. More specifically, the user could try to retrieve visual or acoustic patterns according to a (preset or user-) defined similarity metric. For example, selecting a particular region of a video segment (e.g. by dragging the mouse over it and clicking) would retrieve all (or a specified range of) sufficiently similar segments in the corpus. Likewise for searching for a visual display of a waveform or spectrogram configuration. For searching on an acoustic signal, the segment of interest could be demarcated by clicking on the desired start and end points. Since this type of query, if it is of sufficient generality, is likely to involve a considerable degree of vagueness, it should be augmented with a suitable relevance feedback mechanism. See e.g. (Zhong 2000) for a study in the domain of digital video libraries.

#### *2.3.1.2 Representation of corpus data and annotations*

This is primarily relevant for such issues as speed and efficiency of query and retrieval, but also bears on functionality relating to the representation of the query output. For example, if both object data and annotations are represented in a semistructured format, such as XML (as in, e.g., MATE), then query and retrieval can take advantage of this in selecting specific types of corpus segments, transforming the results (e.g. by means of XSLT), and so on. On the other hand, if the data and annotations have the stricter structure of a relational database, this could facilitate quicker retrieval. The mapping between these two types of representations, and in general their relationship to each other, has been addressed in much recent and current research, cf. e.g. (Suciu 1998, Bird et al. 2000, Isard et al. 2000).

#### *2.3.1.3 Corpus accessibility*

At issue here is whether query and retrieval of the corpora are done strictly locally, strictly in a distributed setting, or in a combination of these modes of operation. Within the local and distributed modes differences in functionality may depend on whether the operations access a single or multiple servers. Also, within the local mode there may be differences depending on whether the corpus and query engine reside on the same machine or are in a (local) client-server relation. Querying over metadata is a relevant functionality here, particularly in a distributed mode of operation, where it is essential to finding resources in the first place. But also with locally accessible corpora, certain kinds of metadata can be advantageous for retrieving particular segments of a corpus. Efficient querying over metadata entails the implementation of strategies to optimise precision and recall; see e.g. (Evert et al. 2001, Schlieder and Meuss 2000). Most flexibility would be achieved if both metadata and annotated properties of the object data could be used as search criteria, at least in a setup where the corpus and the query engine reside in one computer.

#### *2.3.1.4 General query and retrieval functionality*

The functionality implemented for query and retrieval depends on the particular domain of application: linguistic corpora serve different interests than customer databases, for instance. However, a range of query and retrieval functionality is, by and large, application independent. For example, the requirements being developed for the querying of XML documents (W3C 2001) contain a list of twenty kinds of functionality. A number of these are specific to the design of XML (which could also be relevant to the concerns of ISLE, to the extent that XML is the choice of representation language),

but many are not tied to the details of the representation. The following requirements from (W3C 2001) are among those of most general interest:

- universal and existential quantifiers
- hierarchy and sequence operations and traversal
- combination of related information from different parts of a given document or from multiple documents
- aggregation (i.e., summary information about related groups of data, as in statistical queries)
- sorting
- composition of operations, including the use of queries as operands
- structural preservation of the source data in the query output
- structural transformation of the source data as a result of the query
- references, both with a source file (e.g. corpus) and to external files
- operations on schemas, if these are used to regulate the format of the source data
- extensibility to the use of functions other than those defined in the query language
- environment information, such as data, time, locale, user

Additional useful general query functionality, part of which seems to be implied by the above, include the following:

- wild cards (i.e., partial strings as metavariables over sets of strings)
- regular expressions
- Boolean combinations of query expressions
- selecting subcorpora (i.e., start and end points of the query properly included within the current corpus)

It would also be helpful to users to have some interactive editing support, to increase the speed of formulating queries and reduce the occurrences of error. Relevant functionality includes:

- expansion and completion of controlled vocabulary (perhaps extensible by user definitions, e.g. tag tables)
- storing and naming formulated queries for reuse as macros (a library of query modules).

Some of the functionality requirements listed above may be delegated to modules external to the query processor in order to avoid reimplementation of existing efficient tools, or to build more modularity into the system architecture. For example, aggregation and statistical queries could be imported by providing an interface to dedicated tools for these tasks. These functions may be shared with components of the tool set for data analysis (see Section 2.4 below). Sorting and structural transformation could be performed by style sheets, as in MATE. A drawback is that interaction with external modules is often difficult to achieve and may introduce severe performance penalties – specifically within query execution.

An additional general functional requirement of query languages, which bears particularly on the design of the query language, is optimisation, i.e., the ability to express certain queries more compactly and process them more efficiently. This may also relate to the requirement of extensibility listed above. It may be difficult to bring the requirements of generality (in the sense of the widest functional coverage possible) and of optimisation together. Thus, optimisation may be applied to certain, much used, standard operations only. See, e.g. (Suciu 1998) and (Bird et al. 2000) for discussion of query optimisation.

### 2.3.2 Query and retrieval functionality for NIMM corpus data and annotation

In general, annotations of text can refer to multiple distinct levels of linguistic analysis, as in MATE and ATLAS. Hence, query functionality for multilayer annotation is a suitable starting point for considering the additional functionality required for multimodal corpus query and retrieval.

Perhaps the most significant difference between multilayer annotated corpora and corpora with a single annotation layer is in the kinds of hierarchical structure possible. In a single layer the annotations can be related to each other by a single hierarchy, whereas annotations across multiple levels may well involve intersecting hierarchies. If the corpus representation format is a strictly hierarchical language like XML, then some formalism is required to enable navigation of, and reference to, intersecting and overlapping hierarchies, such as stand-off annotation, using hypertext linking mechanisms (cf. (Mengel et al. 1998) and (Isard et al. 2000), or directed graphs (cf. (Bird et al. 2000).

Such representations require a suitable query output. For the MATE query language, it was decided that a subcorpus is not by itself suitable: "[I]t will make no sense at all to provide a sequence of words as result. Rather, it seems useful to provide information on where units of such constellations [i.e., annotated segments at any level] can be found, as it cannot be determined what the user wants to be displayed. It is then up to the user to select appropriate views of the locations that have been found (Mengel et al. 1998)."

While this decision delegates to users both the flexibility and the responsibility of choosing the output from among the relevant annotation layers, there is no reason why this couldn't be implemented by the system design. Especially in the context of NIMM corpora, this seems like a quite useful functionality. For example, a query result could select a particular video or audio segment, and any or all associated annotation layers could be displayed along with it (including a transcription, which corresponds to the object data of a text corpus). In fact, this is essentially how query is implemented in the EUDICO system (cf. (Brugman et al. 2000) and the description in (Dybkjær et al. 2001)). The choice of the layers to be displayed for the query results should be left to the user and may, but need not, be identical with the layers used in query conditions, nor with those which happen to be displayed at query time.

Querying over temporal relations, already implemented in MATE, is equally relevant, if not more so, for NIMM corpora. The functionality is particularly useful in combination with video and audio segments, which will generally (and probably be required to, as an annotation standard) be time-coded. Thus, annotation layers can be readily aligned with the raw object data segments and linked to each other in the output. Again this is implemented by EUDICO.

## 2.4 Data analysis

NIMM data users currently use a great many different functions as part of their overall data analysis strategies. This includes statistical analyses, both basic descriptive analyses and inferential statistics testing specific hypotheses. This requires the full range of functions currently supported by statistics packages: graphing, histograms, and implementation of the most relevant statistical tests. Because many NIMM data resources contain hand-coded annotations, it is important that reliability testing and diagnostics be well-supported. This includes not just calculating agreement statistics across two or more coders of the same phenomenon, but enabling the analyst to determine where unacceptable amounts of disagreement lie. Data analysts also require the ability to construct statistical models of the data, for instance, by determining the probabilities of the different possible sequences of phenomena in the data set using either an element-based or a time-series view. At least NIMM data users from the language engineering tradition will tend to use Markov modelling techniques for this rather than the lag sequential analyses more prevalent in observational analysis.

An important aspect of data analysis needs for NIMM users is that because NIMM resources are expensive to collect, they tend to be annotated with many different kinds of annotations. This is not simply a matter of data re-use, since for many applications the important issues are how annotations interact with each other. (For instance, in constructing the next generation of avatars, one major issue is the relationship between facial expression and what the avatar is saying; the lip shape for a phoneme

depends to some extent on whether the speaker is smiling or frowning, and so on.) This is the reason why the query language for finding and extracting interesting parts of the data set must be more complex than for many tasks; it must allow matching based on both structural and timing relationships. In addition, simply having a query language that can find matches is not enough. Users working with this kind of data require the possibility of visualizing the data in different ways. For instance, one might for the same data need to show transcription with pauses, changes in body posture, and dialogue structure clearly indicated for thinking about what might be a useful indicator of structural boundaries, transcription with deictic expressions and gestures indicated for exploring the timing relationship between deictic expressions and pointing, and so on. Typically NIMM data sources may be annotated in so many ways that viewing all annotations at once overloads the analyst, but, given the exploratory methodologies involved, it is not possible to pre-specify all the data views that might be required; instead, NIMM users must be able to build tailored displays quickly and with as little skill as possible required.

Depending on the amounts of data available, clustering techniques may be useful for explorative analysis of the co-occurrence of several phenomena. Patterns of co-occurrence of phenomena from different modalities and/or annotation layers may be brought out in this way more clearly. The tool should thus offer the possibility to feed the above mentioned selective visualisation component also with data generated by (a possibly external) component for explorative data analysis.

## 2.5 Metadata

So far, we have concentrated on the functionality which NIMM users require for working with individual data resources (typically one video recording representing one dialogue, one session of a user interacting with a system, one small group discussion, or whatever the situation under study) or for analysing the annotations on all of the data resources within some coherent set. As well as supporting these functions, NIMM tools must support the annotation and use of metadata. Individual data resources need metadata explaining the conditions under which they were collected (for instance, in studies of human-human dialogue, often the sex, native language, dialect, and age of the participants, plus any conditions which may have been controlled in an experimental setting). In addition, entire data sets may be held within archives, with the data set identified by the type of material it contains. NIMM tools should support the annotation of individual data resources and of complete data sets with metadata information, and then allow the individual resources and sets to be indexed and searched by the metadata fields. In fact metadata can simply be integrated as document-level annotations so that no special query functionality is required. However, appropriate display modes are desirable.

## 2.6 Summary of NIMM functionality and current provision

In specifying requirements for future NIMM tool development, it is important to distinguish functions for which there is already good provision from those which are under-supported. It is not appropriate to review existing tools here, since this was done in ISLE deliverable 11.1 (Dybkjær et al. 2001). However, at this point it is as well to indicate broadly where the gaps in provision lie.

Currently, support for the annotation of video and audio signals is good as long as the annotations are simply given as time points or time spans on the signal. For instance, Noldus, a Dutch company, distributes a highly popular package for producing and analysing the resulting event sequences already. Some European sites have existing, lower profile tools in this area, such as MPI's MediaTagger. A number of projects in the United States linked with the Linguistic Data Consortium's development of the "annotation graph" formalism intend to provide further support. These tools tend to allow for tags to be grouped into classes, thus providing support for layered data annotation, but do not allow structural relationships among tags to be represented.

There are existing tools which support individual coding schemes that include structural relationships (for instance, coders for Rhetorical Structure Theory). However, in these tools the coding scheme itself is hard-wired in, or only slightly configurable, for instance, by changing the names of the basic tags but assuming one particular structure. This suffices for popular schemes that have managed to

find someone to write the individual tool, but gives poor support for new development in NIMM codings because it is difficult even for the tool developer to adapt the tool to new coding schemes. Despite this, supporting new schemes is very important. Even basic relationships in NIMM data are not yet completely understood (for instance, the best indications that a speaker is coming to the end of a turn), but understanding these relationships will require schemes tailored for specific research questions.

The MATE workbench has demonstrated that current XML technology can be used to support the annotation process in general. It uses stylesheets to transduce XML data into Java display objects with which the user can interact. This technology has been demonstrated with several popular coding schemes for differing kinds of spoken dialogue phenomena, including coreference and dialogue act annotation, two of the most common concerns of language engineers. The strength of the workbench is that using it, one can create annotation and display support for new coding schemes by writing stylesheets, not complete tools. Stylesheets are written in a language based on a W3C standard, XSLT, a technology for transducing XML data to an HTML display. This technology is quite commonly used now, and processing engines for it are included, for instance, in Internet Explorer 5. However, even though MATE demonstrated that the basic concept will work, it was not successful in providing finished, usable, maintained, and supported software. Its main flaws are slowness and difficulty in writing the parts of stylesheets which specify the effects which user input should have on the displayed data. However, even if it were perfect it is not clear that it would be adopted successfully by end users. The best example of high profile software in this area, Noldus' Observer-Pro, relies on initial training sessions and helpdesk support for its continued popularity.

Annotation tools tend to be used for both data display and visualization as well as the annotation process itself. As with annotation, the MATE workbench is currently the only attempt to provide truly configurable display, as needed for the exploration of data described in section 2.4. Statistical analysis, however, is currently well-supported through a range of popular packages (SPSS, Minitab), with the possible exception of reliability analysis. Reliability statistics are not available in all packages, and packages do not provide structured support for diagnosing where data is unreliable.

We see the biggest gaps in support to be in the area of structured annotation, especially for the development of new schemes, and in support for the use of metadata.

# 3 Interface

An adequate interface is crucial for any tool and thus also for NIMM tools. Without an adequate interface many users may never find out about part of the functionality offered by a tool, or they may have difficulties exploiting its functionality because it is not obvious how to use it, or they may simply give up on the tool. The interface should among other things provide easy access to all system functionalities and follow general interface conventions which users are familiar with from other programs. Conventions vary to some extent across platforms. If it is decided that the tool should run on multiple platforms, such variations have to be taken into account, cf. also Chapter 4.

This chapter does not provide a detailed interface specification for one particular NIMM tool. It stays at the same level of abstraction as Chapter 2 on functionality, i.e. it considers requirements and issues to take into account in the development of a NIMM annotation toolset. Given this level of abstraction it seems adquate to take e.g. the GUI principles proposed in (Johnson 2000) as point of departure. Those GUI principles are not specific design rules but rather form the basic background for the design of effective and user-friendly interfaces. Johnson's eight principles include:

- Focus on users and their tasks, not the technology.
- Consider function first, presentation later.
- Conform to the users' view of the task.
- Don't complicate the users' task.
- Promote learning.
- Deliver information, not just data.
- Design for responsiveness.
- Try it out on users, then fix it.

## 3.1 Focus on users and their tasks, not the technology

The target users of the NIMM tool were discussed in Section 1.2. As mentioned, a general-purpose annotation toolset can be expected to be of potential interest to people with different professional backgrounds working with annotation in many different areas, from several different perspectives, and with an unlimited number of different annotation purposes. The user interface discussed in this chapter should, first of all, support users who need a tool which allows them to do their tasks of annotation, data extraction, and analysis without bothering about the internal workings and representations of the tool (cf. Section 1.2, group one users). Secondly, the tool should offer a developers' interface which allows users who have the necessary programming skills to add new functionalities to the tool (cf. Section 1.2, group three users). However, the latter is another kind of interface which is rather related to the architecture of the toolset and its modularity. It requires e.g. that the code of the toolset is well-documented and easy to hook on to. The second user group mentioned in Section 1.2 will not be considered in this report. Access to e.g. XML-representations should not be prohibited but is not something we would encourage from an interface point of view. Moreover, the amount of users belonging to group two may decrease as tools grow better.

Before designing a specific NIMM toolset it would be good to collect more detailed information than we have for the moment on target users' likes and dislikes, their skills and knowledge, their particular tasks, needs and preferences, etc. This could be done by inviting a representative group of users and talking to them, by communicating with them via email, interviewing them on the phone, asking them to fill in questionnaires, etc. Section 1.3. outlined the domain and task coverage of the NIMM toolset in general. Collaboration with real users will also provide the developers with much more detailed knowledge about the task domain. Such detailed knowledge is necessary to optimise users' interaction with the tool.

## 3.2 Consider function first, presentation later

Before considering the actual appearance of the interface it is necessary to make clear the purpose, structure and function of the interface. A common approach is to develop a conceptual model that e.g. describes which concepts are to be included in the tool and presented to the users, which data the users will create, view or manipulate with the tool, and what options, choices, settings and controls the tool will provide.

The purpose of the NIMM toolset interface is to support different user groups as described in Section 1.2. However, in this chapter focus is on the first category of users. The structure of the interface is closely connected to the functions which need a structure that reflects user expectations on how to carry out their tasks.

Overall functionalities for the NIMM toolset were discussed in Chapter 2. A conceptual model would need to include more details. We shall not develop a conceptual model in this report since such an exercise is being made in the NITE project (http://nite.nis.sdu.dk). We shall only mention examples of functions in the following.

The main challenges would seem to be in how best to provide interface support for a tool which cannot be tailored to the individual user in advance since it is supposed to be a general-purpose NIMM annotation tool.

The toolset may come with a set of best practice annotation schemes. However, there will always be a need by some users to either modify these schemes or add new ones. The tool must include an easy to use interface for the addition of new coding schemes. It should be clear to the user which information to enter in order to have a new coding scheme and how to do it. The interface should not require any knowledge at all of the underlying representation of the coding scheme. Conversion to a machine-readable format, such as XML, should be done internally in the tool.

Coding schemes may vary considerably as regards their requirements to how they are presented to the user in an annotation window. In the MATE workbench, the interface to an annotation window for using a particular coding scheme, is tailored to that particular coding scheme. This has been necessary due to the very diverse requirements of annotation schemes for markup of spoken dialogue at different levels. The big problem is that stylesheets are used for the interface definition so that when a user wants to define a visualisation for a new coding scheme, he has to write a style sheet, which requires programming skills and has been a stumbling stone for many users.

A NIMM annotation tool has to support users in defining new visualisations. It is necessary to provide an easy-to-use interface for the definition of visualisations even if this may be difficult to do. This may perhaps be done by offering the user the opportunity to define a visualisation by choosing from a palette or menu how tags should be presented, how many windows or panes are needed, what each of them should contain, which operations (writing, selecting, ...) should be possible in them, etc. Depending on the number of options and combinations to be made available, this may be a flexible way in which to define a new visualisation and definitely more adequate for standard users than stylesheet writing.

The NIMM toolset should enable precise selection of what to mark up either by selecting the relevant piece in the transcription file or by, e.g., encircling the relevant piece on the video frame. Tags may be selected either by clicking on them in a palette or by using shortcuts. Also the issue of how to tag dynamic events has to be addressed.

Automatic synchronisation of an annotation file and a video file and a speech file should be supported at the interface level. It may also be desirable to display synchronised visualisations of different coders' annotations of the same file, and of annotations for different levels or modalities together with the audio/video material.

It is important that the search that the search window which appears when search is activated, is comprehensible. There should be maximal support for query formulation in the search functions, possibly with different levels of search capabilities offered (simple search versus advanced search as in any web search tool). It should be possible, e.g., to type in a text string to search for (either in the entire file or in a selected part of it), or choose from a list one or more relevant tags to search for.

Advanced search could allow the user to enter, e.g., Boolean, temporal or hierarchical queries. The interface should support the user in formulating the queries correctly. Although the notion of query by example is not easy to realise for searches covering the whole complexity of the annotation, graphical support would be most useful. Details on the exact needs of the query interface will have to come from an investigation of what users actually want to query.

Metadata descriptions are desirable in connection with transcription as well as annotation. The metadata for a transcription file may differ from those for an annotation file, and across transcription files and annotation files there may be variations in what users want to indicate. The window for metadata in which some fields may be predefined and some may be mandatory should offer the possibility of adding new fields defined by the user. It should be possible to view the metadata in a text file.

## 3.3 Conform to the users' view of the task

Software tools should be designed from the users' point of view. One of the most important things is to make the interaction with the tool feel natural to the user. This requires that developers know what the users' point of view is. Such knowledge may be achieved e.g. by interviewing and collaborating with representative users. We shall not go into detail with what the users' point of view may be for a NIMM toolset. In the NITE project (http://nite.nis.sdu.dk) this is being explored in more detail. The overall goal of the NITE project is to implement a NIMM toolset as outlined in this report. The approach to specification is to (i) ask potential users of such a toolset what they consider to be important functionality and interface issues and (ii) try out in detail a number of existing NIMM tools among those which were reviewed in (Dybkjær et al. 2001). This provides detailed input on e.g. users' preferences, their interaction with existing tools and on what is natural and what is less natural.

An example of unnaturalness can be drawn from Microsoft Office. This is not a NIMM tool but the following problem is probably well-known to many people and illustrates unnaturalness quite well. In some cases when one has made a selection of a piece of text e.g. to be turned into a different colour or into boldface, the act of colouring or making the text bold not only affects the selection but actually the entire document. In order to only affect the selection one has subsequently to select undo. This is definitely not natural to users.

## 3.4 Don't complicate the users' task

Common tasks which users can be expected to carry out often should be straightforward to do and require a minimum of effort by the user. More rare tasks may require the user to do more in order to achieve the goal. Moreover, the functions provided by the interface should be clear and obvious and the user should not have to spend time on interpreting inappropriately formulated terms and messages from the computer which pop up while carrying out a task.

To accommodate the requirement of good support for common tasks, the NIMM toolset should e.g. provide reasonable defaults for visualisation, a set of frequently used coding schemes which users can go ahead and use for annotation without any additional effort, and easy access to search facilities. The toolset should also support preferences and customisation of the interface so that users may decide on font type and size, colour, toolsets to view, define shortcuts, save certain settings, etc.

As an example, the MATE workbench comes with a number of best practice coding schemes which were added exactly to facilitate annotation for those who want to use one of those frequently used coding schemes. Users may also add their own coding scheme but this is not as well-supported by the workbench user interface as desirable since it requires some XML-literacy to do.

## 3.5 Promote learning

Most tools require some time for learning. However, the more the user has to learn before s/he can use the tool for actual work, the greater is the risk that users will not bother about using the tool. Learning time can be minimised in several ways. Important parameters to consider are consistency and intuitiveness of design. Consistency means that the interface has the same look-and-feel throughout

the toolset while intuitiveness of design means that users normally find what they are looking for in the place in which they are looking for it. Moreover, the toolset should allow exploration without the user running a risk of loosing work.

An example of interface design to avoid is the search facility offered in the MATE workbench. The interface for inputting a query is very difficult to understand and use and the output is presented to the user as a list represented in XML. Another example is the need for writing stylesheets if the user wants a new visualisation as may often be needed for new annotation schemes. This makes many users refrain from using the MATE workbench. The learning overhead is simply too big.

The NIMM toolset should draw on usual conventions as regards where to find which functionalities. For example there should be an edit menu which allows the typical functions of e.g. cut, copy, paste, and undo. Another important lesson learnt from the MATE workbench is that, in particular for category one users, cf. Section 1.2, it is very important to avoid XML at the interface level and to not ask users to become programmers in order to use the toolset.

## 3.6 Deliver information, not just data

Displays must be designed carefully. The users' attention should be directed towards the important information. For example, tags inserted in a text should be clearly highlighted during annotation to tell the user which information s/he has added.

The example in Section 3.5 of how the MATE workbench represents results from the query tool is an example of delivering data rather than information. It is completely left to the user to see if s/he can extract any information or meaning from the data.

In general, attention to detail is important to achieve effective displays and avoid inconsistencies and other inadequacies in the interface.

Since a NIMM tool may provide too much data in the first go, it may be useful to provide, where appropriate, a sort of summary first before allowing the user to expand certain types of information. For example, a list of query results could be given, in the first place, in terms of textual transcripts only or of small video pictures and the user would then choose the instance(s) s/he is interested in. Result summarisation, abstracting and clustering should be key elements of the query/display module.

## 3.7 Design for responsiveness

Responsiveness means that the tool should provide sufficient and timely feedback to its users so that they always know what is going on. A very important factor is perceived speed. Users hate to wait. If the tool has to perform an action which takes time, the user should at least be informed about this and be told approximately how long the operation will take. Time-consuming operations should not block other activities and it should be possible to abort them. Thus, the users should be those who decide on their work pace – not the system. A serious problem we have experienced with the MATE workbench is exactly the (lack of) speed. One of the most frequent complaints is that the workbench is far too slow. Speed is a problem in very central and frequently used parts of the workbench, which makes matters even worse because users cannot really do their work without waiting. This is probably mainly due to a non-efficient design of the internal representation and the query processor. Moreover, there is only a very primitive kind of process feedback telling that the system has not broken down but it does not tell how long the user should expect to wait.

Speed and efficiency may be reasons to optimise certain processes (e.g. query) for certain applications, thereby sacrificing, for frequent usage types, the idea of generality. In the example case of the query module, this would imply to have a general (but slower) facility along with a more efficient subset of the query language.

## 3.8 Try it out on users, then fix it

This points links up to Section 3.1 with its focus on users. It is not enough to consult users in the initial development phase only. Once a model of the functionality and of the interface is in place it can – and

indeed should - be tested on users before the tool is implemented. Testing with users is an iterative process which should continue throughout the development lifecycle of the toolset to obtain the best results.

If the NIMM tool for which a set of requirements is specified in this report, is developed at some point of time it is recommended to involve representative users from the very beginning and throughout the development process to obtain information on what the toolset should and shouldn't do and to get iterative feedback from users while the tool is being developed.

# 4 Ramifications of requirements for NIMM software

Giving implementational details which are opaque to the user is not conventionally part of the requirements specification task but part of the software specification undertaken at the next stage of the design process. Of course, user requirements always have important ramifications for such implementational details. In this section, we describe some of the ramifications of the requirements specified above for any eventual implementation, and what we perceive to be the best implementational choices available.

## 4.1 Data storage

The purpose of the NIMM tool is to support users annotating and working with annotated multimodal data streams. These annotations have the following properties:

- they must be interpretable in relation to the underlying audio and video signals;
- rather than simply representing every annotation as connected to a particular time span of a signal, they must allow complex structural relationships among different annotations to be represented, not all of which are hierarchical;
- they will sometimes be created using other tools, or will be analysed by external means, and therefore the storage format must either be non-proprietary and well-documented or else the tool must incorporate translation into a non-proprietary and well-documented format;
- they are often created by several different people, sometimes at different sites, working on the same data resource, and therefore it must be possible for the storage to be distributed.

Within the MATE project a scheme for data storage was devised which satisfies most of these criteria. It represents annotations using XML, a markup language which has been devised by the W3C and is used extensively in web applications to mark up the structure of written text. To represent data in XML, one first writes a document type definition (DTD) or schema. This defines a set of markup tags and their expected structure on the data. For instance, in order to mark up morphosyntactic information on a transcription of speech, XML data is inherently hierarchically structured, since each XML document conforms to one DTD, which specifies a tag which spans the entire document and how that tag can be decomposed into other tags and eventually, character data. However, XML admits the possibility of using hyperlinks to connect data from different documents.

The XML used in the MATE workbench decomposes the structure of a set of annotations on the same data into a number of interlocking hierarchies. Often it is sufficient for there to be one transcription and then a number of hierarchies cantilevered from it representing different linguistic phenomena (for instance, one for speech disfluencies, one for syntactic information, one for discourse function, and so on). Nothing precludes connection between hierarchies above the transcription level (for instance, one might wish to indicate discourse functions which assume syntactic sentences as their base), or the possibility of multiple transcriptions for the same signal (for instance, separate orthographic and phonetic transcription). Each hierarchy in the data decomposition is specified in a different DTD and stored in a different file. Use of the data requires only those files relevant to the task at hand, reducing processing load and allowing for distributed data development. The relationship between annotation and signal is specified by hard-coding an interpretation of specific XML attributes; if a “start” attribute is given for a tag, this signifies the offset from the beginning of the audio stream at which the annotation begins. Similarly, an “end” attribute specifies offset from the beginning of the audio stream at which the annotation ends. Where to find the signal itself is indicated in header information associated with each document. This method of data storage allows for the representation of complex structural relationships whilst retaining the spirit of the formats recommended by the Text Encoding Initiative and by EAGLES for transcription, both of which had more limited structural requirements.

One shortcoming of adopting a MATE-style XML structure for data storage is that it does not make provision for multimodal data. Annotations in MATE relate to an audio signal by specifying a file location for the signal and time offsets from the beginning of the signal which specify the beginning

and end of the annotated event. The NIMM tool can allow annotations which cover a particular time span on a video stream in an entirely isomorphic manner. However, for working with NIMM data we further require the ability to specify a spatial location, or possibly an area, on the video signal during that time span. This is to allow the annotator to focus attention on the exact part of the picture indicated by the annotation. In these cases, it is generally sufficient to allow the annotator to specify a visible box which is static for the duration of the time span. It is relatively straightforward to extend the MATE conventions about how to interpret links to signal in order to allow the box's location to be given in terms of, for instance, proportional offsets from the top left of the picture.

An additional shortcoming of MATE's XML is that at the time it was developed, the W3C had not yet standardised the mechanisms for hyperlinking in XML. They are currently in the process of this standardisation, via XLink (http://www.w3.org/TR/2000/PR-xlink-20001220) and XPointer (http://www.w3.org/TR/2001/WD-xptr-20010108). Whilst it is unlikely that MATE's choices of how to specify hyperlinks will be adopted verbatim by the W3C, we expect the eventual standard to be set in 2001 and be a minor syntactic variant of what MATE has adopted. In order to facilitate data exchange and external processing, the NIMM toolset should adopt the standard format.

## 4.2 Internal representation

The internal representation used in the MATE workbench suffices for NIMM software, and is both relatively straightforward and standard, in that it is based on the XML Document Object Model (DOM) (http://www.w3.org/DOM/).

In MATE, each XML element and piece of text is represented as a Java object. Each object consists of pairs of **properties** and an associated **value**. Thus the whole internal representation consists of triples of <*node, property, value*>. Properties generalise the attributes of an XML element and most are string valued, but some have values which are lists of other nodes in the internal representation, for example the **children* and **parent* properties. As a simple extension to the standard Document Object Model (DOM), DTDs are also represented as objects. The types of nodes and their interrelationships are shown in figure 4.2.1.

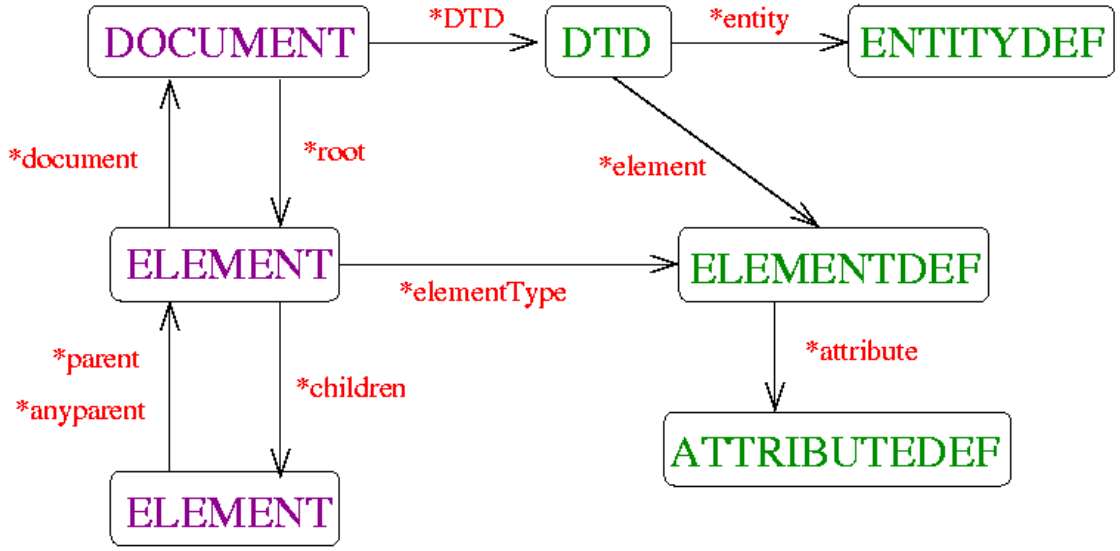


**Figure 4.2.1.** Internal representation of data for the MATE workbench.

MATE made one major extension to the DOM model of XML structure in order to support multiple documents and overlapping hierarchies which are needed to describe multiple speakers and multiple levels of annotation in dialogue corpora. This is supported by allowing nodes in the IR to have multiple parents. Each node has a privileged parent which is the parent element in the same file as the node. However, a node may be a child of other elements in different files, attached using hyperlinking,

as previously described. For example a <word> element might be a child of a <wordlist> element, but also be a child of a <phrase> element.

An important point about this internal representation is that it allows absolutely all information about the data to be processed in the same format. This format is used for the annotation data, but it is also used to represent the results of queries expressed over that data, and to represent the stylesheets which are used to describe the appearance and behaviour of the user interface. This has the advantage that, for example, the MATE workbench could be used to provide a stylesheet editor, or the query language could be deployed over stylesheets, which is useful for debugging. More importantly, since the IR internal structure is isomorphic to the structure of an XML file (as long as extensions are not used, e.g. multiple parents), query results can be output as XML files, with stylesheets then applied to the results XML so that they can be displayed to the user in a number of formats. In particular, since query results can be specified to contain pointers to the elements that satisfy the query, this form of internal representation allows one to display results either separately from the corpus or by highlighting elements in their corpus context. This form of internal representation thus has major advantages for making a workbench extensible for new annotation schemes and new functionality. Note that since the relationship between an annotation and the speech signal in MATE is handled as XML data with a privileged interpretation, and since relations to a video signal can be handled in the same way, this internal representation is wholly compatible with providing support for multimodal data.

Note that this is only one of the possible internal representations for this form of data. This form privileges the information about structure in the data by presenting the annotations as a forest. One might instead privilege only the information about parents in the same document, reducing the structure to the core hierarchies, or only timing information in relation to the underlying speech or video signal, ignoring structure among the tags altogether. Each of these choices has ramifications for efficiency and for how easy different kinds of queries are to express. For data which is annotated for a wide range of coding schemes — a hallmark of the MATE approach — structural relationships are of great interest. Therefore MATE employed a forest view of the data. Structural relationships are also paramount for multimodal data (section 1.1 point 6) and so the same representation is appropriate.

Before a NIMM tool is implemented other means of external representation than the one used in MATE and described above should be considered. For instance a database representation may turn out to provide a more efficient tool.

## 4.3 Platform requirements

Section 1.4 briefly mentions that it may be desirable for NIMM software to run on several different platforms (Sun workstation, PC, Mac) using a wide variety of operating systems (Unix, Linux, all versions of Windows, Mac-OS). On the other hand it is well-know that platform independence comes at a price. The only way to construct software which works on this range of platforms currently is to write NIMM software in Java. Java is in principle available for all of the platforms although, historically, implementation for the Mac has lagged behind the others. Using Java does have some serious disadvantages. The most important of these is that Java runs slowly. The MATE workbench runs slowly enough that there are some realistic annotation tasks, which represent actual user needs, for which it cannot be used. Some of this slowness could of course be corrected with more efficient re-implementation; in addition, Java's speed itself is currently being improved. However, code implemented in any platform-independent mechanism is necessarily relatively slow because it cannot take advantage of platform details in order to gain efficiency. Thus, choosing Java in order to please users across all platforms necessarily means not serving subcommunities which use specific platforms as well as we might. The second disadvantage is that cross-platform development requires extra effort because, even though by and large the code will work on different platforms, there are always some snags. If nothing else, extra effort is required to determine software installation procedures for the various platforms.

Another current development could help in the delivery of cross-platform systems: software which allows for interoperability of different operating systems on the same machine. VMware (http://www.vmware.com) allows interoperability among Windows 95, Windows 98, Windows NT, Windows 2000, FreeBSD, and Linux. In theory at least, software developed for any of these platforms

can be run on any of the other via VMware, even without setting up a machine as dual-boot. Since a reasonable number of NIMM users use Sun workstations, and since the easiest port between Unix and any of these platforms is with Linux, it might be possible to write Linux software, assume users on the other VMware platforms will use VMware to run it, and port the software to Unix. Although to our knowledge none of the current activity in interoperability involves the Mac platform, this platform is perhaps losing popularity among our prospective user community. VMware provides us with a useful fallback position, should Java fail to provide true cross-platform capabilities, or should implementation of some NIMM capabilities be too difficult in the Java environment. This approach would require NIMM software users to obtain commercial licenses to these environments, but that may be an acceptable price to pay for easier software development. It is still too early to say whether this approach actually performs adequately in practice and what the extra development costs might be.

## 4.4 Software architecture

Because we require the NIMM tool to be extensible, this places certain constraints on the software architecture. The most important of these is that it should be possible for users to define and employ novel types of annotation on multimedia data. With few exceptions, existing tools hard-wire in one or more types of annotation. If the user wants to annotate using different categories, they cannot use the tool. Occasionally (cf. nB, http://www.sls.lcs.mit.edu/sls/publications/1998/nb.zip) a limited amount of tag configuration can be specified in a control file or (cf. The Ethnograph, http://www.qualisresearch.com/) free-structured tagging is allowed without specifying relationships among different tags. The real limitation here is that there are no mechanisms for specifying the graphical user interface for viewing data or for performing an annotation.

The MATE workbench allows the user to specify the structure and meaning of annotations via a set of document type definitions (DTDs). It also allows the user to specify as many specialist views of the data and graphical user interfaces for modifying the data as the user requires, through the development of style sheets. A style sheet specifies a transduction of XML data into some other form, which can variously be XML with a different format, HTML (for display in a web browser), or, in this case, Java display objects with which the user can interact. MATE developed its own version of style sheets for this purpose because at that time the W3C standard, XSLT, had not yet been finalised. The parts of the MATE stylesheet language which handle display are a syntactic variant of the new standard. However, XSLT does not, and is never expected to, cover GUI specification. A reasonable architectural choice for NIMM tools would be to revise the MATE stylesheet concept by adopting standard XSLT where it is relevant and further developing the constructs which extend it to graphical specifications, using a modular concept. An important thing is to allow users an easy way in which to specify new visualisations, i.e. users should not be asked to do any kind of implementation such as writing a stylesheet without a good editor.

The software architecture for the core MATE functionality is shown in figure 4.4.1. In addition to these components, there was an overarching user interface which could be used to specify the user input for these functions (e.g., which stylesheet/data combination to run in order to produce a display, data extraction, or GUI interface; query specification on already loaded data using a GUI or textual query interface) or to call external functions, such as an audio tool which played entire sound files at a time, or converters to and from external data formats. This architecture is just as appropriate for multimodal data as it was for spoken language data.

Cf. Section 1.1 point 1 it should be possible for users to add entirely new functionality. Not only should it be possible to specify new types of annotation over new types of data, but also to specify new processes over annotations. For instance, the MATE workbench does not contain tools for statistical analysis or for transcription, even though users clearly need these functions at some point in the research process. Users also need tools which convert other data formats into, and out of, the MATE XML structures; some such converters are accessible from within the MATE workbench, but not all, and certainly not converters for data formats which have not been developed yet.

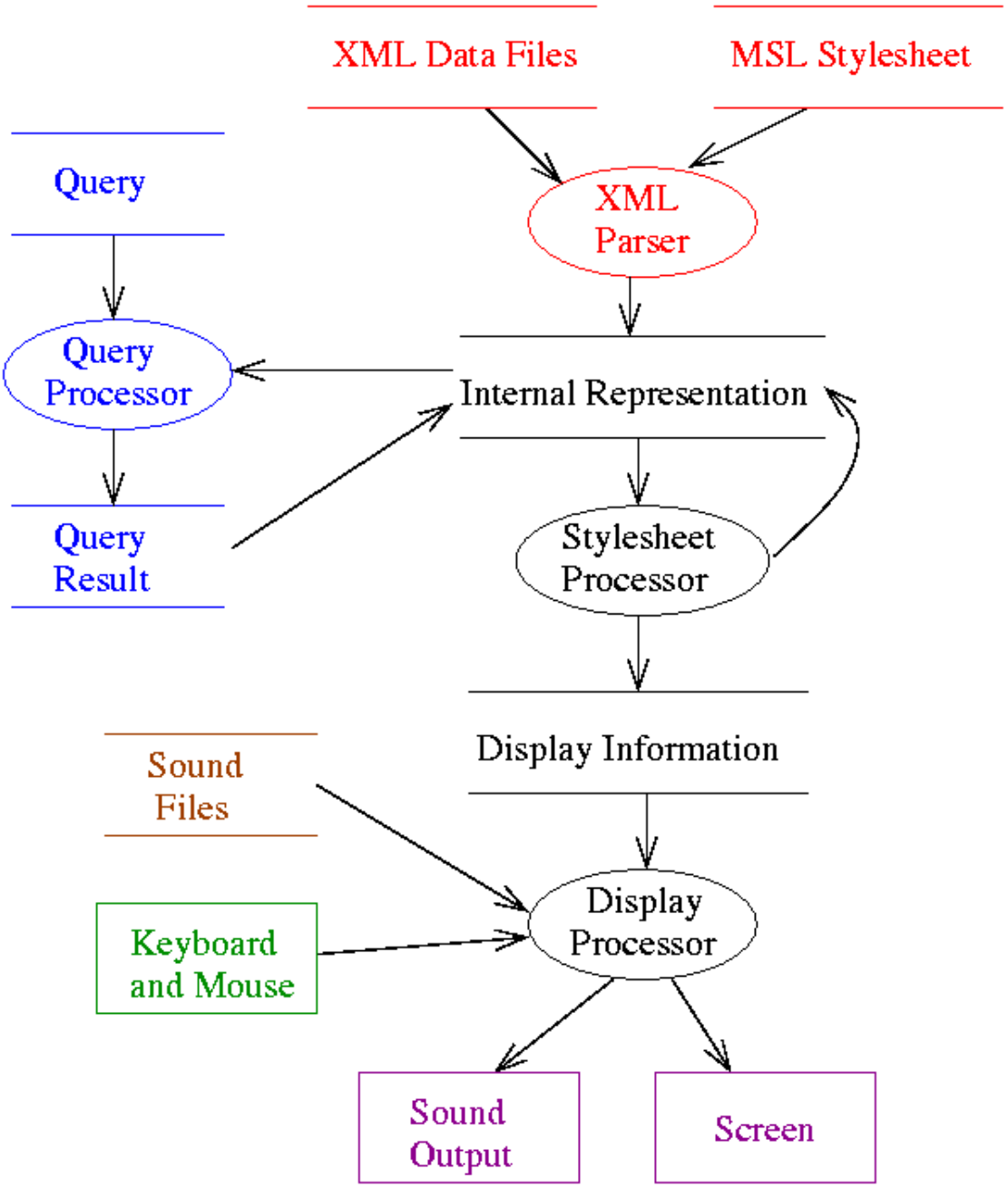


**Figure 4.4.1.** Architecture for core MATE functionality.

Some users, especially ones who are less comfortable with the entire set of steps needed for working with data, would prefer all of these functions to be accessible from within one workbench. However, the strategy of presenting software to the user in one monolithic structure creates difficulties. The first is that monolithic software structures tend to be slow and cumbersome. The second is that not all software can be hooked into an existing structure; the structure itself imposes implementational constraints. For instance, with the MATE workbench, only Java programs could be inserted into the overall structure. The third is that, paradoxically, the users who most require a monolithic structure are likely to be the least able to incorporate new tools into an existing framework, simply because of the computational skills required — but every subcommunity of users has different ideas about what tools are essential, making it impossible for the tool developers to perform the integration. This means that,

as with popular software packages such as Microsoft Word, even where developer effort is put into making software tailorable to individual users, this ability is rarely used in practice.

One possible approach to help this sort of user is to build something like the Mac OS “Launcher” or Windows toolbar which simply gives a convenient place where users can go to find the start-up methods for all of the tools which they use for working with multimodal data. This would include new NIMM tools, enabling new functionality, as well as old favourites, and, as they arise, converters among formats needed by the different tools. This in no sense integrates the tools, but at least makes it easier to remember what tools are available and where to find them. Whether or not users could actually tailor this sort of panel remains to be seen. The Mac OS Launcher, for instance, allows for drag-and-drop installation, the simplest sort of interface technique available, but even this many users fail to master.

# 5 Final discussion

This report has discussed overall functionality, interface, architecture and platform requirements to a toolset in support of transcription, annotation, information extraction and analysis of natural and multimodal interaction (NIMM) data. The report builds on input from ISLE deliverable D11.1 (Dybkjær et al. 2001), the authors' joint experience from developing the MATE workbench (http://mate.nis.sdu.dk), and ongoing work in ISLE WP8 and WP9 on natural and multimodal interaction data resources and coding schemes, respectively. The report will serve as input to the NITE project (http://nite.nis.sdu.dk) which started in April 2001.

We have proposed the MATE workbench as a starting point for a NIMM toolset. The workbench has various shortcomings as mentioned above. Among these are

- Bugs: The workbench has a number of bugs which makes it unstable.
- Speed: The workbench is slow.
- Stylesheets: Too few style sheets are provided.
- GUI: The interface is inadequate in a number of respects. It is e.g. difficult for users to add their own coding scheme or even just to add a new dialogue for which they want to use an existing coding scheme. The query interface is difficult to understand and the representation of query results is in XML. It is difficult for the user to define new visualisations because this requires the user to write a new style sheet.

To bring the MATE workbench into a better position to serve as a point of departure for a NIMM toolset several improvements have been made to the software as part of ISLE WP11, addressing several of the points on the list of shortcomings. The result of this work is ISLE deliverable D11.3 which is the improved version of the MATE workbench. The most recent version of the MATE workbench can be downloaded from the MATE web site (http://mate.nis.sdu.dk).

The improvements which have been made include:

- lots of bug fixes;
- some documentation of MATE's bugs and limitations, which can be found at http://www.cogsci.ed.ac.uk/~jeanc/STYLESHEET-TIPS
- making the workbench faster;
- writing a communication problems style sheet including coding action;
- writing a coreference style sheet;
- writing some additional well-documented style sheets outside of the MATE level. These have to do with working with "operator" dialogues;
- expanding the set of actions available for coding style sheets; making the language for expressing actions in style sheets easier to read and write;
- making it easier for the user to add a new dialogue and use an existing coding scheme;
- improving the interface as regards file icons;
- adding an export function so that Transcriber (http://www.etca.fr/CTA/gip/Projets/Transcriber/) can be used for transcription of dialogues which can then be exported to MATE format for further annotation in the workbench.

Some task which haven't been done in ISLE but which are still on the to-do list include:

- finishing a half-done prosody display stylesheet;
- adding a pause button to the AudioTool plus some way to play from someplace besides the beginning;
- fixing the x and y scales allowed to be more sensible for long sound files;

- setting up a number of sound manipulation actions for style sheets beyond just play a given segment;
- user documentation (which is currently quite patchy);

HCRC has been using the workbench to do something akin to move coding on a UK internal project where the dialogues are quite short (average of 10 turns), and the workbench now works for this in practice, with a little bit of nurture (it is a bit slow starting up for longer dialogues, although they haven't been using the sped up version, and human coders sometimes want to press buttons faster than they respond. Also, is has been necessary to quit and restart every five dialogues or so due to a memory leak.

Although the MATE workbench could still be improved in several ways we believe that the above improvements and experience from use in another project has brought us into a much better position to use the MATE workbench as a point of departure for a NIMM annotation tool. The MATE workbench does not support annotation of video files but as we have argued above it should be possible to extend the workbench to also support NIMM annotation. This will be further explored in the NITE project.

# Acknowledgements

We gratefully acknowledge the support of the ISLE project by the European Commission's HLT Programme.

# 6 References


Abercrombie, D: *Elements of general phonetics*. Edinburgh: Edinburgh University Press, 1967.

Bernsen, N. O.: Multimodality in language and speech systems - from theory to design support tool. Chapter to appear in Granström, B. (Ed.): *Multimodality in Language and Speech Systems*. Dordrecht: Kluwer Academic Publishers, 2001. See http://www.nis.sdu.dk/~nobBird, S., Buneman, P. and Tan, W.: Towards a query language for annotation graphs. Proceedings of the Second International Conference on Language Resources and Evaluation, 2000, 807-814. http://morph.ldc.upenn.edu/sb/home/publications.html#lrec00-query, ftp://ftp.cis.upenn.edu/pub/sb/papers/lrec00-query/lrec00-query.ps.

Brugman, H., Russel, A., Broeder, D. and Wittenburg, P.: EUDICO, Annotation and Exploitation of Multi Media Corpora. Proceedings of the LREC 2000 Pre-Conference Workshop on Data Architectures and Software Support for Large Corpora, ELRA, Paris, 2000, 6-9. http://www.mpi.nl/world/ISLE/documents/papers/anno-Workshop-after.pdf.

Cohn, J.F., Zlochower, A.J., Lien, J., and Kanade, T.: Automated face analysis by feature point tracking has high concurrent validity with manual FACS coding. Psychophysiology 36(1), 1999, 35-43.

Crystal, D.: *Prosodic Systems and Intonation in English*. Cambridge: Cambridge University Press (Cambridge Studies in Linguistics, 1), 1969.

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 NIMM Deliverable D11.1, January 2001.

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

Edwards, J.A.: Principles and Contrasting Systems of Discourse Transcription. In Edwards, J.A.-Lampert, M.D. (Eds) *Talking Data: Transcription and Coding in Discourse Research*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1993, 3-32.

Evert, S., Berman, S. and Heid, U.: Querying IMDI Meta Descriptions. IMDI Workshop, MPI Nijmegen, March, 2001.

Fant, L.: *Estructura informativa en español. Estudio sintáctico y entonativo*. Upsala: Acta Universitatis Upsaliensis, 1984.

Gibbon, D., Moore, R., and Winski, R. (Eds.): *Spoken Language Systems and Corpus Design*. Berlin: Mouton De Gruyter. (Handbook of Standards and Resources for Spoken Language Systems, Volume I), 1998.

Gibbon, D.: S*urvey of Prosodic Labelling for EC Languages*. SAM-UBI-1/90, 12 February 1989; Report e.6, in ESPRIT 2589 (SAM) Interim Report, Year 1. Ref. SAM-UCL G002. University College London, February 1990.

Grønnum Thorsen, N.: Suprasegmental transcription. *ARIPUC - Annual Report of the Institute of Phonetics University of Copenhagen* 21, 1987, 1-27.

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.

Isard, A., McKelvie, D., Mengel, A., Møller, M. B., Grosse, M. and Olsen, M. V.: *Data Structures and APIs for the MATE Workbench*. MATE Deliverable D3.2, 2000.

Johnson, J.: GUI Bloopers. Don'ts and Do's for Software Developers and Web Designers. Morgan Kaufmann, Academic Press, 2000.

Klein, M., Bernsen, N. O., Davies, S., Dybkjær, L., Garrido, J., Kasch, H., Mengel, A., Pirrelli, V., Poesio, M., Quazza, S. and Soria, S.: Supported Coding Schemes. MATE Deliverable D1.1, July 1998. http://mate.nis.sdu.dk/about/D1.1/

Leech, G., Myers, G., and Thomas, J. (Eds): *Spoken English on Computer: Transcription, Markup and Applications*. Harlow: Longman, 1995.

Lehiste, I. and Peterson, G. E.: Some basic considerations in the analysis of intonation. *Journal of the Acoustical Society of America* 33, 4, 1961, 419-425.

Llisterri, J.: *Prosody Encoding Survey*. WP 1 Specifications and Standards. T1.5. Markup Specifications. Deliverable 1.5.3. Final version, 15 September 1994. LRE Project 62-050 MULTEXT, 1994.

Llisterri, J.: *Preliminary Recommendations on Spoken Texts*. EAGLES Document EAG-TCWG-STP/P, May 1996. http://www.ilc.pi.cnr.it/EAGLES96/spokentx/spokentx.html

López Gonzalo, E.: *Técnicas de procesado lingüístico-prosódico y acústico para conversión texto-voz mediante concatenación de unidades*. PhD Thesis. E.T.S. I. de Telecomunicación, Universidad Politécnica de Madrid, 1993. ftp://ftp.gaps.ssr.upm.es/pub/tts/tesis/TesisEdu.ps.tar.Z

Mengel, A., Heid, U., Fitschen, A. and Evert, S.: Improved Query Language (Q4M). In Isard et al. 1998.

Navarro Tomás, T.: *Manual de entonación española*. New York: Hispanic Institute. Cuarta edición: Madrid: Guadarrama (Punto Omega, 175), 1944.

Quazza, S. and Garrido, J.M.: Prosody. In Klein et al. 1998. http://technovoice.cselt.it/voce/chap6/chap6.html, http://www.ims.uni-stuttgart.de/projekte/mate/mdag/pd/pd_1.html

Quilis, A.: *Fonética acústica de la lengua española*. Madrid: Gredos (Biblioteca Románica Hispánica, Manuales, 49), 1981.

Quilis, A.: *Tratado de Fonología y Fonética españolas*. Madrid: Gredos (Biblioteca Románica Hispánica, Manuales, 74), 1993.

Schlieder, T. and Meuss, H.: Result Ranking for Structured Queries Against XML Documents. First DELOS Workshop on Information Seeking, Querying and Searching in Digital Libraries, 2000. http://www.cis.uni-muenchen.de/~meuss/DELOS00.ps.gz.

Sperberg-McQueen C. M. and Burnard, L. (Eds.): Guidelines for Electronic Text Encoding and Interchange. TEI P3. Chapter 11: Transcriptions of Speech. Association for Computational Linguistics - Association for Computers and the Humanities – Association for Literary and Linguistic Computing: Chicago and Oxford, 1994. http://etext.lib.virginia.edu/bin/tei-tocs?div=DIV1&id=TS

Suciu, D.: Semistructured Data and XML. *Proceedings of International Conference on Foundations of Data Organization*, Kobe, Japan, citeseer.nj.nec.com/suciu98semistructured.html, 1998.

Terken, J. and Nooteboom, S.: Opposite effects of accentuation and deaccentuation on verification latencies for Given and New information, *Language and Cognitive Processes (2)*, 1987, 145-163.

Thorsen, N.: On the identification of selected Danish intonation contours. *ARIPUC - Annual Report of the Institute of Phonetics, Copenhagen University* 12, 1978, 17-74.

Van Donselaar, W: The function of Prosody in speech perception. *Proceedings of the XIIth International Congress of Phonetic Sciences*, Aix-En-Provence, Vol. 3, 1991, 466-469.

W3C: XML Query Requirements (Working Draft). 15 February, 2001. http://www.w3.org/TR/xmlquery-req.

Williams, B.: The formulation of an intonation transcription system for British English. In Knowles, G., Wichmann, A.and Alderson, P. (Eds.): *Working with Speech: Perspectives on research into the Lancaster/IBM Spoken English Corpus*. London & New York: Longman, 1996, 38-58.

Zhong, Y.: Apply Multimodal Search and Relevance Feedback In a Digital Video Library. Carnegie Mellon University, School of Computer Science, 2000. http://www.informedia.cs.cmu.edu/documents/zhong_thesis_may00.pdf.