Project ref. no.	IST-2000-26095
Project title	NITE: Natural Interactivity Tools Engineering

Deliverable status	Restricted
Contractual date of delivery	28 June 2002
Actual date of delivery	5 September 2002
Deliverable number	D1.1 Addendum
Deliverable title	Lists of Requirements
Type	Report
Status & version	Draft version 1
Number of pages	18
WP contributing to the deliverable	WP1
WP / Task responsible	Jean Carletta, University of Edinburgh
Author(s)	Niels Ole Bernsen, Niels Cadée, Jean Carletta, Laila Dybkjær, Stefan Evert, Ulrich Heid, Amy Isard, Mykola Kolodnytsky, Christoph Lauer, Wolfgang Lezius, Lucas Noldus, Norbert Reithinger, and Andreas Vögele
EC Project Officer	Brian Macklin
Keywords	Tools specification, functionality, natural interactivity and multimodality data, annotation support, tools evaluation
Abstract (for dissemination)	This report is an addendum to NITE deliverable D1.1, *Software Specification and Workplan*, from December 2001 [Carletta et al. 2001]. The report presents tool goals, succinct lists of requirements, and evaluation criteria for the three development strands in the NITE project, i.e. the NITE WorkBench for Windows, or NWB, the NITE XML Toolkit, or NXT, and the Noldus Observer.

Lists of Requirements

Addendum to NITE D1.1

22 August 2002
DRAFT version 1

This report is an addendum to NITE deliverable D1.1, *Software Specification and Workplan*, from December 2001 [Carletta et al. 2001]. The report presents tool goals, succinct lists of requirements, and evaluation criteria for the three development strands in the NITE project, i.e. the NITE WorkBench for Windows, or NWB, the NITE XML Toolkit, or NXT, and the Noldus Observer.

Authors

Niels Ole Bernsen[2], Niels Cadée [3], Jean Carletta[1], Laila Dybkjær[2], Stefan Evert[4], Ulrich Heid[4], Amy Isard[1], Mykola Kolodnytsky[2], Christoph Lauer[5], Wolfgang Lezius[4], Lucas Noldus [3], Norbert Reithinger[5], and Andreas Vögele[4].

1: HCRC, Edinburgh, UK. 2: NISLab, Odense University, Denmark. 3: Noldus Information Technology bv, Wageningen, The Netherlands. 4: IMS, Stuttgart University, Germany. 5: DFKI, Saarbrücken, Germany.

Section responsibilities

Section 1	Introduction	Niels Ole Bernsen
Section 2	NITE Workbench for Windows (NWB)	Niels Ole Bernsen, Laila Dybkjær, Mykola Kolodnytsky
Section 3	NITE XML Toolkit (NXT)	Jean Carletta, Stefan Evert, Ulrich Heid, Amy Isard, Christoph Lauer, Wolfgang Lezius, Norbert Reithinger, Andreas Vögele
Section 4	Development on The Observer relevant to the NITE project	Niels Cadée and Lucas Noldus

All authors have approved the final deliverable.

Contents

1. Introduction

The goal of the NITE (Natural Interactivity Tools Engineering) project is to develop software in the form of a tool, or a toolset, for the annotation and analysis of natural interactive communication between humans as well as between humans and machines. Natural interactive communication makes use of some or all the communication modalities used by humans when they communicate with one another, including speech, facial expression, gesture, gaze, bodily posture, and the handling of objects as an integral part of the communication.

This report is an addendum to NITE deliverable D1.1, *Software Specification and Workplan,* from December 2001 [Carletta et al. 2001]. NITE D1.1 presents the general requirements specifications to the NITE software, specifications which are largely shared among all NITE partners. The present deliverable takes into account the fact that NITE software development is distributed across three different development strands. One strand is the development of the NITE WorkBench for Windows, or NWB, aimed at users who are not experts in the XML world. A second strand is the development of the NITE XML Toolkit, or NXT, aimed at users who are experts in the XML world. The third strand is the preparation of future releases of the Noldus Observer tool which increasingly will include functionality for enabling annotation and analysis of natural interactive communication. For each of these strands, this document presents the following information:

1. the goal of the tool;
2. a succinct list of requirements on functionality, usability, technical quality, etc.;
3. a specification of architecture and data flow (optional);
4. a set of evaluation criteria.

NWB requirements and evaluation criteria are presented in Section 2, NXT requirements and evaluation criteria in Section 3, and Noldus Observer requirements and evaluation criteria in Section 4. The Appendix provides some useful references to the XML world.

2. NITE Workbench for Windows (NWB)

2.1 Introduction

The NITE Workbench for Windows (NWB) will enable the many future coders of natural interactive behaviour who are not XML experts to work with recorded audio and video data from human-human and human-system interaction. With the NWB, these users will be able to (i) transcribe and annotate their multimodal data at multiple levels of abstraction as well as across abstraction levels, in principle without encountering any obstacles in the form of limitations to the tool's annotation capabilities. They will so either by using a pre-defined coding scheme offered by the tool and entered into the tool by a previous user or by the NWB team, or one which (ii) they have built themselves using the tool's facilities for specifying new coding schemes, and subsequently (iii) analyse the annotated file using advanced information extraction mechanisms. It is obvious that the targeted user group, and the general-purpose natural interactivity coding aims to be met by the NWB, makes it mandatory to put particular emphasis on ease of use of the tool during all phases of its use, from the entry of a new coding scheme through to coding, information extraction, and analysis. This, again, enforces strong requirements to the tool's interface.

2.2 A brief use case

As an example of the complexity involved in natural interactivity coding, consider the following use case which has been applied as a driver of development in NITE. In this use case, the aim is to investigate negotiation of initiative during human-human dialogue in all its aspects, including turn-taking cues provided through speech, facial expression and gaze, gesture, and bodily posture. As the following perfunctory analysis shows, the complexity involved is rather high. The coder will, or may, need to do annotation wrt.:

- speech transcription **(coding scheme 1)**
- speech dialogue acts – these include whatever semantics are involved as well as "speech-like" cues **(coding scheme 2)**
- facial dialogue acts **(coding scheme 3)**
- gesture dialogue acts **(coding scheme 4)**
- body posture dialogue acts **(coding scheme 5)**
- emotions and moods revealed **(coding scheme 6)**
- cross-linking of tagged phenomena at different levels and in different modalities suggesting a particular global act of seizing, or trying to seize, initiative **(coding scheme 7)**
- longitudinal linking of tagged phenomena (same level, same modality) which constitute parts of a single effort to seize initiative at that level **(coding scheme 8)**
- speech syntax codes might come in handy for analysing linguistic peculiarities of initiative-grabbing efforts **(yet another coding scheme)**
- whatever else might be felt needed in the scientific exploration process, such as facial or gaze mini-cues if these turn out to be worth investigating in the context **(another coding scheme)**, or prosody **(yes, another one)**
- depending on the format of representation chosen, one would perhaps have to multiply the above by two or three because there are several interlocutors

2.3 Requirements specification

The NWB requirements specification below has been developed following the ideas shown in Figure 1. More details on the NWB requirements specification are presented in [Bernsen et al. 2002].

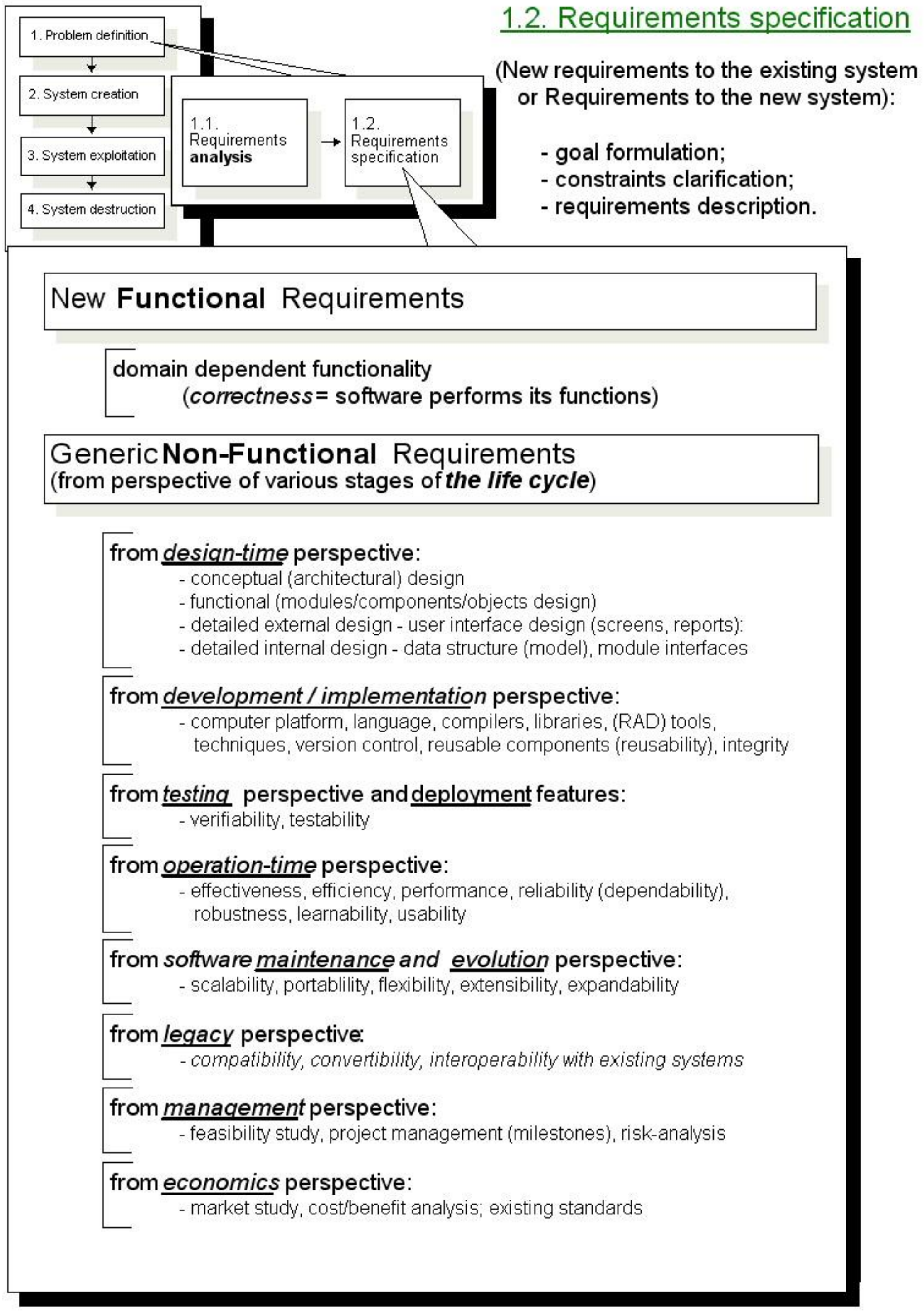

1.2. Requirements specification
(New requirements to the existing system or Requirements to the new system):
- goal formulation;
- constraints clarification;
- requirements description.
1. Problem definition
2. System creation
3. System exploitation
4. System destruction
1.1. Requirements analysis
1.2. Requirements specification
New Functional Requirements
domain dependent functionality
(correctness = software performs its functions)
Generic Non-Functional Requirements
(from perspective of various stages of the life cycle)
from design-time perspective:
- conceptual (architectural) design
- functional (modules/components/objects design)
- detailed external design - user interface design (screens, reports):
- detailed internal design - data structure (model), module interfaces
from development / implementation perspective:
- computer platform, language, compilers, libraries, (RAD) tools, techniques, version control, reusable components (reusability), integrity
from testing perspective and deployment features:
- verifiability, testability
from operation-time perspective:
- effectiveness, efficiency, performance, reliability (dependability), robustness, learnability, usability
from software maintenance and evolution perspective:
- scalability, portablility, flexibility, extensibility, expandability
from legacy perspective:
- compatibility, convertibility, interoperability with existing systems
from management perspective:
- feasibility study, project management (milestones), risk-analysis
from economics perspective:
- market study, cost/benefit analysis; existing standards

Figure 1. An approach to requirements specification.

2.3.1 Goal formulation

The main objective of NWB development is to build an integrated best practice workbench for multi-level, cross-level and cross-modality annotation, retrieval and exploitation of multi-party natural interactive human-human and human-machine dialogue data.

2.3.2 Requirements description: functional requirements

The core functional requirements to the tool can be combined into six following groups:

1. to create annotated corpus files;
2. to control raw data (i.e. audio-video files);
3. to enable users to specify coding schemes;
4. to support annotation using coding schemes;
5. to visualise information in a customized way;
6. to enable information extraction and analysis of annotated data.

Each group of requirements has been specified in detail below.

2.3.2.1 Creating annotated corpus files

To create an annotated corpus file means to create a project which includes:

- metadata information (to be specified);
- names of (or links to) raw data files, i.e. audio and video data files, log-files;
- information about the coding schemes used for annotation (transcription);
- file(s) containing actual coding information;
- possibly, files with the results of analysis applied to the coding files.

It should be possible to:

- create a new project from scratch;
- open / save a project;
- print components of a project;
- import/export components of a project into/from XML files.

2.3.2.2 Controlling raw data (i.e. audio-video files)

To control audio-video files means to:

- visualise and play video files created in current standard formats, such as *.mpg, *.avi, etc. A list of supported formats should be specified;
- listen to the audio track created in current standard formats, such as *.mp3, *.wav, etc. A list of supported formats should be specified;
- display a log-file in plain text with information about, for instance, machine events that occurred during human-machine communication;
- graphically represent the acoustic information:
  - wave-form (i.e. in the time-domain);
  - spectrum (i.e. in the frequency-domain);
- switch on/off the raw data windows;
- have a visible timeline;
- navigate back and forth in the raw data based on the timeline, i.e. scroll, go step-by-step along the timeline, jump into the beginning or the end of the timeline);
- zoom in/out on the time-line;
- display the value of timeline units, such as seconds, frame numbers for video data, or milliseconds for both video and audio data;
- open several video raw data windows at the same time;
- control the audio track and the video track independently of one another;

- synchronise the playing or displaying of data in different windows on the basis of the common timeline.

2.3.2.3 Enabling users to specify coding schemes

A *coding scheme* is a set of elements for marking up some class of phenomena in linguistics, communication or behaviour studies. A particular class of phenomena which may be conceptualised in different ways in different coding schemes is sometimes called a *level of annotation*.

To specify coding schemes means to:

- create new annotation schemes for any class of phenomena in spoken dialogue, facial expression, emotion, gaze, gestures of all kinds, lip movements, bodily posture, actions, etc.;
- use or modify existing annotation schemes;
- perform an orthographic transcription and possibly a phonetic transcription of the acoustics;
- be able to add free-form comments.

A comprehensive list of existing coding schemes for different classes of phenomena which could be supported by the NITE WorkBench has been investigated during requirements specification. The list includes coding schemes for the following annotation levels: orthographic transcription, phonetic transcription, (morpho-) syntactic annotation, prosody annotation, semantic annotation, co-references annotation, discourse annotation, dialogue acts annotation, communication problems annotation, gesture annotation, body annotation, emotion and mood annotation, behaviour annotation. The list also includes a few multi-level or cross-level coding schemes.

In order to specify a coding scheme for use in the NITE WorkBench, the coding scheme should be represented hierarchically as a set of tables of phenomena. An example of such a representation for prosody annotation – the ToBi coding scheme – is:

I. **The 1st level** of categories (phenomena) hierarchy

No	Name	(Tag) Visualisation	Notes
1	Prosodic boundaries		See table 2.1
2	Prosodic phenomena		See table 2.2
3	Downstep		

Table 1. the 1st level (table of *categories* and value)

II. **The 2nd level** of categories (phenomena) hierarchy

No	Name	(Tag) Visualisation	Notes
1	clitic group boundary	0	
2	word boundary	1	
3	boundary with no tonal mark	2	
4	intermediate Phrase boundary	3	
5	intonative Phrase boundary	4	

Table 2.1. Prosodic boundaries (table of values)

No	Name	(Tag) Visualisation	Notes
1	pitch accents		See table 3.1
2	boundary tones		See table 3.2
3	phrase accents		See table 3.3

Table 2.2. Prosodic phenomena (table of categories)

III. **The 3rd level** of categories (phenomena) hierarchy

No	Name	(Tag) Visualisation	Notes
1	peak accent (high pitch accent)	H*	
2	low accent (low pitch accent)	L*	
3	scooped accent	L*+H	
4	rising peak accent	L+H*.	
5	downstepped accent	H+!H*	

Table 3.1. Pitch accents (table of *values*)

No	Name	(Tag) Visualisation	Notes
1	final low boundary tone	L%	
2	final high boundary tone	H%	
3	initial high boundary tone	%H	

Table 3.2. Boundary tones (table of *values*)

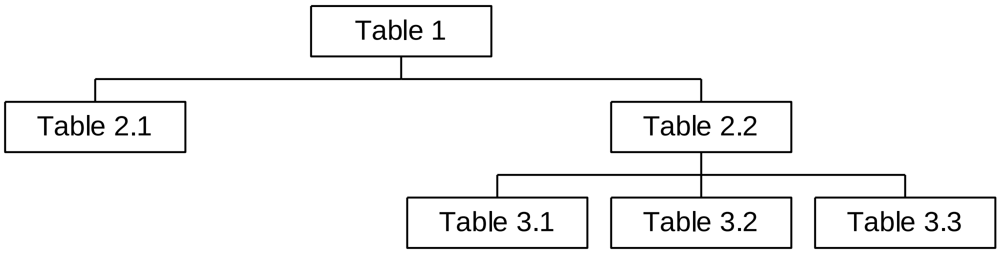
No	Name	(Tag) Visualisation	Notes
1	low phase accent	L-	
2	high phase accent	H-	

Table 3.3. Phrase accents (table of *values*)

IV. The relationships between the tables are:

Table 2.1. – Table 1., entry 1;
Table 2.2. – Table 1., entry 2;
Table 3.1. – Table 2.2., entry 1;
Table 3.2. – Table 2.2., entry 2;
Table 3.3. – Table 2.2., entry 3;

V. Graph (tree) of coding scheme structure



As a matter of fact, coding schemes may contain two different type of information:

- *permanent*, raw data-independent information, and
- *variable*, raw data dependent information.

For instance, in the case of the ToBi coding scheme, every category and all values are permanent, raw data-*independent*. In the case of, for instance, the Transcriber coding scheme, some values, such as "speakers", are raw data-*dependent*, while some others, such as "type of speaker" (male, female, unknown) or "type of dialect" (native, non-native), etc., are raw data-*independent*.

2.3.2.4 Supporting annotation using coding schemes

To support annotation using coding schemes means to:

- to use a special-purpose (i.e. special format) file created as part of a project and containing the coding information. We call this file a coding file (or mark up file). The coding file has to reflect the timeline of events presented in the raw data. We call it "the common timeline" or merely "the timeline";
- *edit* the coding file:
  - *insert* a tag from the coding scheme:
    - select (indicate) the time point or time interval onto the timeline;
    - choose an appropriate coding scheme;
    - choose a tag from list of tags for the coding scheme presented on a palette;
    - insert the tag onto the selected time point or time interval;
  - *delete* a tag from the timeline.
- visualise the result of the coding (see next section).

2.3.2.5 Customised information visualisation

Customisation of information visualisation means to:

- provide a window or windows (or its part like a split pane) for displaying the contents of a coding file. Let us call such a window an "annotation panel";
- display several annotation panels at the same time;
- control the appearance of a panel on the screen, i.e. to show, hide, move, resize it, and probably change some attributes of it like the title, background colour, etc.;
- each annotation panel should correspond to a certain class of phenomena to be coded;
- probably, one annotation panel may contain several classes of phenomena at the same time;
- since a class of phenomena could be a hierarchy of categories and values, it should be possible to reflect this structure using a set of "tiers" (or "bands", or " layers"), each of which aims to display a particular subset;

- show the cursor at the current position on the timeline;
- synchronise the cursors in different windows: as soon as the cursor moves in one window, it moves in the second one and appears within the same timeline segment;
- view annotations and transcriptions at different levels of resolution in each annotation panel;
    - zoom in/out along timeline;
    - scroll along timeline;

Visualisation and customisation of the tags means:

- to visualise the result of the coding, i.e. to provide various way of displaying the tags inserted onto the timeline within the annotation panel.
    - the visualisation could be done using:
        - a special-purpose window (in this context called an annotation panel), and/or;
        - directly on the window displaying the raw data (e.g. graphical video mark up, GVM or on the wave-form of audio data (or elsewhere));
    - the visualised and customised tags should be inserted onto the timeline using the annotation panel.

In addition:

- available tag palettes must always correspond to the items in the selected annotation panel;
- it should be possible to link several annotation levels during annotation;
- it should be possible to insert (add) free-form comments as a tag;
- it should be possible to customise the visualisation of tags;
- annotation levels (including orthographic transcription) should visibly share a common timeline;
- while playing a video or audio file, the common timeline must visibly “run through” all annotations;
- it should be possible to handle, in some way, the situation when the links between two different levels of annotation may cause a certain clutter on the screen.

2.3.2.6 Enabling information extraction and analysis of annotated data

To enable information extraction and analysis of annotated data means to:

- extract arbitrary parts of the data;
- build statistical descriptions of the data;
- analyse data using inferential statistics.

For more detail, see the Observer 4 implementation as an example of a query task.

2.3.3 Requirements description: non-functional requirements

1. A flexible and open architecture which allows for easy addition of new tool components (a modular workbench).

2. Separation of user interface from application logic and internal data representation.

3. The tool must be robust, stable, and work in real time even with relatively large data resources and complex coding tasks.

2.3.4 Constraints clarification

It must be possible:

- to show up to ten annotation levels simultaneously on the screen.
- full audio filename with path: 100 chars.

2.4 Evaluation criteria

The following evaluation criteria for the NITE WorkBench are based on internal project discussion held in preparation of the NITE presentations and demonstrations to be made at LREC'2002. The discussion was initiated by the NITE evaluation group consisting of Vito Pirelli, Claudia Soria and Uli Heid.

The main test parameters are:

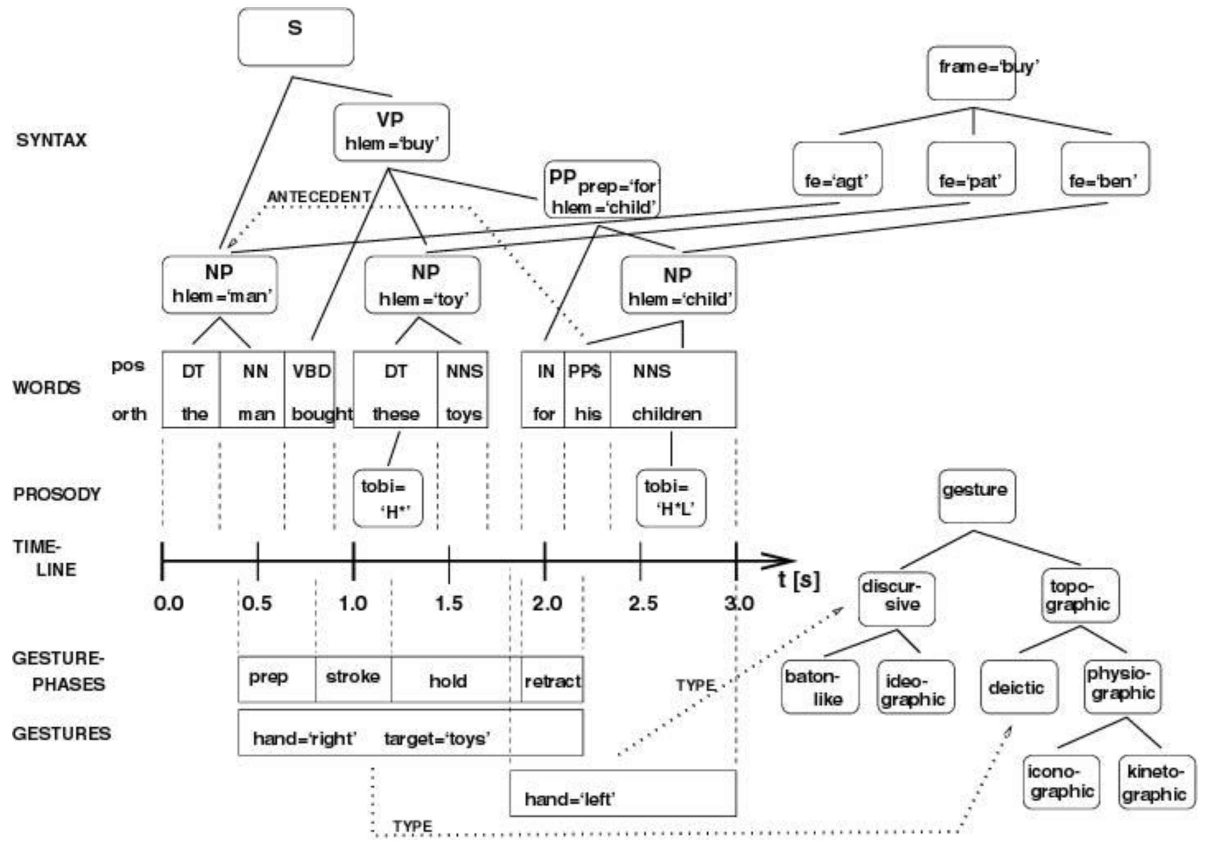
- usability
- usefulness
- functionality
- technical quality

Test users of the NITE WorkBench will be asked the following evaluation framework questions, some of which may become further elaborated into sub-questions, depending on the nature of the test users and the occasion of the test.

- **Overall Perception**
    - o Do users like the tool or not. For which reason(s)?
- **Usability**
    - o Do users find the tool intuitive and easy to use?
    - o Are common tasks easily identified?
- **Usefulness**
    - o Do users feel their work would benefit from using the tool?**Functionality**
    - o Is NWB capable of supporting the full range of natural interactivity coding tasks it aims to support? Which coding task(s) cannot be done using the NWB (if any)?
    - o Are users satisfied with existing features, or would they like to see something added/eliminated/changed?
    - o Are basic functionalities shaped according to common standards?
    - o Do users think that the sort of flexibility that the tool accommodates is a useful advance on current functionality?
- Technical quality
    - o Is the NWB robust and stable?
    - o Does the NWB work in real time even with relatively large data resources and complex coding tasks?
- **Overall design**
    - o According to their experience, do evaluators think that the tool satisfies the purposes for which it was created?
    - o Do evaluators agree with the basic concept underlying the tools' development?

3. NITE XML Toolkit (NXT)

Users of multimedia language data face two problems at the moment: they can't share data because different tools use different data formats, and they can't mark up data with annotations that have complex structures, or combine two simple-structured annotations on the same data, because the data representations employed by the tools at best allow for trees. Figure 1 gives a quite simple constructed example of the richness of structure needed for multimedia annotation. The HCRC Map Task Corpus and the Switchboard Corpus, available from TalkBank, are examples of non-multimedia corpora which have run into the analogous difficulty about representation of structure, but for speech and language annotation without video. Such heavily annotated corpora do not exist yet for multimedia data because of the lack of tools, but are needed for work in, e.g., animation and human-computer interfaces.

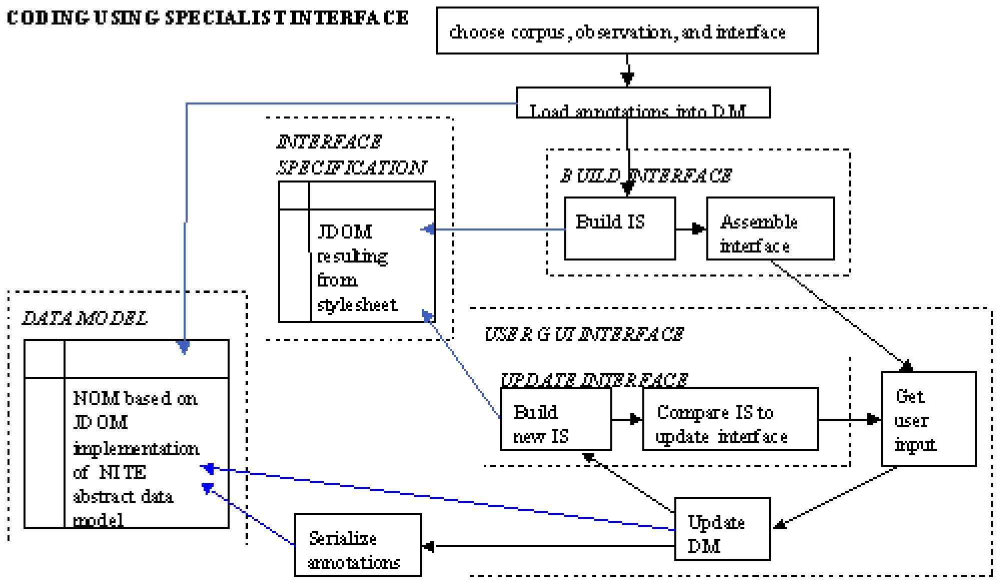


End users of a data set want tools that they can just start up and run for displaying, annotating, and analysing data. This creates a basic tension. Where the annotations are simple in structure, it's possible to write general-purpose tools that have reasonable data displays and interfaces (such as Anvil or The Observer). These tools are fine when the structure of the annotation needed fits the model they have in mind. However, the more exotic the structure needed, or the more annotations given on the same data, the less likely that a pre-defined general interface will fit user needs. This is why in speech and language annotation, most corpus projects ensure that they keep someone around who has technical skills and can dedicate time to making one-off tools for each task that needs to be done with their specific data set.

Other strands of the NITE project are building general purpose tools to suit the common denominator users and which require no skilled technician to intervene between data and end user. The NITE XML Toolkit is instead aimed at the skilled user in the project and will allow him or her to build the more specialized displays, interfaces, and analyses that are required by end users when working with highly structured or cross-annotated data. We are not alone in taking this approach. The American Annotation Graph Toolkit is also aimed at the skilled user supporting a wider group. However, they currently only intend to provide library routines for the loading and manipulation of annotations, leaving the skilled user to call these from a programming language and giving no help in building displays and interfaces. We intend to provide more library support for the skilled user, as well as a stylesheet engine that allows him or her to specify many types of tools declaratively, based on ideas developed originally by the MATE project and discussed in [McKelvie et al. 2000] and [Carletta et al., to appear].

The NITE XML Toolkit is intended to provide:

- an abstract data model which is rich enough to allow even a structurally complex set of multi-media annotations to be mapped into it
- a canonical data storage format using stand-off XML, including a schema format which can be used optionally for data validation
- Java library routines for loading data in the storage format into the model and for serializing from the model to the storage format
- Java library routines for manipulating data in the model
- an implemented query language for working with the data loaded into the model
- Java library classes that can be used to build data displays and interfaces
- a stylesheet-based annotation engine in Java for combining these elements into working display and annotation tools, with the architecture shown in figure 2
- a query engine in Java for the simple execution of queries expressed in our language
- a data visualization engine in Java displaying both the timeline of the audio signal and the video stream, if available.
- Java library routines to annotate the video signal with markers which can be linked to e.g. the gesture track


CODING USING SPECIALIST INTERFACE
choose corpus, observation, and interface
Load annotations into DM
INTERFACE SPECIFICATION
JDOM resulting from stylesheet
BUILD INTERFACE
Build IS
Assemble interface
DATA MODEL
NOM based on JDOM implementation of NITE abstract data model
USER GUI INTERFACE
UPDATE INTERFACE
Build new IS
Compare IS to update interface
Get user input
Update DM
Serialize annotations

There are three ways in which skilled users will be able to build tools using the toolkit. First, as with the Annotation Graph Toolkit, programmers can call on our library routines from their own programs. Second, they will be able to call our data model and query language from within a stylesheet in order to perform data transduction tasks such as export and data extraction. Third, they will be able to use the stylesheet annotation engine as the program, so that rather than writing an annotation tool that calls our data model and interface library classes, they can write a stylesheet that specifies declaratively how to build the tool, with the engine building it for them. We wish our toolkit to be as flexible as possible. For this reason, we intend to make it possible, for instance, to employ other data models with other query languages (such as JDOM with XPATH, or those associated with annotation graphs) from within our stylesheet annotation engine.

It is in the nature of the work that this skilled user will already be trained in XML and understand standard ways of manipulating XML such as stylesheets. To use the full power of the NITE XML Toolkit, they will have to learn in addition:

- Our schemas for describing the structure of a set of annotations, which is a use of the XML Schema standard (but the standard may not be familiar to those who have used DTDs instead. However, the use of schemas is optional within the Toolkit).
- Our syntax for the stand-off links that make richly structured data possible in XML, which is a cut-down version of the standard (but the standard may not be familiar to those who have only used single documents before).
- Our data model and query language for working with the data model; standards such as JDOM and XPATH could be used but are not a good match to non-tree-structured data.
- Additionally, to build displays and interfaces using the stylesheet annotation engine, the set of GUI display objects and interactions that users can have with them and the syntax for declaratively specifying GUI elements and actions from the stylesheet.

This remit is obviously ambitious, but the rewards are great; without this approach, we risk users restricting themselves to what the more general purpose tools will do. Scientific breakthroughs come with a change in instrumentation. Even if we only succeed partially (for instance, with a stylesheet annotation engine that only works with simpler data models, or a richer data model that works for data extraction and querying but not fast enough for annotation), we will still have made an important contribution in terms of functionality that other tools do not provide. Because our toolkit is based on XML processing and other tools are taking up XML at least as an import and export format, users will be able to mix use of other tools and our toolkit as best suits their needs.

Possible evaluation criteria for the success of the different elements in this strand of the NITE project include determining:

- how well the data model and query language fill the needs of users for multimodal corpora.
- whether the skilled technician is able to understand the data storage format and to validate data.
- that the loading and serialization functions work correctly and that the data model API handles the data properly.
- that the query language implementation correctly evaluates query expressions.
- that the stylesheet engine be able to render data for display purposes (by provision of examples).
- that the stylesheet engine be able to render tools that can be used for annotation (by provision of examples).
- whether users can effectively explore a corpus using the query engine.

4. Development on The Observer relevant to the NITE project

4.1 Goal

Versatile Observation System (VOS) is the working title for a major upgrade of The Observer. Currently we are defining global requirements for VOS. At the end of the NITE project, development on VOS will be well under way. VOS will be released later than the end of the NITE project. The possibility remains open that VOS will eventually be called The Observer rather than VOS.

VOS may support some high-priority NITE requirements, but it will not be a full-scale NITE tool. If the NITE functionality catches on and many multimodality researchers buy the program, we may implement more NITE functionality in later releases of VOS.

4.2 Requirements

This section lists required functionality, with regard to the NITE project, for VOS. As VOS is a generic tool for behavioural observation and analysis, Noldus Information Technology needs to weigh these requirements against the needs of several other user communities (such as psychologists, usability testers and movement scientists). The requirements listed below are based on the priorities (fully listed in Section 4.3 of D1.1) expressed by the NITE partners present at the project meeting in Odense (July 2001) and the priorities defined by the Noldus marketing department. Part of the requirements on the list in section 4.3 have already been met in The Observer 4.1, which was released in June 2002.

Ideally, all items on the requirements list can be developed, but this is eventually dependent on available R&D capacity at Noldus and a more detailed cost-benefit analysis. As soon as the list of priorities for VOS has been finalized by the VOS development team at Noldus, development will start. The priorities have been discussed with the NITE partners in Saarbrücken (May 2002). Niels Cadée will specify them further and has started discussing with the VOS development team about which requirement to include in VOS. To further specify these requirements, Noldus can benefit from the development of the NITE for Windows development strand, and from requirement specifications written by other NITE partners as well.

These four requirements have the highest priority

- **XML import/export** — Exporting coding schemes and data files from The Observer to XML files, and importing XML data files and maybe also coding schemes into The Observer. One of our programmers is currently working on a prototype for this, drawing on the XML knowledge of the NITE partners.
- **Hierarchical configuration structure** — Allowing for more hierarchical complexity in coding schemes than is currently possible in The Observer 4.1. We still have to specify this requirement in more detail. We also have to find out how important links between coding elements are. This is not supported in our current software. Since NITE for Windows uses an Access database for specifying coding schemes like The Observer, we can benefit from their development.
- **Timeline display** — The Observer currently only has a vertical table to display coded data during annotation. The vertical table is useful for displaying dialogs with long texts. However, for gesture coding a horizontal timeline is more appropriate (like in the ANVIL software). In a horizontal display it is also easy to display the many tracks of multimodal annotations (speech, gestures, facial expressions, …). The need for a

horizontal timeline display is currently being investigated, also for other user groups outside the NITE domain.

- **Visualizing speech transcription and code names** — This requirement is linked to the previous one. In a horizontal timeline, coding elements can be shown as bars with a length corresponding to their duration. In these bars, the names of the elements, or the text the observed person said, can be displayed. Further specification of this requirement has to wait until we know if we are going to make a horizontal timeline display.

For the following two requirements, we hope to get more information from specifications for the software in the other NITE development strands. Depending on the details, and the need among NITE partners for these functions, we may include them in the VOS requirements specifications.

- **Record metadata** — Expand the existing independent variables to record more extensive metadata on projects and their contents. For example, data on the persons doing the coding, the age and gender of the subjects, the language, and any information about specific tasks the subjects were carrying out.
- **More advanced queries** — Support more advanced queries than currently possible, especially with hierarchical levels.

These two requirements have low priority:**Graphical mark-up of video** — This should allow the user to mark an area in the video image that deserves special attention, like a gesture or a specific facial expression. The marking should stay visible for the entire duration of the corresponding coding element. A prototype of graphical markup has been developed in the NITE XML Toolkit strand. If this JAVA implementation is easy to translate to C++, it may not take much time to implement this in VOS. We will look into this.

- **Customizable user interface** — Allow the user to switch between a "full" UI (for use by the expert) to a "restricted" UI (for use by a novice), where the “restricted” mode only shows a subset of the functionality. This will make the program easier to use for non-experts, e.g. students. At this stage it is not yet clear if different user profiles are necessary, and if so, how many, and what the feature sets of each profile should be.

4.3 Evaluation criteria

At the end of the NITE project, there will not yet be a release of VOS. But detailed requirements specifications and user interface designs will be available for review. We propose the following evaluation criteria for these materials:

- Do the requirements match the needs of the NITE partners and natural interactivity researchers in general?
- Are the requirements we choose from the list the most useful ones for natural interactivity research?
- Will annotating a multimodal corpus in VOS save researchers time compared to scoring with their own home grown application?
- Usability for natural interactivity research (as far as this can be judged from the designs and requirements).

5. Acknowledgements

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

6. References

Bernsen, N. O., Dybkjær, L., and Kolodnytsky, M.: An Interface for Annotating Natural Interactivity. In J. v. Kuppevelt and R. W. Smith (Eds.): *Current and New Directions in Discourse and Dialogue,* Dordrecht: Kluwer 2002 (to appear).

Carletta, J., Bernsen, N. O., Cadée N., Dybkjær, L., Evert S., Heid, U., Isard A., Kolodnytsky M., Lauer C., Lezius W., Noldus L., Reithinger N, and Vögele A. Software Specification and Work Plan. *NITE Deliverable D1.1,* December 2001.

Carletta, J., McKelvie, D., and Isard, A.: A generic approach to software support for linguistic annotation using XML. In G. Sampson and D. McCarthy (Eds.): *Readings in Corpus Linguistics*. London and New York: Continuum International, (to appear).

McKelvie et al.: The Mate Workbench: An Annotation Tool for XML Coded Speech Corpora, *Speech Communication* 33(1-2), 97-112, 2000.

Appendix: URLs for XML standards and software

dbXML Core 1.0b1, native XML database, http://www.dbxml.org/

Document Object Model (DOM) Level 2 Core Specification Version 1.0, W3C Recommendation 13 November 2000, http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113. Latest version: http://www.w3.org/TR/DOM-Level-2-Core.

Java APIs for XML Processing (JAXP) Release 1.1, http://java.sun.com/xml/xml_jaxp.html.

Kweelt XML Query Processor, http://db.cis.upenn.edu/Kweelt/. Currently, Kweelt Deux is under development.

Quilt: An XML Query Language, http://www.almaden.ibm.com/cs/people/chamberlin/quilt.html.

SAX 2.0: The Simple API for XML, http://www.megginson.com/SAX/index.html.

SAXON, The XSLT Processor (version 6.4.4), http://saxon.sourceforge.net/.

Xalan-Java (version 2.2.D10), http://xml.apache.org/xalan-j/.

Xerces2 Java Parser, http://xml.apache.org/xerces2-j/.

XML Linking Language (XLink) Version 1.0, W3C Recommendation 27 June 2001, http://www.w3.org/TR/2000/REC-xlink-20010627/. Latest version: http://www.w3.org/TR/xlink/.

Fujitsu XLink Processor (XLip), http://www.labs.fujitsu.com/free/xlip/en/. This is only available as part of a demo specification. It doesn't implement full XPointer syntax, just barenames (..#anchor) and “child sequences”.

Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006. Latest version: http://www.w3.org/TR/REC-xml.

XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/1999/REC-xpath-19991116. Latest version: http://www.w3.org/TR/xpath.

XPath Requirements Version 2.0, W3C Working Draft 14 February 2001, http://www.w3.org/TR/2001/WD-xpath20req-20010214. Latest version: http://www.w3.org/TR/xpath20req.

XML Pointer Language (XPointer) Version 1.0, W3C Candidate Recommendation 11 September 2001, http://www.w3.org/TR/2001/CR-xptr-20010911/. Latest version: http://www.w3.org/TR/xptr.

XML Query Use Cases, W3C Working Draft 08 June 2001, http://www.w3.org/TR/2001/WD-xmlquery-use-cases-20010608. Latest version: http://www.w3.org/TR/xmlquery-use-cases.

XML Query Requirements, W3C Working Draft 15 February 2001, http://www.w3.org/TR/2001/WD-xmlquery-req-20010215. Latest version: http://www.w3.org/TR/xmlquery-req.

XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116. Latest version: http://www.w3.org/TR/xslt.

XSL Transformations (XSLT) Version 1.1, W3C Working Draft 24 August 2001, http://www.w3.org/TR/2001/WD-xslt11-20010824/. This document is no longer updated and will be replaced by the XSLT 2.0 Requirements, which is not yet available