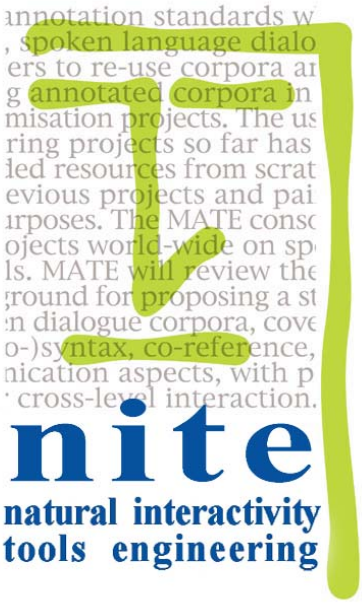
NITE project (IST-2000-26095)



Natural Interactivity Tools Engineering

NITE Deliverable D3.3a

NITE Workbench for Windows (NWB) Documentation

10 September 2003

Authors

Niels Ole Bernsen, Mykola Kolodnytsky, Dmytro Kupkin

NISLab, Odense, Denmark

Project ref. no.	IST-2000-26095
Project acronym	NITE
Deliverable status	Final
Contractual date of delivery	31 July 2003
Actual date of delivery	11 September 2003
Deliverable number	D3.3a
Deliverable title	NITE WorkBench for Windows (NWB) Documentation
Nature	Report
Status & version	Version 1.5 Final
Number of pages	34
WP contributing to the deliverable	WP3
WP / Task responsible	WP: NISLab-SDU, U. Edinburgh Task (this deliverable D3.3a): NISLab
Editor	NISLab
Author(s)	Niels Ole Bernsen, Mykola Kolodnytsky, Dmytro Kupkin
EC Project Officer	Philippe Gelin
Keywords	Coding tools, natural interactivity, multimodality, NITE WorkBench for Windows, NWB, documentation
Abstract (for dissemination)	This report from the NITE (Natural Interactivity Tools Engineering) project 2001-2003 provides documentation of the NITE WorkBench for Windows, a general-purpose natural interactivity coding tool developed in the NITE project.

Table of Contents

1 Introduction 5
2 Functionality 6
3 Architecture 9
3.1 General 9
3.2 Class diagrams 9
3.3 Object and sequence diagrams 12
3.4 Database structure 15
4 Program structures 22
4.1 Visual C++ projects 22
4.2 Project file types 22
4.3 C++ header and source files 23
4.3 Classes 28
5 Future work 33

1 Introduction

This report provides software documentation on Version 3 of the NITE Workbench for Windows (NWB3) natural interactivity data annotation and query tool which was publicly released at the NWB website nite.nis.sdu.dk on 31 August 2003. The present report may be consulted in conjunction with the NWB3 User Manual and the NWB3 Release Notes which are available at the NWB website. The two latter documents complement the present report in being far more oriented towards presenting functionality and how-to-operate information.

In what follows, Section 2 presents a concise overview of the functionality of NWB3. Section 3 presents architecture diagrams of the system at two adjacent levels of detail and documents the NWB3 database structure. Section 4 provides tables describing the NWB3 projects, project file types, header and source files, and classes. Section 5 describes future work.

2 Functionality

Table 2.1 shows the functionality which is available in NWB3. The overall structure of Table 2.1 is that of the common NITE specification presented in NITE Deliverables D1.1 and D1.1 Addendum. The detailed contents of Table 2.1 goes far beyond the common NITE specification, reflecting added user functionality requirements from the second NITE Tools Evaluation Workshop in Pisa in June 2003 as well as, and primarily, new requirements discovered during in-house analysis, continuing in-house user testing, and, increasingly, feedback from external users of the tool.

Comparison of Table 2.1 and the common NITE requirements specification shows that it is possible, using NWB3, to carry out general-purpose annotation, query analysis, and XML export of natural interactivity and multimodal data resources recorded in a range of different audio and video formats. This is demonstrated by a series of example coding projects which have been included in the NWB3 release.

Table 2.1 enables a quick overview of the two next general steps in NWB development, which may be briefly described as follows.

The first general step covers the four requirements marked '?' in the centre column of Table 2.1. These requirements address two new functionalities which, we have found, are top priority requirements from NWB3 users. One is to enable millisecond control of raw data file audio through a graphical (waveform) presentation of audio files. This will enable high-precision transcription at word and sub-word-level. The second is to enable structure coding, not only, at in NWB3, through time-stamping the structures coded, such as speech act tags referring to transcribed utterances or utterance parts, but by enabling the structure tags to inherit the timestamps of the tags they refer to. NWB4 will include these functionalities which we are working on at the time of writing.

The second general step covers the requirements listed in the right-most column of Table 2.1. Many of these are matters of detail. One main future objective, however, is to enable users to freely switch between the current symbolic-tabular view of the codings made and an information-equivalent analogue view. In the analogue view of codings, it is possible to directly perceive relative durations, onset times, and end times of the phenomena tagged according to one or several different coding schemes by relating the tagged phenomena to one another and/or to a common visible timeline.

List of Requirements	**Ready**	**Functionality candidates** **Comments**
Project file		
Create a new project	**x**	
Open / save a project	**x**	
Print coding files from Access	**x**	
		Print query results from Access
Copy and paste	**x**	
		Work with multiple projects
		Work with multiple raw data files having different timelines
Export/import data among projects	**x**	
Export coding files into XML files, preserving data structure	**x**	

Coding schemes specification		
Create new coding schemes	**x**	
Create coding scheme without link to raw data	**x**	
Modify existing coding schemes	**x**	
Provide orthographic transcription	**x**	
		Provide phonetic transcription
Insert real coding schemes and leave them in the workbench	**x**	
Remove last dummy coding scheme	**x**	
Coding scheme entry	**x**	
Choice between time-stamped coding schemes and structure coding schemes	**?**	
		Split tag into sub-tags Subsume two sub-tags under tag
Raw data control		
List of data types supported (see Table 2.2)	**x**	
		Enable playing of more sound file types?
		Enable playing of more video file types?
Visualise and play video files	**x**	
Play the audio track	**x**	
Navigate back and forth in the raw data	**x**	
Go frame-by-frame in video data	**x**	
Synchronize (one-way) all playing/displaying of data via control board	**x**	
Raw data control via graphical sound view	**?**	
Go by millisecond in sound data	**x**	
		Re-size video window
Annotation using coding schemes		
Underlying data structure	**x**	
		Automatically number turns, words, and syllables consecutively according to onset time
		Transcription: link speaker number to turns, words, syllables, transcription tags
		Link speaker to all tags for that speaker
Cross-level/structure coding using timestamps	**x**	
Real cross-level (structure) coding	**?**	
Coding scheme tag palette	**x**	
Two tag pallettes for timestamped and structure coding, respectively	**?**	
Annotated corpus editing (insert/delete tags)	**x**	
		Text editor for free-form annotation and comments
Order coding	**x**	
Information visualisation and customization		
Visualise audio data	**x**	

		Visualise the sound of more file types? If yes, which?
		Visualise audio data in several ways through waveform signal processing.
Visualise transcription	**x**	
Visualise annotated corpus (non-transcription)	**x**	
		Analogue coding visualisation.
		Zoom in and out in analogue timeline view.
Display object management	**x**	
		Hot keys for fast coding instead of tag palette.
Display object customisation: table and text manipulation	**x**	
Colour and font coding customisation	**x**	Single cell customisation.
Separate speakers in vertical columns/tiers for transcription	**x**	
Information extraction and analysis		
Run SQL query commands	**x**	
Easy-to-use, typical queries	**x**	
		GUI to describe and build SQL query.
		Statistical analysis: any? which?
Additional Requirements		
Installation kit V.3	**x**	
Module documentation V.2	**x**	
General architecture description (Fig. 2.1)	**x**	
User guide V.3	**x**	
Sort out license problems	**x**	
User support website	**x**	

Table 2.1. NWB3 functionality August 2003. Ongoing implementation tasks are marked with **?** The right-most column shows new tasks which have not started yet.

3 Architecture

This section presents a series of NWB architecture and database structure diagrams which jointly provide an understanding of the NWB as a Windows database application.

3.1 General

Figure 3.1 shows a conceptual view of the NWB architecture.

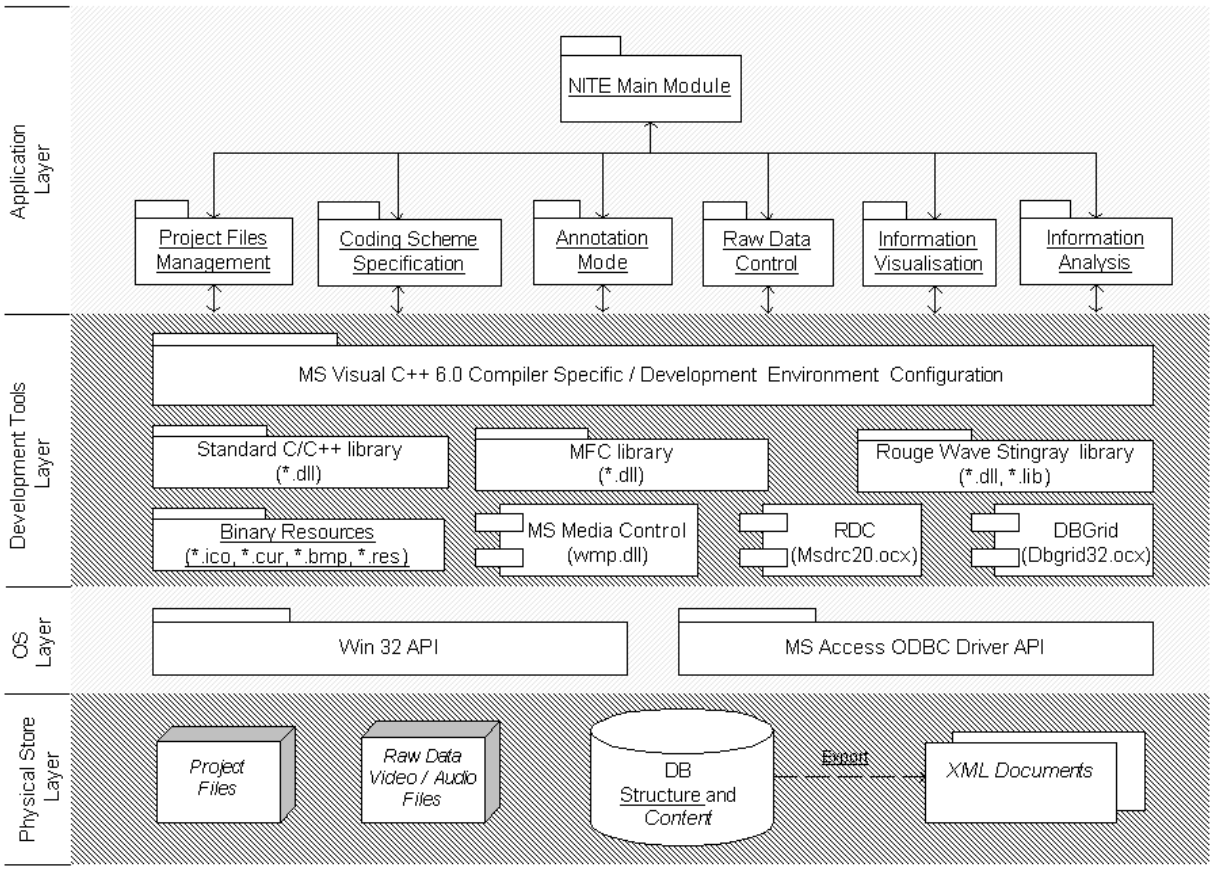


Figure 3.1. Conceptual view of the NWB architecture. Underlining shows NWB software developed in the NITE project, italics show user-provided resources, non-underlined, non-italicised font shows development resources used.

3.2 Class diagrams

The class diagrams presented in Figures 3.2 and 3.3 relate to the application layer shown in Figure 3.1. The class diagrams show the basic classes involved in the processes of project file management, coding scheme specification, annotation, raw data control, information visualisation and analysis.

Two different frameworks are being used to develop the NWB, i.e. the MS Visual C++ 6.0 MFC framework and the Rogue Wave Stingray Studio 2002 framework. These frameworks have been chosen in order to build an annotation tool with a familiar and easy-to-use interface.

The key advantage of using the MS Visual C++ MFC document/view architecture is that this architecture supports multiple views of the same document class. The single object of the document class provides access to the data stored in the NWB database. Multiple objects of

the view classes allow the user to process these data in accordance with the functional requirements of the NWB. In the NWB, the view classes provide a symbolic-tabular presentation of coding files as well as a graphical presentation of media file data (waveform presentation).

Rogue Wave Stingray Studio 2002 provides pre-built graphical and database access components for the Windows platform integrated into the MFC architecture. Mainly, these components support the grid presentation of the data fundamental to the annotation process.

Figure 3.2 shows the basic class structure of the NWB as an MFC Multiple Document Interface (MDI) application with some Rogue Wave Stingray library classes included.

Figure 3.3 shows the most important Rogue Wave Stingray classes involved in the Document/View architecture of the NWB, also illustrating the inheritance of the NWB classes and the polymorphism of their methods.

The entire hierarchy of derived and base classes is presented in Figures 3.2 and 3.3. Important NWB class functions and members are also shown. Figures 3.2 and 3.3 do not show the classes of auxiliary user interface elements included in the NWB. Only the fundamental classes creating the architecture of the application are presented.

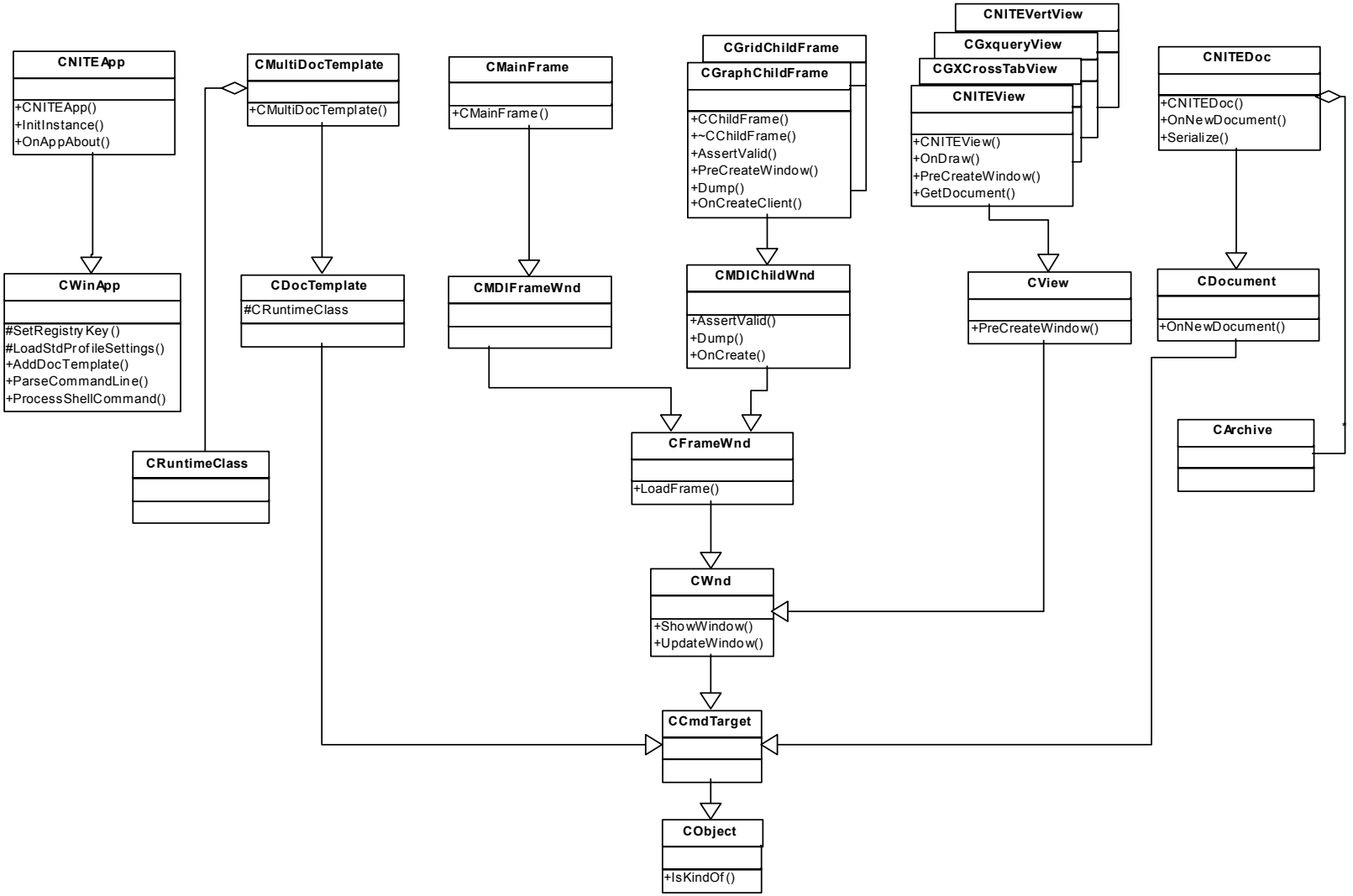


Figure 3.2. NWB class hierarchy

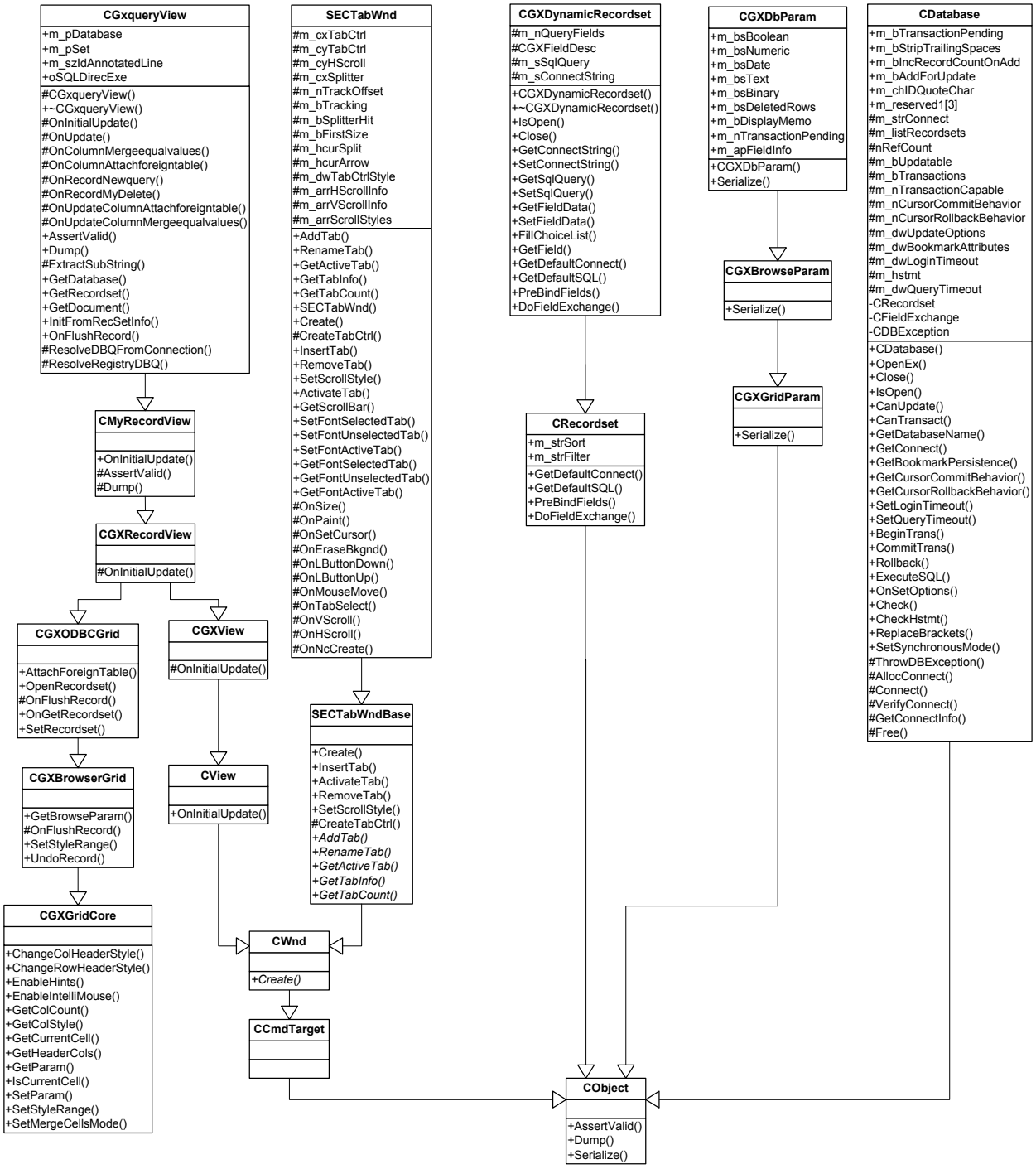


Figure 3.3. Rogue Wave Stingray classes included in NWB.

3.3 Object and sequence diagrams

The NWB object diagram (Figure 3.4) and sequence diagram (Figure 3.5) are linked to the class diagrams presented in Section 3.2 as follows.

Figure 3.4 illustrates the relations between class instances which are being created at run-time. The diagram also shows some important methods involved in the process of object initialisation. In particular, the Visual C++ framework uses the special MFC class RuntimeClass to create the set of objects. The MFC document template CMultiDocTemplate

is the key MFC class defining a set of view and document classes. Figure 3.4 shows four different objects of the CMultiDocTemplate class. The specific set of child frame and view objects corresponds to a CMultiDocTemplate object. Different CMultiDocTemplate objects (m_pTemplateHorView, m_pTemplateVerView, m_pTemplateCrossView, m_pTemplateFormView) use the same document class object named pDoc.

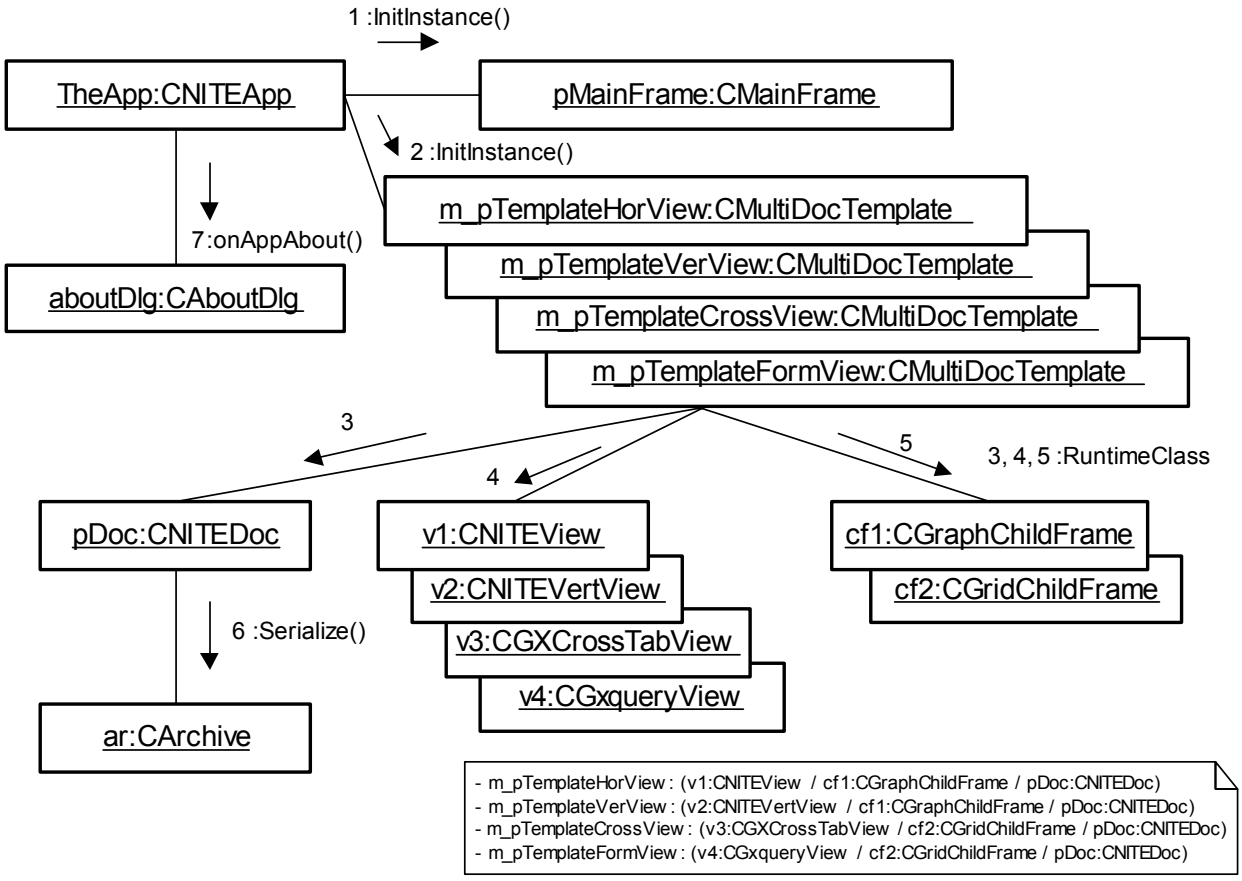


Figure 3.4. NWB object diagram.

The sequence diagram (Figure 3.5) emphasizes the relationships between objects as opposed to the sequence of messages. Figure 3.5 shows a scenario in which user starts the NWB. We see the interactions among classes in terms of an exchange of messages over time. The presented objects constitute the core of the NWB application.

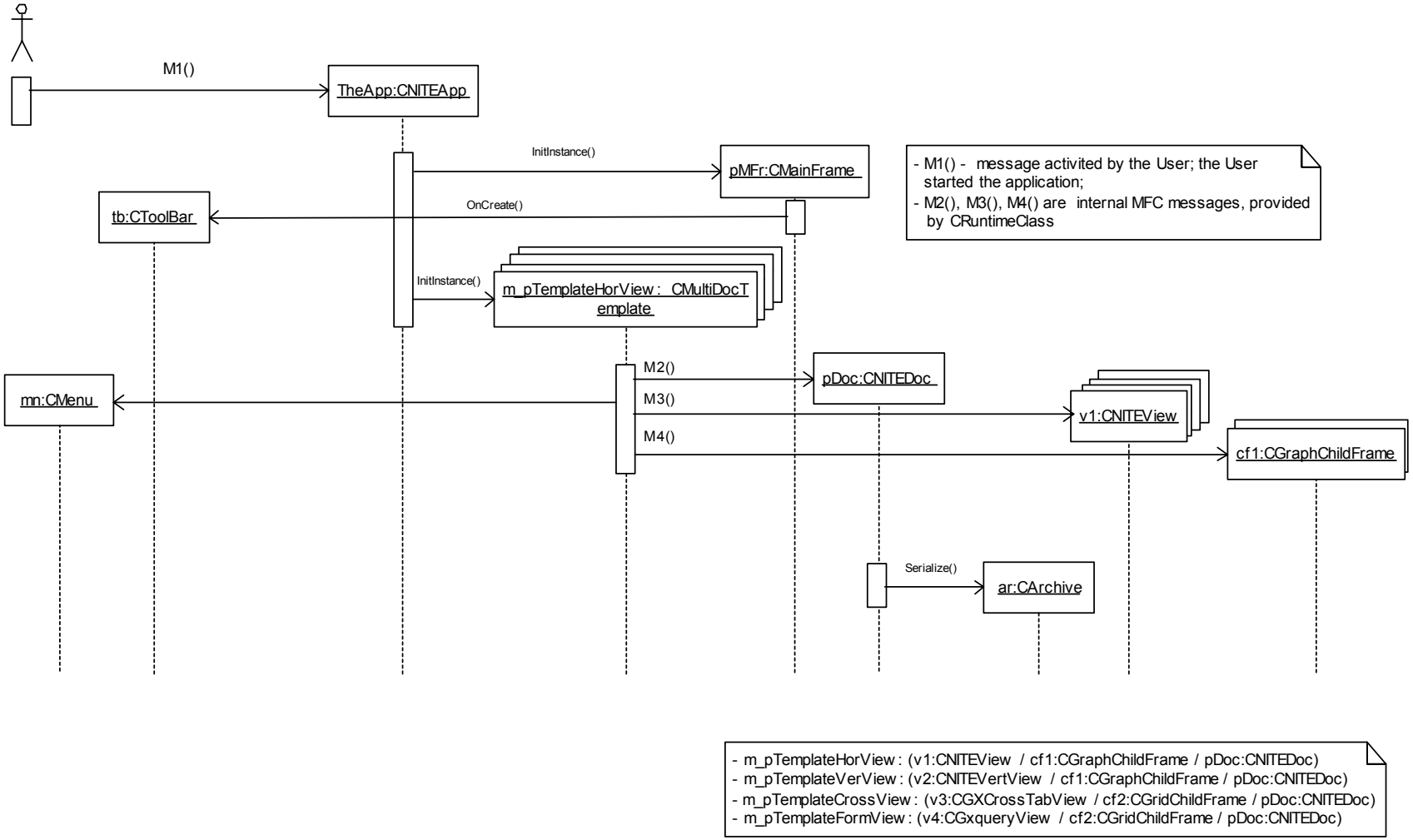


Figure 3.5. NWB sequence diagram.

3.4 Database structure

This section presents and illustrates the NWB relational database structure which is fundamental to the application.

The relationships between the tables in the NWB database are presented in Figure 3.6. The system works through the use of three basic queries, i.e. the default query QDefault (Figures 3.7, 3.8 and 3.9), the query for presenting speaker and transcription in the same row QCross (Figures 3.10, 3.11 and 3.12), and the cross-table query QCrossTimeLine, presenting the annotated timeline in a horizontal view (Figures 3.13, 3.14 and 3.15).

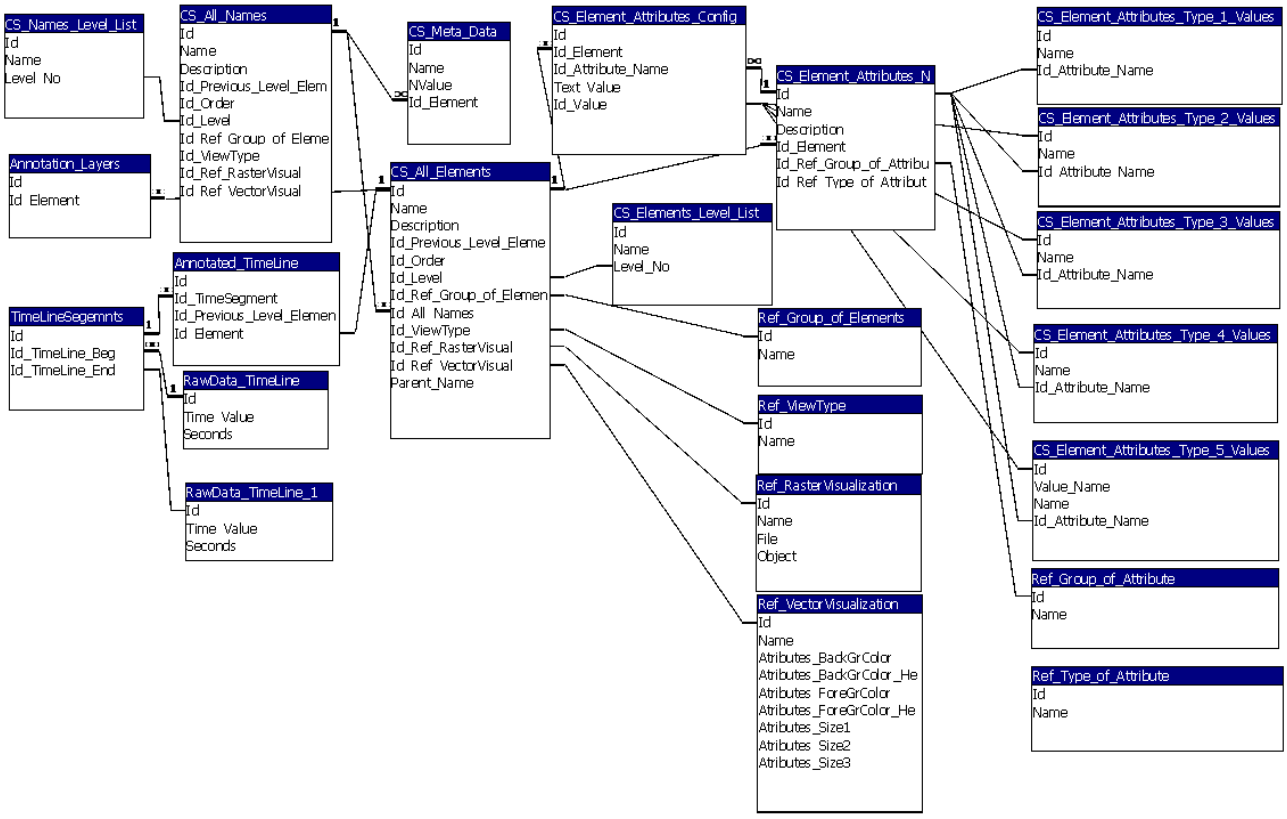


Figure 3.6. NWB database relationships.

SQL View of QDefault

```
SELECT TimeLineSegemnts.Id, Annotated_TimeLine.Id, RawData_TimeLine.Time_Value
AS Start, RawData_TimeLine_1.Time_Value AS [End], CS_All_Elements.Name AS
Elements
FROM (RawData_TimeLine AS RawData_TimeLine_1 INNER JOIN (RawData_TimeLine
INNER JOIN TimeLineSegemnts ON RawData_TimeLine.Id =
TimeLineSegemnts.Id_TimeLine_Beg) ON RawData_TimeLine_1.Id =
TimeLineSegemnts.Id_TimeLine_End) INNER JOIN (CS_All_Elements INNER JOIN
Annotated_TimeLine ON CS_All_Elements.Id = Annotated_TimeLine.Id_Element) ON
TimeLineSegemnts.Id = Annotated_TimeLine.Id_TimeSegment
ORDER BY RawData_TimeLine.Time_Value, RawData_TimeLine_1.Time_Value;
```

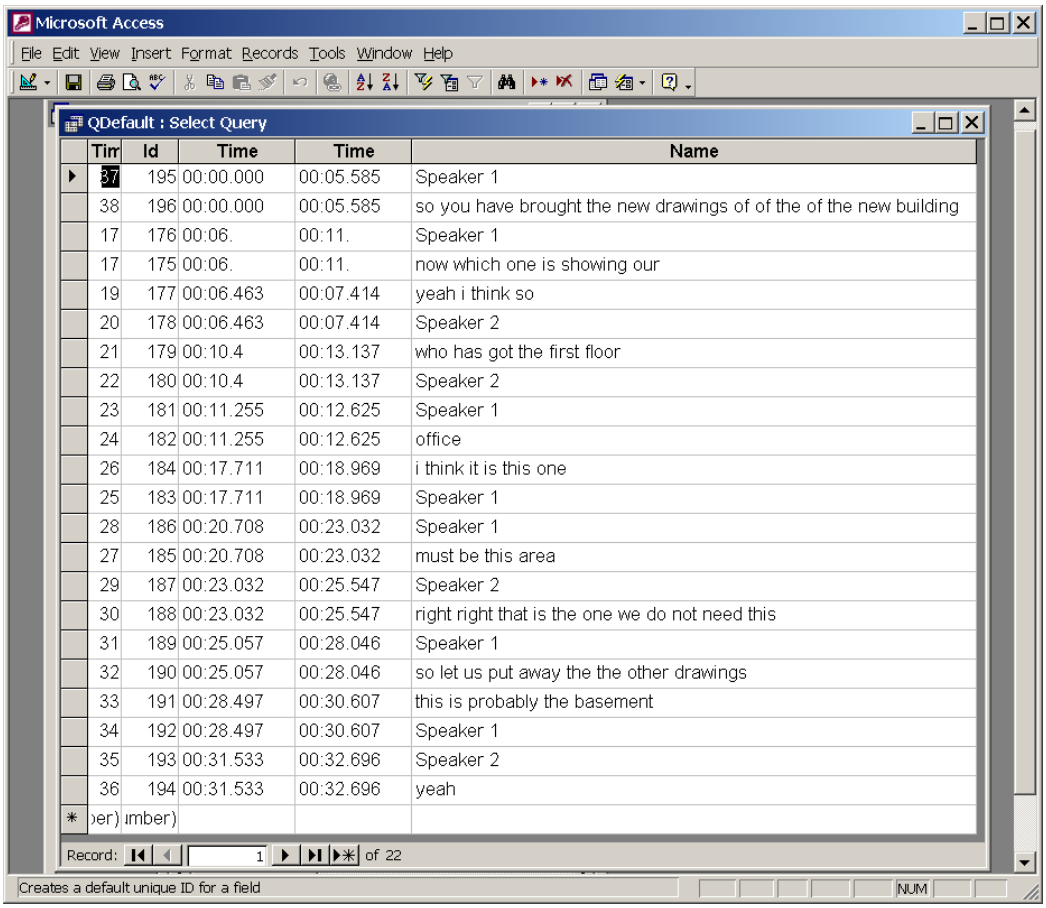


Figure 3.7. Result of executing the QDefault query in MS Access.

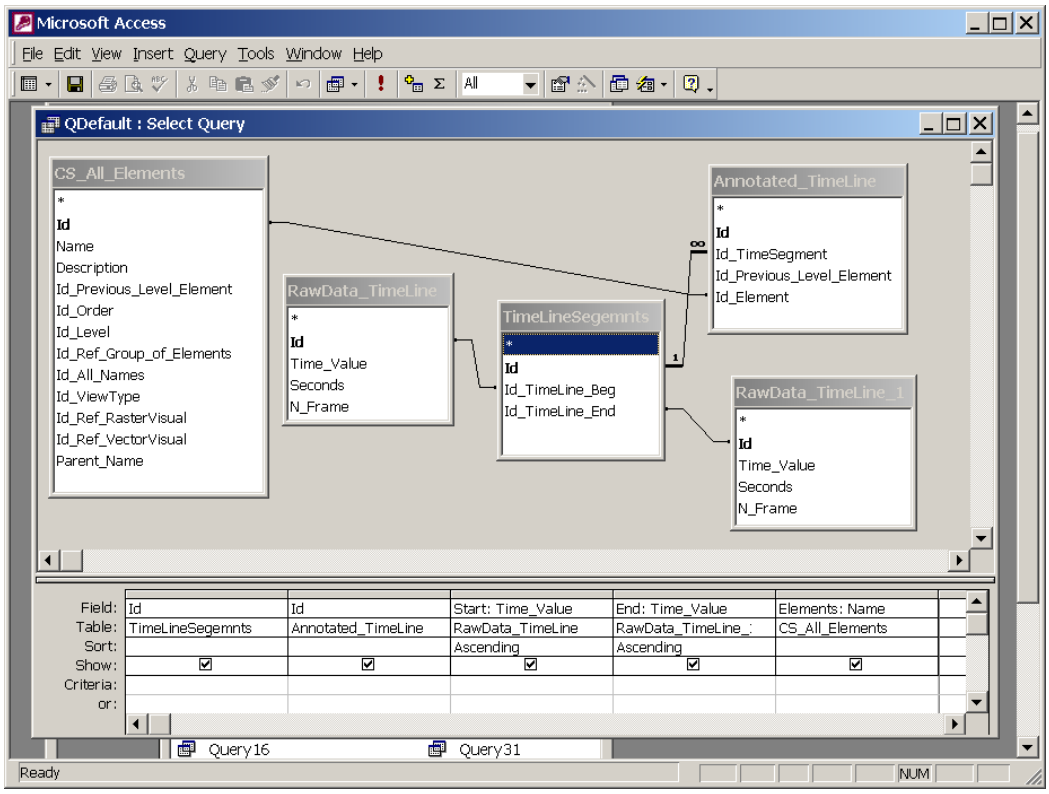


Figure 3.8. QDefault in MS Access query designer.

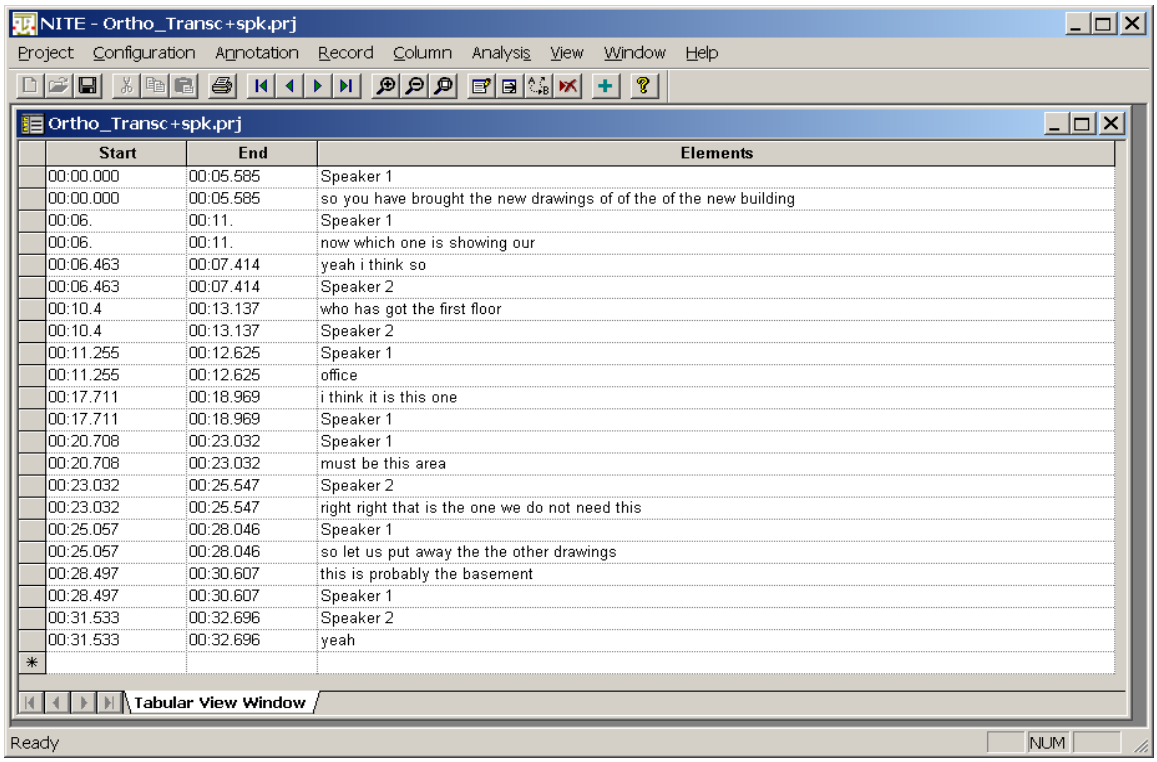
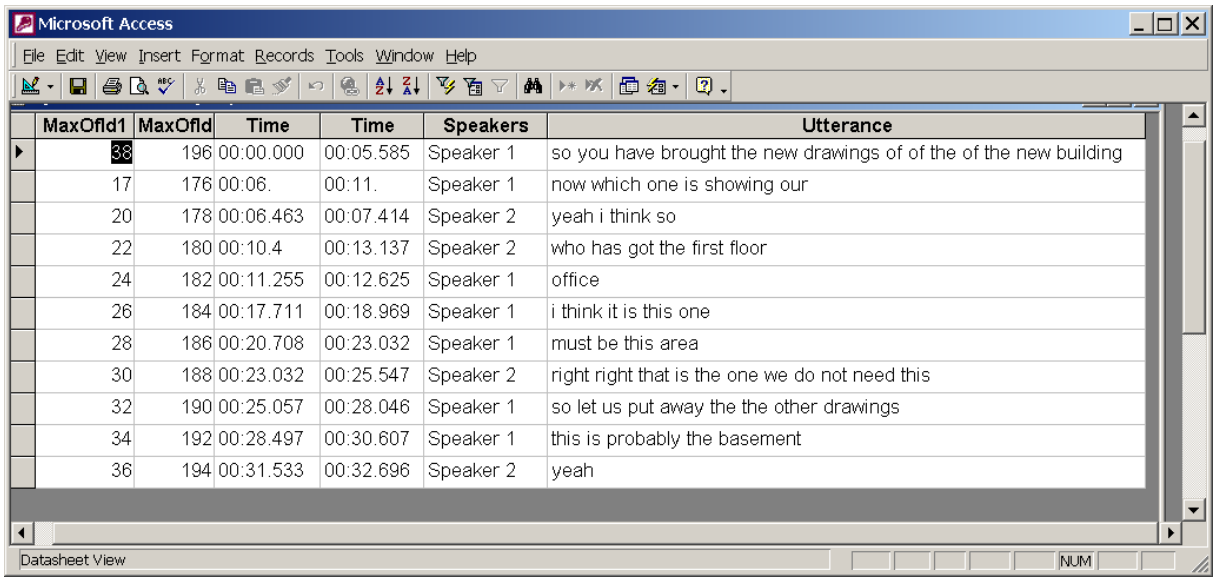


Figure 3.9. QDefault query in NWB3.

SQL View of QCross

TRANSFORM Max([CS_All_Elements].[Name]) AS MaxOfName
SELECT Max([TimeLineSegemnts].[Id]) AS MaxOfId1, Max([Annotated_TimeLine].[Id]) AS MaxOfId, [RawData_TimeLine].[Time_Value] AS Start, RawData_TimeLine_1.Time_Value AS [End]
FROM (RawData_TimeLine AS RawData_TimeLine_1 INNER JOIN (RawData_TimeLine INNER JOIN TimeLineSegemnts ON [RawData_TimeLine].[Id]=[TimeLineSegemnts].[Id_TimeLine_Beg]) ON RawData_TimeLine_1.Id=[TimeLineSegemnts].[Id_TimeLine_End]) INNER JOIN (CS_All_Elements INNER JOIN Annotated_TimeLine ON [CS_All_Elements].[Id]=[Annotated_TimeLine].[Id_Element]) ON [TimeLineSegemnts].[Id]=[Annotated_TimeLine].[Id_TimeSegment]
GROUP BY [RawData_TimeLine].[Time_Value], RawData_TimeLine_1.Time_Value
PIVOT [CS_All_Elements].[Parent_Name];



MaxOfId1	MaxOfId	Time	Time	Speakers	Utterance
38	196	00:00.000	00:05.585	Speaker 1	so you have brought the new drawings of of the of the new building
17	176	00:06.	00:11.	Speaker 1	now which one is showing our
20	178	00:06.463	00:07.414	Speaker 2	yeah i think so
22	180	00:10.4	00:13.137	Speaker 2	who has got the first floor
24	182	00:11.255	00:12.625	Speaker 1	office
26	184	00:17.711	00:18.969	Speaker 1	i think it is this one
28	186	00:20.708	00:23.032	Speaker 1	must be this area
30	188	00:23.032	00:25.547	Speaker 2	right right that is the one we do not need this
32	190	00:25.057	00:28.046	Speaker 1	so let us put away the the other drawings
34	192	00:28.497	00:30.607	Speaker 1	this is probably the basement
36	194	00:31.533	00:32.696	Speaker 2	yeah

Figure 3.10. Result of executing the QCross query in MS Access.

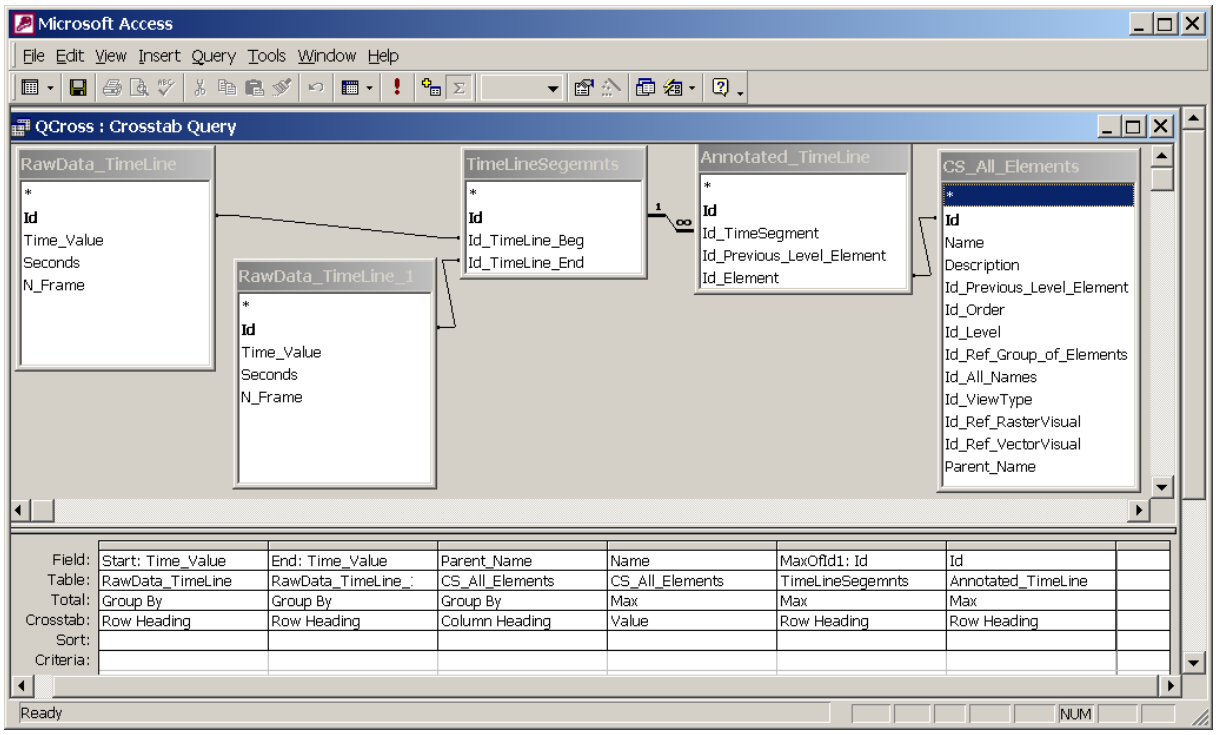


Figure 3.11. QCross in MS Access query designer.

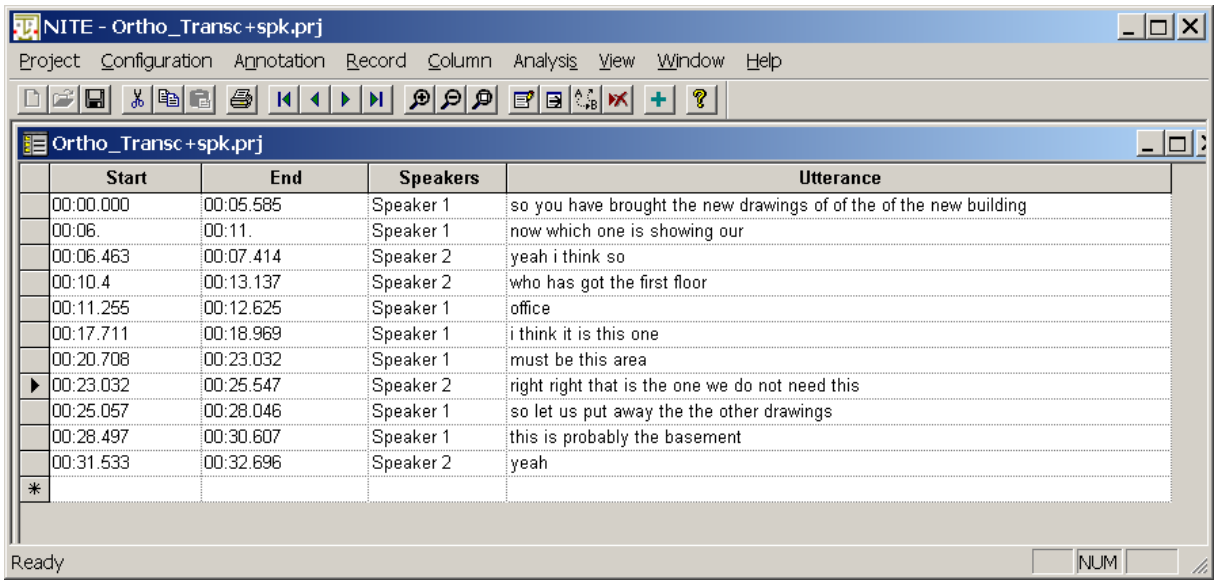


Figure 3.12. QCross query in NWB3.

SQL View of QCrossTimeLine

```
TRANSFORM Max([CS_All_Elements].[Name]) AS MaxOfName
SELECT Max([TimeLineSegemnts].[Id]) AS MaxOfId1, Max([Annotated_TimeLine].[Id]) AS MaxOfId, [CS_All_Elements].[Parent_Name]
FROM (RawData_TimeLine INNER JOIN TimeLineSegemnts ON [RawData_TimeLine].[Id]=[TimeLineSegemnts].[Id_TimeLine_Beg]) INNER JOIN (CS_All_Elements INNER JOIN Annotated_TimeLine ON [CS_All_Elements].[Id]=[Annotated_TimeLine].[Id_Element]) ON [TimeLineSegemnts].[Id]=[Annotated_TimeLine].[Id_TimeSegment]
GROUP BY [CS_All_Elements].[Parent_Name]
PIVOT [RawData_TimeLine].[Time_Value];
```

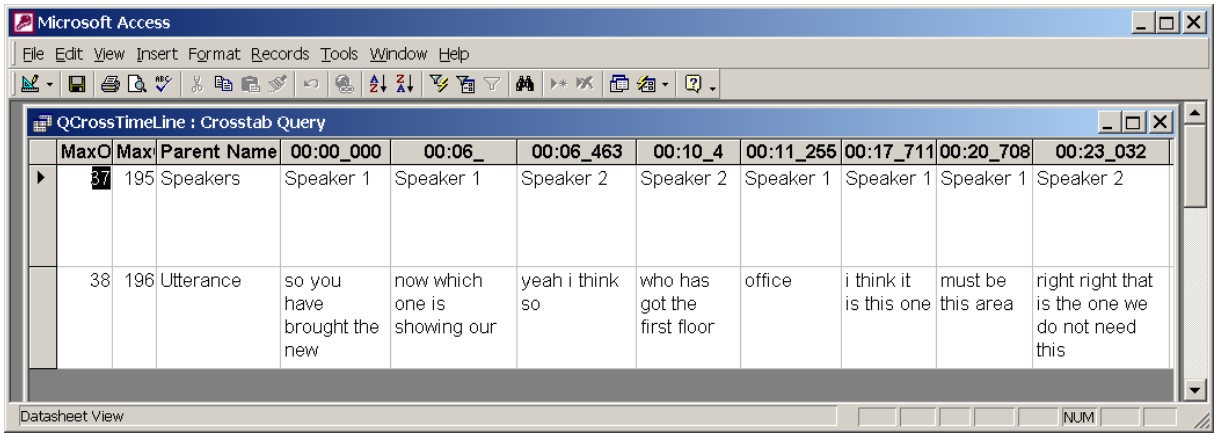


Figure 3.13. Result of executing the QCrossTimeLine query in MS Access.

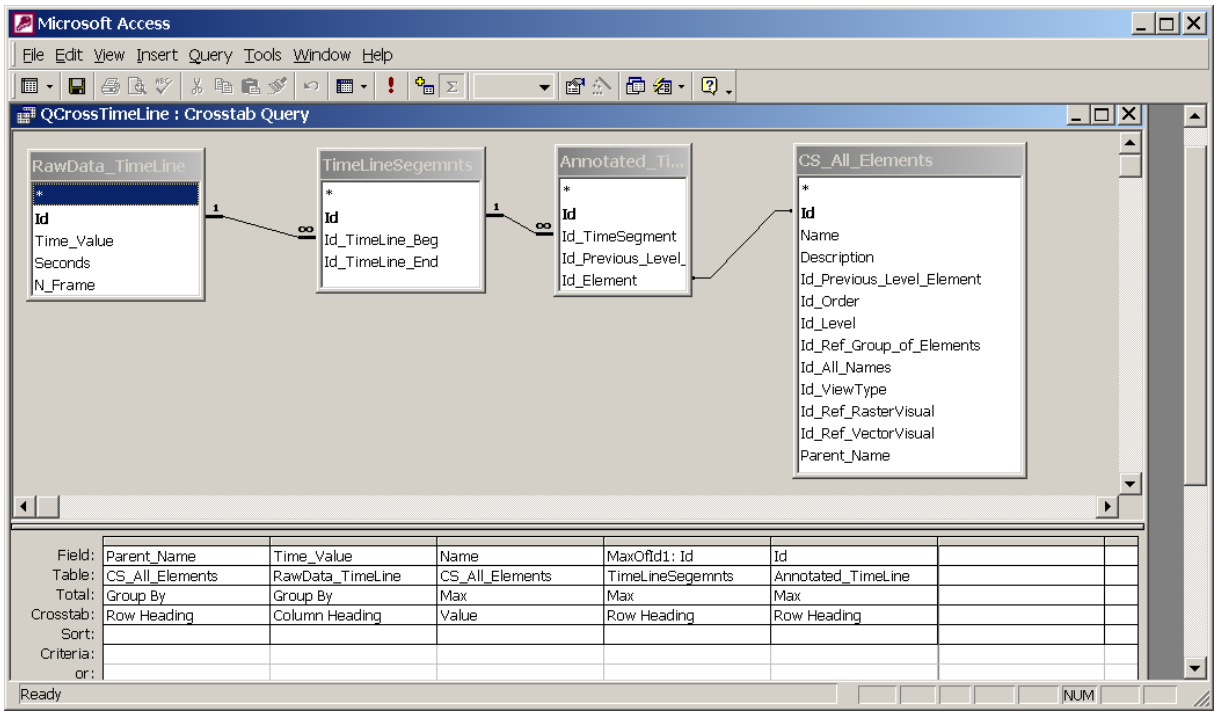


Figure 3.14. QCrossTimeLine in MS Access query designer.

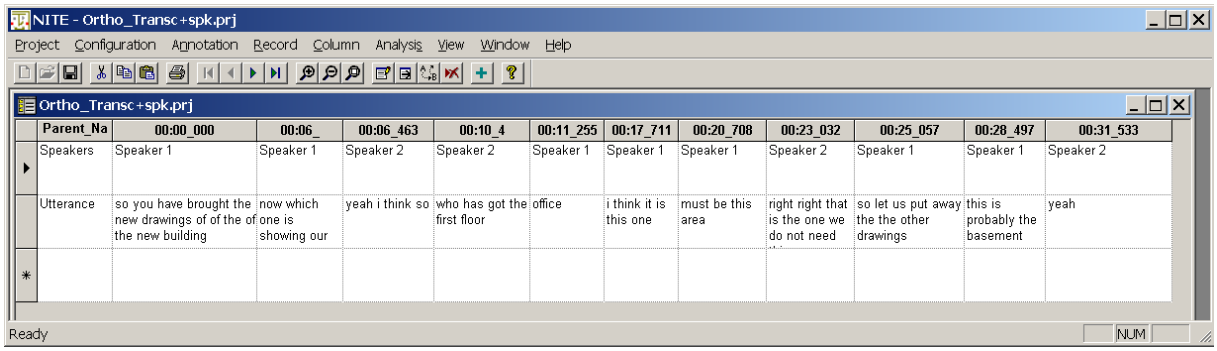


Figure 3.15. QCrossTimeLine query in NWB3.

Table 3.1 shows the NWB database tables.

	Table Name	**Description**
1.	CS_Names_Level_List	Keeps level numbers and level names
2.	CS_All_Names	Keeps element names
3.	CS_Meta_Data	Keeps meta data information
4.	CS_Element_Attributes_Config	Keeps attribute's configuration information
5.	CS_Element_Attributes_Name	Keeps names of element's attributes
6.	CS_Element_Attributes_Type_1_Values	Keeps description of attribute groups
7.	CS_Element_Attributes_Type_2_Values	
8.	CS_Element_Attributes_Type_3_Values	

9.	CS_Element_Attributes_Type_4_Values	
10.	CS_Element_Attributes_Type_5_Values	
11.	Ref_Group_of_Attribute	Reference table keeps values of attribute's groups
12.	Ref_Type_of_Attribute	Reference table keeps values of attribute's types
13.	CS_Elements_Level_List	Keeps element level's information
14.	CS_All_Elements	Keeps element names and their IDs
15.	Ref_Group_of_Elements	Reference table which describes groups of elements
16.	Ref_ViewType	Reference table which describes the type of view for possible tags of coding schemes
17.	Ref_RasterVisualization	Reference table which contains list of images (with associated files) for tag visualisation
18.	Ref_VectorVisualization	Reference table which contains list of possible vector objects with their attributes for tag visualisation
19.	RawData_TimeLine	Keeps start-time and end-time values for translation turns
20.	Annotation_Layers	Keeps annotation layers' information
21.	Annotated_TimeLine	Keeps translation turn's time IDs
22.	TimeLineSegemnts	Keeps time segment's information

Table 3.1. List of NWB database tables.

4 Program structures

This section shows the projects composing the NWB (Section 4.1), the file types composing the projects (Section 4.2), the header and source files created (Section 4.3), and the project classes (Section 4.4).

4.1 Visual C++ projects

- NITE project generates NITE.exe file.
- About project generates About.dll file.
- ChildDocView project generates ChildDocView.dll file.
- CS_Config project generates CS_Config.dll file.
- CS_Palette project generates CS_Palette.dll file.
- CtrlBoard project generates CtrlBoard.dll file
- DBBoundCtrls project generates DBBoundCtrls.dll file.
- TransDialog project generates TransDialog.dll file.
- Video_Dlg project generates Video_Dlg.dll file.
- NITEExportPrinting project generates NITEExportPrinting.exe file.

4.2 Project file types

Each Visual C++ project mentioned in Section 4.1 includes all or most of the following file types:

- Files with CLW extension. These files contain information used by the VC++ ClassWizard to edit existing classes or add new classes. The ClassWizard also uses these files to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.
- Files with DSP extension. The project files contain information at the project level and are used to build a single project or subproject DSW VC++ Workspace file.
- Files with NCB extension. These are VC++ Class View support files.
- Files with OPT extension. These are current VC++ workspace configuration files.
- Files with REG extension. REG files show the kind of registration settings VC++ framework sets.
- Files with PLG extension. These are VC++ Build Log files.
- Files with RC extension. This are listings of all of the Microsoft Windows resources that the program uses. RC file includes the icons, bitmaps, and cursors that are stored in the RES subdirectory.
- Files with APS extension. These are VC++ Resource View support files.
- Files with DEF extension. DEF files contain information about the DLL that must be provided to run with Microsoft Windows. They define parameters such as the name and description of the DLL. They also export functions from the DLL.
- Files with ICO extension. Icon files.
- StdAfx.cpp, StdAfx.h. These files are used to build precompiled header (PCH) files and precompiled types files named StdAfx.obj.
- resource.h files. This are the standard header files, which define new resource Ids.

4.3 C++ header and source files

File Name		Lines of Code	Description	Project Name
1	NITE.cpp	240	This is the main application source file that contains the application class - CNITEApp	Nite
2	NITE.h	67	This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CNITEApp application class	
3	AboutDialog.cpp	50	This is the main DLL source file that contains the definition of DllMain()	About
4	AboutDialog.h	47	Header file for AboutDialog.cpp file	
5	ChildDocView.cpp	52	This is the main DLL source file that contains the definition of DllMain()	ChildDocView
6	ChildFormFrame.cpp	37	Contains definitions of CchildFormFrame class members and methods	
7	ChildFormFrame.h	57	Header file for ChildFormFrame.cpp file	
8	ChildFrm.cpp	997	Contains definitions of CGraphChildFrame class members and methods	
9	ChildFrm.h	177	Header file for ChildFrm.cpp file	
10	GXQUEDOC.CPP	194	Contains definitions of CGxqueryDoc class members and methods	
11	GXQUEDOC.H	61	Header file for GXQUEDOC.CPP file	
12	GXQUEVW.CPP	2212	Contains definitions of CGXCrossTabView, CgxqueryView, CMyRecordView classes members and methods	
13	GXQUEVW.H	204	Header file for GXQUEVW.CPP file	
14	MainFrm.cpp	479	Contains definitions of CMainFrame class members and methods	
15	MainFrm.h	95	Header file for MainFrm.cpp file	
16	myrecvw.cpp	305	Contains definitions of CmyStylesDialog, CMyStyleSheet classes members and methods	
17	MYRECVW.H	114	Header file for myrecvw.cpp file	
18	NITEDoc.cpp	1520	Contains definitions of CNITEDoc class members and methods	
19	NITEDoc.h	189	Header file for NITEDoc.cpp file	
20	NITEView.cpp	917	Contains definitions of CgrSlider, CNITEFormView, CNITEVertView, CNITEView classes members and methods	
21	NITEView.h	254	Header file for NITEView.cpp file	
22	Querydlg.cpp	77	Contains definitions of CMyQueryDialog class members and methods	

23	QUERYDLG.H	43	Header file for Querydlg.cpp file	
24	RECMDI.CPP	94	Contains definitions of CRecordStatusMDIChildWnd class members and methods	
25	RECMDI.H	64	Header file for RECMDI.CPP file	
26	RenameTabDialog.cpp	53	Contains definitions of RenameTabDialog class members and methods	
27	RenameTabDialog.h	56	Header file for RenameTabDialog.cpp file	
28	CS_Config.cpp	52	This is the main DLL source file that contains the definition of DllMain()	CS_Config
29	CS_Elements_Modif_Dlg.cpp	269	Contains definitions of CCS_Elements_Modif_Dlg class members and methods	
30	CS_Elements_Modif_Dlg.h	98	Header file for CS_Elements_Modif_Dlg.cpp file	
31	CS_Element_Attr_Config_Dlg.cpp	267	Contains definitions of CCS_Element_Attr_Config_Dlg class members and methods	
32	CS_Element_Attr_Config_Dlg.h	82	Header file for CS_Element_Attr_Config_Dlg.cpp file	
33	CS_Element_Attr_List_Dlg.cpp	285	Contains definitions of CCS_Element_Attr_List_Dlg class members and methods	
34	CS_Element_Attr_List_Dlg.h	92	Header file for CS_Element_Attr_List_Dlg.cpp file	
35	CS_Element_Attr_Select_Name_Dlg.cpp	132	Contains definitions of CCS_Element_Attr_Select_Name_Dlg class members and methods	
36	CS_Element_Attr_Select_Name_Dlg.h	83	Header file for CS_Element_Attr_Select_Name_Dlg.cpp file	
37	CS_Element_Attr_Select_Value_Dlg.cpp	102	Contains definitions of CCS_Element_Attr_Select_Value_Dlg class members and methods	
38	CS_Element_Attr_Select_Value_Dlg.h	79	Header file for CS_Element_Attr_Select_Value_Dlg.cpp file	
39	CS_Element_Attr_Values_Dlg.cpp	141	Contains definitions of CCS_Element_Attr_Values_Dlg class members and methods	
40	CS_Element_Attr_Values_Dlg.h	85	Header file for CS_Element_Attr_Values_Dlg.cpp file	
41	CS_Meta_Data_Dlg.cpp	119	Contains definitions of CCS_Meta_Data_Dlg class members and methods	
42	CS_Meta_Data_Dlg.h	65	Header file for CS_Meta_Data_Dlg.cpp file	
43	CS_Names_Config_Dlg.cpp	250	Contains definitions of CCS_Names_Config_Dlg class members and methods	
44	CS_Names_Config_Dlg.h	92	Header file for CS_Names_Config_Dlg.cpp file	
45	CS_Selection_Dlg.cpp	126	Contains definitions of CCS_Selection_Dlg class members and methods	
46	CS_Selection_Dlg.h	77	Header file for CS_Selection_Dlg.cpp file	

47	DB_TreeView.cpp	897	Contains definitions of CDB_TreeView class members and methods	
48	DB_TreeView.h	127	Header file for DB_TreeView.cpp file	
49	EditString_Dialog.cpp	54	Contains definitions of CEditString_Dialog class members and methods	
50	EditString_Dialog.h	57	Header file for EditString_Dialog.cpp file	
51	SQLConstStr.cpp	35	Contains definitions of CSQLConstStr class members and methods	
52	SQLConstStr.h	62	Header file for SQLConstStr.cpp file	
53	SQLDirectExec.cpp	217	Contains definitions of CSQLDirectExec class members and methods	
54	SQLDirectExec.h	70	Header file for SQLDirectExec.cpp file	
55	Coding_Segment_Dlg.cpp	206	Contains definitions of CCoding_Segment_Dlg, CODBCSampleGridWnd classes members and methods	CS_Palette
56	coding_segment_dlg.h	118	Header file for Coding_Segment_Dlg.cpp file	
57	CSPalette_Page1.cpp	167	Contains definitions of CCSPalette_Page1 class members and methods	
58	CSPalette_Page1.h	76	Header file for CSPalette_Page1.cpp file	
59	CSPalette_Page2.cpp	347	Contains definitions of CCSPalette_Page2 class members and methods	
60	CSPalette_Page2.h	87	Header file for CSPalette_Page2.cpp file	
61	CS_Palette.cpp	52	This is the main DLL source file that contains the definition of DllMain()	
62	PaletteCSPropSheet.cpp	28	Contains definitions of CPaletteCSPropSheet class members and methods	
63	PaletteCSPropSheet.h	63	Header file for PaletteCSPropSheet.cpp file	
64	PaletteMiniFrame.cpp	138	Contains definitions of CPaletteMiniFrame class members and methods	
65	PaletteMiniFrame.h	56	Header file for PaletteMiniFrame.cpp file	
66	CtrlBoard.cpp	52	This is the main DLL source file that contains the definition of DllMain()	CtrlBoard
67	ctrlboardminifrm.cpp	90	Contains definitions of CCtrlBoardMiniFrm class members and methods	
68	ctrlboardminifrm.h	64	Header file for ctrlboardminifrm.cpp file	
69	innerdialog.cpp	928	Contains definitions of CinnerDialog, CinnerDialog2 classes members and methods	
70	innerdialog.h	167	Header file for innerdialog.cpp file	
71	DBBoundCtrls.cpp	52	This is the main DLL source file that contains the definition of DllMain()	DBBoundCtrls
72	font.cpp	111	Contains definitions of COleFont class members and methods	
73	font.h	37	Header file for font.cpp file	

74	msdgridctrl.cpp	728	Contains definitions of CMsDgridCtrl class members and methods	
75	msdgridctrl.h	169	Header file for msdgridctrl.cpp file	
76	picture.cpp	55	Contains definitions of CPicture class members and methods	
77	picture.h	39	Header file for picture.cpp file	
78	rdc.cpp	577	Contains definitions of CRdc class members and methods	
79	rdc.h	136	Header file for rdc.cpp file	
80	rdocolumns.cpp	40	Contains definitions of CrdoColumns class members and methods	
81	rdocolumns.h	39	Header file for rdocolumns.cpp file	
82	rdoconnections.cpp	51	Contains definitions of CrdoConnections class members and methods	
83	rdoconnections.h	40	Header file for rdoconnections.cpp file	
84	rdoenvironments.cpp	52	Contains definitions of CrdoEnvironments class members and methods	
85	rdoenvironments.h	70	Header file for rdoenvironments.cpp file	
86	rdoerror.cpp	64	Contains definitions of CrdoError class members and methods	
87	rdoerror.h	39	Header file for rdoerror.cpp file	
88	rdoerrors.cpp	41	Contains definitions of CrdoErrors class members and methods	
89	rdoerrors.h	39	Header file for rdoerrors.cpp file	
90	rdoparameter.cpp	99	Contains definitions of CrdoParameter class members and methods	
91	rdoparameter.h	44	Header file for rdoparameter.cpp file	
92	rdoparameters.cpp	35	Contains definitions of CrdoParameters class members and methods	
93	rdoparameters.h	38	Header file for rdoparameters.cpp file	
94	rdopreparedstatement.cpp	257	Contains definitions of CrdoPreparedStatement class members and methods	
95	rdopreparedstatement.h	71	Header file for rdopreparedstatement.cpp file	
96	rdopreparedstatements.cpp	19	Contains definitions of CrdoPreparedStatements class members and methods	
97	rdopreparedstatements.h	36	Header file for rdopreparedstatements.cpp file	
98	rdoqueries.cpp	35	Contains definitions of CrdoQueries class members and methods	
99	rdoqueries.h	38	Header file for rdoqueries.cpp file	
100	rdoresultsets.cpp	35	Contains definitions of CrdoResultsets class members and methods	
101	rdoresultsets.h	38	Header file for rdoresultsets.cpp file	

102	rdotable.cpp	65	Contains definitions of CrdoTable class members and methods	
103	rdotable.h	43	Header file for rdotable.cpp file	
104	rdotables.cpp	41	Contains definitions of CrdoTables class members and methods	
105	rdotables.h	39	Header file for rdotables.cpp file	
106	_rdocolumn.cpp	225	Contains definitions of C_rdoColumn class members and methods	
107	_rdocolumn.h	60	Header file for _rdocolumn.cpp file	
108	_rdoconnection.cpp	268	Contains definitions of C_rdoConnection class members and methods	
109	_rdoconnection.h	76	Header file for_rdoconnection.cpp file	
110	rdoconnections.cpp	51	Contains definitions of C_rdoConnections class members and methods	
111	rdoconnections.h	40	Header file for rdoconnections.cpp file	
112	_rdoengine.cpp	150	Contains definitions of C_rdoEngine class members and methods	
113	_rdoengine.h	55	Header file for _rdoengine.cpp file	
114	_rdoenvironment.cpp	124	Contains definitions of C_rdoEnvironment class members and methods	
115	_rdoenvironment.h	53	Header file for _rdoenvironment.h file	
116	_rdoresultset.cpp	405	Contains definitions of C_rdoResultset class members and methods	
117	_rdoresultset.h	93	Header file for _rdoresultset.cpp file	
118	transdialog.cpp	421	This is the main DLL source file that contains the definition of DllMain(). Contains also definitions of CTransDialog class members and methods	TransDialog
119	transdialog.h	92	Header file for file	
120	mediaplayer2.cpp	1398	Contains definitions of CMediaPlayer2 class members and methods	Video_Dlg
121	mediaplayer2.h	239	Header file for mediaplayer2.cpp file	
122	mediaplayerdvd.cpp	436	Contains definitions of CMediaPlayerDvd class members and methods	
123	mediaplayerdvd.h	91	Header file for mediaplayerdvd.cpp file	
124	Video_Dialog.cpp	124	Contains definitions of CVideo_Dialog class members and methods	
125	Video_Dialog.h	74	Header file for Video_Dialog.cpp file	
126	Video_Dlg.cpp	52	This is the main DLL source file that contains the definition of DllMain()	
127	NITEExportPrinting.cpp	79	This is the main application source file that contains the application class CNITEExportPrintingApp	NITEExportPrinting

128	NITEExportPrinting.h	60	This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares application class	
129	NITEExportPrintingDlg.cpp	356	Contains definitions of CNITEExportPrintingDlg and CAboutDlg classes members and methods	
130	NITEExportPrintingDlg.h	75	Header file for NITEExportPrintingDlg.cpp file	
131	XMLExpAllElements.cpp	83	Contains definitions of CXMLExpAllElements class members and methods	
132	XMLExpAllElements.h	67	Header file for XMLExpAllElements.cpp file	
133	XMLExportAnnTLNITE3.cpp	71	Contains definitions of CXMLExportAnnTLNITE3 class members and methods	
134	XMLExportAnnTLNITE3.h	59	Header file for XMLExportAnnTLNITE3.cpp file	

4.3 Classes

	Class	**File**	**Purpose**	**Project**
1.	CNITEApp	NITE.cpp, .NITE.h	Creating global application object	Nite
2.	CAboutDlg	AboutDialog.h, AboutDialog.cpp	Creating about application dialogue window	About
3.	CChildFormFrame	ChildFormFrame.h, ChildFormFrame.cpp	Creating child window with analogue data presentation	ChildDocView
4.	CGraphChildFrame	ChildFrm.h, ChildFrm.cpp	Creating child window with graphic data presentation	
5.	CGrSlider	NITEView.h, NITEView.cpp	Creating slider object	
6.	CGXCrossTabView	GXQUEVW.H GXQUEVW.CPP	Creating view object	
7.	CGxqueryDoc	GXQUEDOC.H GXQUEDOC.CPP	Creating document object	
8.	CGxqueryView	GXQUEVW.H GXQUEVW.CPP	Creating view object	
9.	CMainFrame	MainFrm.h MainFrm.cpp	Creating main frame object	
10.	CMyQueryDialog	QUERYDLG.H Querydlg.cpp	Creating new query window	

11.	CMyRecordView	GXQUEVW.H GXQUEVW.CPP	Basic class CgxqueryView class	
12.	CMyStylesDialog	MYRECVW.H, myrecvw.cpp	Creating styles dialogue object	
13.	CMyStyleSheet	MYRECVW.H, myrecvw.cpp	Creating styles sheet object	
14.	CNITEDoc	NITEDoc.h, NITEDoc.cpp	Creating document object	
15.	CNITEFormView	NITEView.h, NITEView.cpp	Creating view object	
16.	CNITEVertView	NITEView.h, NITEView.cpp	Creating view object	
17.	CNITEView	NITEView.h, NITEView.cpp	Creating view object	
18.	CRecordStatusMDIChild Wnd	RECMDI.H, RECMDI.CPP	Creating child window object	
19.	RenameTabDialog	RenameTabDialog.h, RenameTabDialog.cpp	Creating rename tab dialogue object	
20.	CCS_Element_Attr_Config_Dlg	CS_Element_Attr_Config_Dlg.h, CS_Element_Attr_Config_Dlg.cpp	Creating attributes configuration dialogue object	CS_Config
21.	CCS_Element_Attr_List_Dlg	CCS_Element_Attr_List_Dlg.h, CCS_Element_Attr_List_Dlg.cpp	Creating attributes list dialogue object	
22.	CCS_Element_Attr_Select_Name_Dlg	CS_Element_Attr_Select_Name_Dlg.h CS_Element_Attr_Select_Name_Dlg.cpp	Creating attributes selection name dialogue object	
23.	CCS_Element_Attr_Select_Value_Dlg	CS_Element_Attr_Select_Value_Dlg.h CS_Element_Attr_Select_Value_Dlg.cpp	Creating attributes selection value dialogue object	
24.	CCS_Element_Attr_Values_Dlg	CS_Element_Attr_Values_Dlg.h CS_Element_Attr_Values_Dlg.cpp	Creating elements selection value dialogue object	
25.	CCS_Elements_Modif_Dlg	CS_Elements_Modif_Dlg.h CS_Elements_Modif_Dlg.cpp	Creating elements modification dialogue object	
26.	CCS_Meta_Data_Dlg	CS_Meta_Data_Dlg.h CS_Meta_Data_Dlg.cpp	Creating meta data dialogue object	

27.	CCS_Names_Config_Dlg	CS_Names_Config_Dlg.h CS_Names_Config_Dlg.cpp	Creating names configuration dialogue object	
28.	CCS_Selection_Dlg	CS_Selection_Dlg.h CS_Selection_Dlg.cpp	Creating selection dialogue object	
29.	CDB_TreeView	DB_TreeView.h DB_TreeView.cpp	Creating tree view object	
30.	CEditString_Dialog	EditString_Dialog.h EditString_Dialog.cpp	Creating string edit dialogue object	
31.	CSQLConstStr	SQLConstStr.h SQLConstStr.cpp	Creating SQL connection string object	
32.	CSQLDirectExec	SQLDirectExec.h SQLDirectExec.cpp	Creating SQL execute object	
33.	CCoding_Segment_Dlg	coding_segment_dlg.h Coding_Segment_Dlg.cpp	Creating coding segment dialogue object	CS_Palette
34.	CCSPalette_Page1	CSPalette_Page1.h CSPalette_Page1.cpp	Creating coding palette page 1 object	
35.	CCSPalette_Page2	CSPalette_Page2.h CSPalette_Page2.cpp	Creating coding palette page 2 object	
36.	CODBCSampleGridWnd	coding_segment_dlg.h Coding_Segment_Dlg.cpp	Creating coding segment dialogue object	
37.	CPaletteCSPropSheet	PaletteCSPropSheet.h PaletteCSPropSheet.cpp	Creating property sheet object	
38.	CPaletteMiniFrame	PaletteMiniFrame.h PaletteMiniFrame.cpp	Creating palette miniframe object	
39.	C_rdoColumn	_rdocolumn.h _rdocolumn.cpp	Creating RDO column object	DBBoundCtrls
40.	C_rdoConnection	_rdoconnection.h _rdoconnection.cpp	Creating RDO connection object	
41.	C_rdoEngine	_rdoengine.h _rdoengine.cpp	Creating RDO engine object	

42.	C_rdoEnvironment	_rdoenvironment.h _rdoenvironment.cpp	Creating RDO environment object	
43.	C_rdoResultset	_rdoresultset.h _rdoresultset.cpp	Creating RDO resultset object	
44.	CMsDgridCtrl	msdgridctrl.h msdgridctrl.cpp	Creating MS datagrid object	
45.	COleFont	font.h font.cpp	Creating OLE font object	
46.	CPicture	picture.h picture.cpp	Creating OLE picture wrapper object	
47.	CRdc	rdc.h rdc.cpp	Creating RDC object	
48.	CrdoColumns	rdocolumns.h rdocolumns.cpp	Creating RDO columns object	
49.	CrdoConnections	rdoconnections.h rdoconnections.cpp	Creating RDO connections object	
50.	CrdoEnvironments	rdoEnvironments.h rdoEnvironments.cpp	Creating RDO environments object	
51.	CrdoError	rdoerror.h rdoerror.cpp	Creating RDO error object	
52.	CrdoErrors	rdoerrors.h rdoerrors.cpp	Creating RDO errors object	
53.	CrdoParameter	rdoparameter.h rdoparameter.cpp	Creating RDO parameter object	
54.	CrdoParameters	rdoparameters.h rdoparameters.cpp	Creating RDO parameters object	
55.	CrdoPreparedStatement	rdopreparedstatement.h rdopreparedstatement.cpp	Creating RDO prepared statement object	
56.	CrdoPreparedStatements	rdopreparedstatements.h rdopreparedstatements.cpp	Creating RDO prepared statements object	

57.	CrdoQueries	rdoqueries.h rdoqueries.cpp	Creating RDO queries object	
58.	CrdoResultsets	rdoresultsets.h rdoresultsets.cpp	Creating RDO resultset object	
59.	CrdoTable	rdotable.h rdotable.cpp	Creating RDO table object	
60.	CrdoTables	rdotables.h rdotables.cpp	Creating RDO tables object	
61.	CCtrlBoardMiniFrm	ctrlboardminifrm.h ctrlboardminifrm.cpp	Creating controlboard miniframe object	CtrlBoard
62.	CInnerDialog	innerdialog.h innerdialog.cpp	Creating inner dialogue object	
63.	CinnerDialog2	innerdialog.h innerdialog.cpp	Creating inner dialogue 2 object	
64.	CTransDialog	transdialog.h transdialog.cpp	Creating transcription dialogue object	TransDialog
65.	CMediaPlayer2	mediaplayer2.h mediaplayer2.cpp	Creating media player object	Video_Dlg
66.	CMediaPlayerDvd	mediaplayerdvd.h mediaplayerdvd.cpp	Creating media player for DVD object	
67.	CVideo_Dialog	Video_Dialog.h Video_Dialog.cpp	Creating video dialogue object	
68.	CNITEExportPrintingDlg	NITEExportPrintingDlg.h NITEExportPrintingDlg.cpp	Creating export-printing dialogue object	NITEExportPrinting
69.	CXMLExpAllElements	XMLExpAllElements.h XMLExpAllElements.cpp	Creating all elements table recordset object	
70.	CXMLExportAnnTLNITE3	XMLExportAnnTLNITE3.h XMLExportAnnTLNITE3.cpp	Creating coding file recordset object	
71.	CAboutDlg	NITEExportPrintingDlg.h NITEExportPrintingDlg.cpp	Creating dialogue object	

5 Future work

Table 2.1 above clearly shows that building an easy-to-use, general-purpose tool for the annotation and analysis of natural interactivity data is a complex task which needs to be staged in a number of development steps, each of which involving both major steps forward and a series of less perspicuous but nevertheless important steps.

By the end of the NITE project on 31 July, 2003, we have demonstrated a solid and, for a research prototype, stable tool core, the NWB3, which enables: new coding scheme insertion, minute raw video data source control, time-stamped coding using a coding palette created on the basis of the coding scheme inserted and used, symbolic-tabular views of the coding files made by users, waveform view of sound data in various data formats, limited customisability of coding files, unlimited querying of coding files by using SQL, limited querying of coding files by using easy-access, pre-formed queries, and XML export of coding files and query results. Structure coding, including the coding of clusters of coordinated natural interactive behaviours, can be done as time-stamped codings. Jointly, these capabilities qualifies the NWB3 as a limited-purpose natural interactivity coding tool with a strong potential to become the world's first easy-to-use general-purpose natural interactivity coding tool.

The obvious question is: what does it take to turn the NWB into the world's first easy-to-use general-purpose natural interactivity coding tool? The answer is: not very much! Before answering the question, however, we must explain our operational distinction between special-purpose, limited-purpose, and general-purpose natural interactivity coding tools. A *special-purpose coding tool* enables annotation (and possibly coding scheme insertion, querying, import-export, customisation, etc.) of a particular modality of natural interactivity data, such as orthographic transcription (e.g. the Transcriber tool), phonetic transcription (e.g. the PRAAT tool), gesture transcription, or otherwise. A *limited-purpose coding tool* has more general versatility for coding natural interactivity data than has a special-purpose tool. However, a limited-purpose coding tool still suffers from limitations which does not qualify the tool as a general-purpose one, such as data-structure limitations which prevent the tool for handling and/or displaying complex coding schemes and coding files, cross-level and/or cross-modality coding, structure coding done on the basis of time-stamped coding, and/or the need to do complex programming in order for the tool to do any particular complex natural interactivity coding task. A *general-purpose coding tool* does not have these limitations. It must be noted that this is an operational definition of a general-purpose coding tool. Ideally, a general-purpose tool would satisfy, as it stands, any coding purpose and any user preference as to the viewing and customisation of coding schemes, coding palettes, raw data, coding files, query results, etc. However, such a tool is not likely to exist, ever, which is why we have to resort to an operational, i.e. implementationally feasible, definition of a general-purpose tool.

Given the definitions above, the NWB lacks three properties only to become a general-purpose coding tool in the operational sense. These are:

- raw data control via the waveform raw audio data view. This will enable millisecond precision in sound file segmentation, including word-level and subword-level segmentation;
- single-cell customisation of coding file views. This will enable the use of colour, font, etc. for displaying coordinated clusters of natural interactive behaviours, such as long-range dependencies, speech and gesture deictics, and a lot more; and
- "real" structure coding in which structure tags are inserted onto time-stamped tags without the need for additional, sometimes error-prone, time-stamped coding.

These functionalities constitute our top priorities for future NWB development. The functionalities will be included in the NWB4 to be released later in 2003. With those functionalities, the NWB will become the world's first general-purpose natural interactivity coding tool. With this tool, users will be able to code raw data audio and video files using an unlimited number of coding schemes, exploring the full complexity of natural interactivity behaviours through customised control of the display of their findings. Moreover, the users will be able to do so by using the NWB coding tool "as-is", i.e. without having to do any special-purpose programming to achieve their aims.

Once the above has been achieved in NWB4, two major next steps remain in order optimise the NWB for all or most users. These are:

- add information-equivalent analogue coding views to the NWB's current symbolic-tabular views;
- add an easy-to-use interface to the NWB's current SQL query interface.

The first of these steps will enable users to switch from the current symbolic-tabular view of coding files to an analogue view in which each time-stamped or structure-coded phenomenon will be immediately visible on a common timeline in its temporal relations to other natural interactive phenomena, including phenomena coded using an unlimited number of other coding schemes. NWB4 customisation options will take are of the display of long-range dependencies as well as clusters and other cross-level and cross-modality relationships. The second step will enable non-SQL literate users to easily query their coding files. Customisation options will take care of the display of important aspects of the query results.

To summarise, the NWB3 is the core for a general-purpose coding tool which will emerge when we have taken the three steps described above. Once the NWB has become a general-purpose coding tool, we aim to take the next two steps described above, which will make the NWB an easy-to-use general-purpose natural interactivity coding tool likely to satisfy all or most users in this important field for the future of natural interactive and multimodal systems development.