universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

“Achieving Flexibility in Courseware Design:

Concept and Programming of a Microsoft® PowerPoint®

to eduWEAVER Plug-in Transformation Tool”

Verfasser

Ivaylo Velikovski

angestrebter akademischer Grad

Magister der Sozial- und Wirtschaftswissenschaften

(Mag. rer. soc. oec.)

Wien, im Februar 2008

Studienkennzahl lt. Studienblatt	A 157
Studienrichtung lt. Studienblatt	Internationale Betriebswirtschaft
Betreuer:	o.Univ.-Prof. Dr. Dimitris Karagiannis

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Diplomarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben.

Alle Stellen, die den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Wien, im Februar 2008

Ivaylo Velikovski

To my family

Table of Contents

I ABBREVIATIONS

ADL	Adonis definition language
API	Application Programming Interface
COM	Component Object Model
DOM	Document Object Model
DTD	Document type definition
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated development environment
OLE	Object Linking and Embedding
RAD	Rapid application development
SAX	Simple API for XML
VBA	Visual Basic for Application
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

II LIST OF FIGURES

III LIST OF TABLES

IV LIST OF CODE

1 Introduction

In the “Information Age” where the need for knowledge increases as the need for manual labour decreases, lifelong learning is the only way to the continued development of modern world[1]. One of the means that can help in this approach is e-Learning. A new term that expresses one of the most modern approaches in the learning process, being strongly dependent of modern technologies like the World Wide Web.

However, current practices in e-Learning show a desperate picture of a wide gap between the good wishes and the reality. This means that the implementation of e-Learning solutions is happening randomly, sporadically, and with varying degrees of success. Although everybody recognises that e-Learning has the potential to improve greatly the learning process and experience, there are still many others who see lots of problems in the current stage.

Plenty of discussions are held, but it still seems not to be defined how e-Learning should best be used.

The knowledge consumers nowadays have a total different approach in a way they search, find and adopt knowledge. Information has become easy to reach, easy to be updated and is multimedia richer than every before. Students want to acquaint knowledge fast and easy. They do not want to know how difficult and cost consuming it is to prepare a good piece of courseware. At the other side the teaching community seems not to be so much enthusiastic about developing the ultimate e-Learning material. In spite of that, an increasing number of institutions are attempting to initiate their own style of e-Learning content production.

The generic term e-Learning or e-Education does not properly describe what is happening in today’s teaching.”… The letter “e-“meaning using the internet and information technology for teaching is only the technical part to be solved…”.[2] Although the second term “Learning” seems to be well known in an appropriate E-Learning concept it has a totally new meaning and following new didactical concepts.

E-Learning can help the learning community to overcome some vital problems of the traditional class teaching process like:

- Lack of individuality (mass universities versus individual online learning courses)

[1] See Gizella (2004)
[2] See Karagiannis et al. (2004).

- Speech misunderstanding (sitting at the back of a class room and unable to hear the speaker’s voice properly)
- Speed of knowledge transfer (some students need more time to understand the transferred knowledge)
- Feel inconvenient to ask questions (some students do not like to ask questions when they do not understand something at a given moment)
- Teaching material being not up-to-date[3] (the teaching staff does not have the resources to maintain up-to-date teaching materials, printed documents are more expensive than digital ones)
- Lack of flexibility concerning lecture time arrangements (online learning is 24 hours available)

All these problems can be overcome by a well designed and developed e-Learning platform. Such a platform should enable the teaching staff and the learning community to collaborate for an optimal knowledge transfer.

Before delivering content to that platform, the teaching staff has to solve at least two big problems. A technical one, concerning teaching materials development by using their own available resources like books, lecture notes and slides, in order to become independent of expensive and inflexible multimedia managing solutions in the educational sector. The second one is a didactical one, concerning the creation of new didactical models for e-Learning[4], in order to use the new aspects of information and communication technologies in the world of instructional design.

An optional aspect can also be the simulation and optimization of educational processes regarding times and costs, in order to measure the advantage of e-Learning regarding those two vital components (less costs and time savings).

Teachers need a tool that integrates these supporting functionalities and helps them bridge the gap between classroom teaching and the new e-Learning approach.

A large problem that the teaching staff is facing is that they often need to deliver educational content in different forms (digital or print). But at the same time they want to use an already existing one. So they are forced to rewrite or copy-paste the content from one document into another, almost inventing the wheel every time.

[3] See Meyer [Last visited 12.11.2007]
[4] See Bajnai et al. (2004).

In the time of high-end IT technology is this “waste of time = waste of money” process not acceptable. Once created, the content of teaching materials should be easily managed and transformed into another presentation form as quickly and easily as possible. Not the transformation between files is what the teaching staff should be burdened with, but the didactic problems should stay in the front. For example such a situation can be to re-concept the teaching methods, used when transferring knowledge in an e-Learning platform environment. Having only materials, which are created for in class teaching can bring the teaching staff great problems. So they should have the possibility to manage the content of those materials to suit the new tasks. Some of the possibilities, that the teaching staff will prefer to use are to include the PowerPoint® speaker notes on a web site, warehousing the content in a repository for further manipulation, or transport the content in CSM (Content Management System) or LMS (Learning Management System) using a different web site layout and many more.

This diploma thesis will take a close look at a possible scenario for the teaching staff to use their old existing teaching materials and leverage the educational process with a new e-Learning component. For this case an existing modelling tool named eduWEAVER modelling library that has been created within the eduBITE project[5] will be use. The eduWEAVER modelling library is based on the meta-modelling tool ADVISOR®[6]. ADVISOR® is a product of BOC Information Technologies Consulting AG[7], developed within an ESPRIT[8] project of the European Union from 1998 until 2000. ADVISOR®’s main feature is its method independence. This means that, starting from the ADVISOR® meta tool level additional new instructional modelling tools can be derived. Such tools are represented by the so-called method libraries (for example- eduWEAVER), which allow particularly the definition of arbitrary modelling languages without any programming effort. A close look at eduWEAVER will be taken in chapter *2.5 eduWEAVER and ADL*.

The aim of eduWEAVER will be to play a mediator role and a technology enabler between the traditional “Class Teaching World” and the modern “E-Learning” one.

A possible scenario for solving the problem of delivering quality e-Learning instructional materials is outlined in figure 1.1.

[5] See eduBITE – Projektinformationen. [Last visited 04.05.2007]
[6] See ADVISOR: Advanced instruction technology for services organisation.[Last visited 16.08.2007]
[7] See BOC Information Technologies Consulting AG. [Last visited 16.08.2007]
[8] See Espirit Project.[Last visited 16.08.2007]

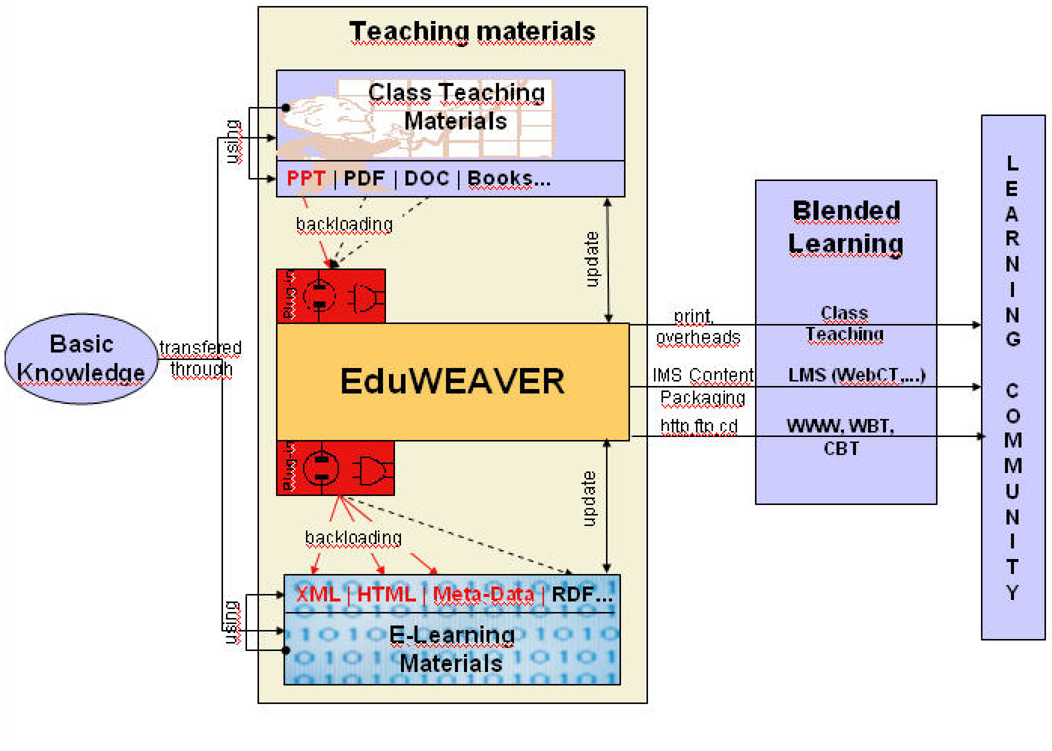


Figure 1.1 Solution scenario of the eduWEAVER plug-in application

The process of transferring the "Basic Knowledge" to the "Learning Community" can have different paths. Two of them are shown in figure 1.1. The first possible knowledge transferring path can be one of a traditional class teaching. This process involves the usage of class teaching materials (PowerPoint® presentation on a beamer, printed .pdf or .doc files, text books and other printed materials). The presence of an instructor and members of the learning community is also a vital part of this educational pattern. The second possible scenario can be the one of e-Learning. This process normally consists of the intensive usage of personal computer for distance learning with a direct or indirect communication between the instructor and the learning community. The e-Learning materials are mostly in a digital form (.html, .pdf, .doc, .ppt). For the optimal usage in a modern e-Learning platform those digital materials should be all converted into .html or .xml (plus formatting through css, xsl), to enable search possibilities and easy update of the courseware. The two learning processes can be sometimes combined and in that way forming the so called “Blended learning”[9] educational process.

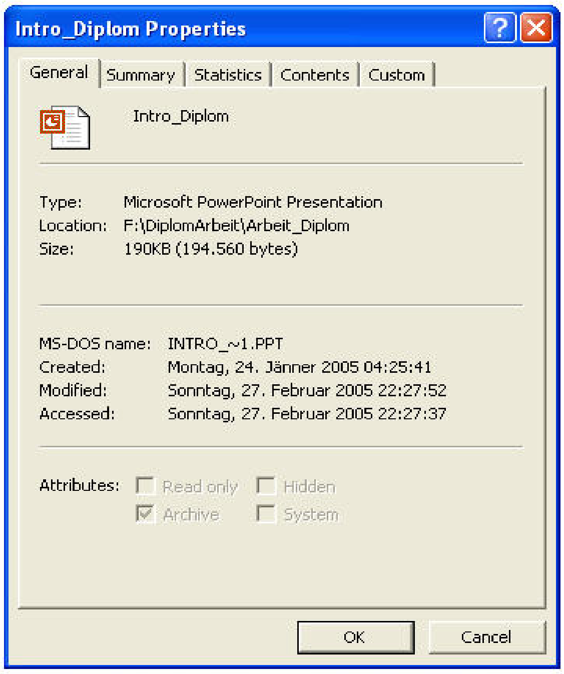
The aim of the diploma thesis will be to show a possible scenario in form of a plug-in application for the eduWEAVER modelling library. It should enable the easy transaction of existing class teaching materials (mostly Microsoft® Office files and particularly PowerPoint® presentations) into quality e-Learning materials with a minimum of manual work. For that reason the diploma thesis will take a close look at the possibility of transforming (through a “backloading” process) .ppt files to .html, .xml and particularly .adl (Adonis Definition Language) files for import in eduWEAVER.

PowerPoint® presentations consist of slides, which may have pictures, tables, figures, videos and text. For the purpose of building the plug-in an atomization process regarding a PowerPoint® presentation object should be carried out. A possible atomization of a PowerPoint® presentation is the dividing the presentation in slides as the little possible piece of information regarding any kind of presentation. A further atomization down to a shape object (each sentence, each arrow, each image or each auto shape element) will not represent any kind of valuable information because it will be an out of the context chunk of information. The meaning of PowerPoint® is to represent ideas through a visual union of graphical forms and text. Out of that context are those elements only valuable for the process of searching in an index system, where they will play a role as a reference to a slide of a presentation.

As PowerPoint® belongs to the family of Microsoft® Office its files have meta data which is kept invisible for other programs other than Microsoft® Office. This data is not searchable by any other programs or web search machines until the data is not presented in the content body of the file.

This meta data is usually represented by File >> Properties in any Microsoft® Office application (see figure 1.2 -1.6):

[9] See Grillitsch [Last visited 16.08.2007]



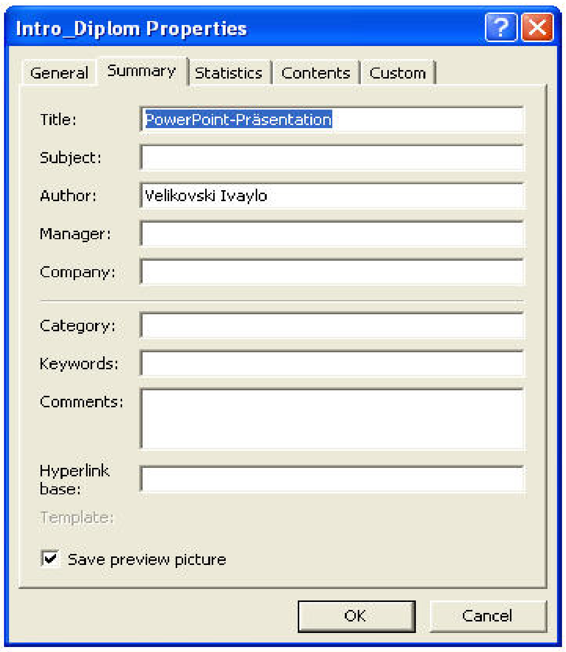
Intro_Diplom

Type: Microsoft PowerPoint Presentation
Location: F:\DiplomArbeit\Arbeit_Diplom
Size: 190KB (194.560 bytes)

MS-DOS name: INTRO_~1.PPT
Created: Montag, 24. Jänner 2005 04:25:41
Modified: Sonntag, 27. Februar 2005 22:27:52
Accessed: Sonntag, 27. Februar 2005 22:27:37

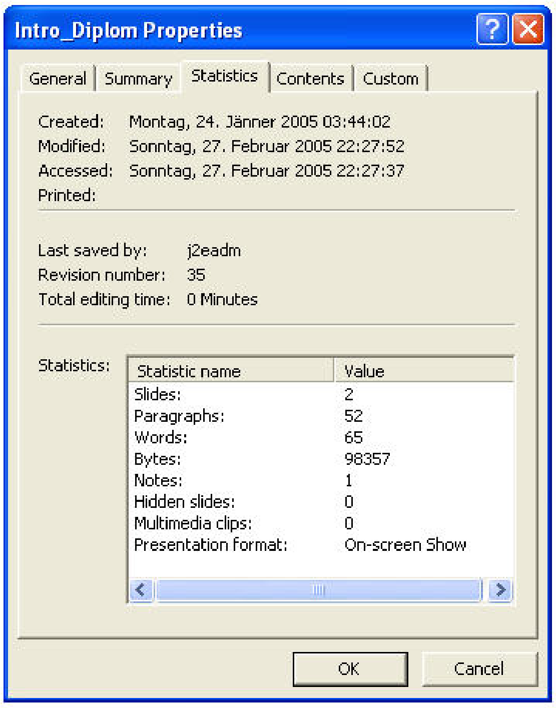
Attributes: ☐ Read only ☐ Hidden
☑ Archive ☐ System

Figure 1.2 (Type, Location, Size, MS-DOS name, Created, Modified, Accessed)



Title: PowerPoint-Präsentation
Subject:
Author: Velikovski Ivaylo
Manager:
Company:
Category:
Keywords:
Comments:
Hyperlink base:
Template:
☑ Save preview picture

Figure 1.3 (Title, Subject, Auhtor, Manager, Company, Category, Keywords, Comments, Hyperlinkbase)



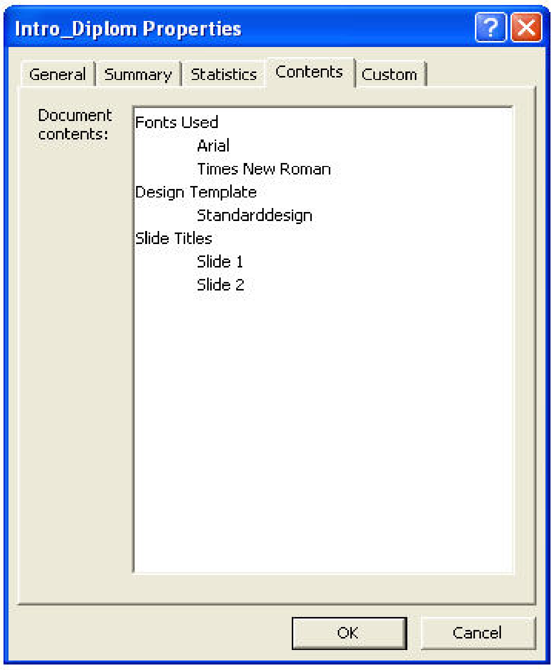
Created: Montag, 24. Jänner 2005 03:44:02
Modified: Sonntag, 27. Februar 2005 22:27:52
Accessed: Sonntag, 27. Februar 2005 22:27:37
Printed:

Last saved by: j2eadm
Revision number: 35
Total editing time: 0 Minutes

Statistics:

Statistic name	Value
Slides:	2
Paragraphs:	52
Words:	65
Bytes:	98357
Notes:	1
Hidden slides:	0
Multimedia clips:	0
Presentation format:	On-screen Show

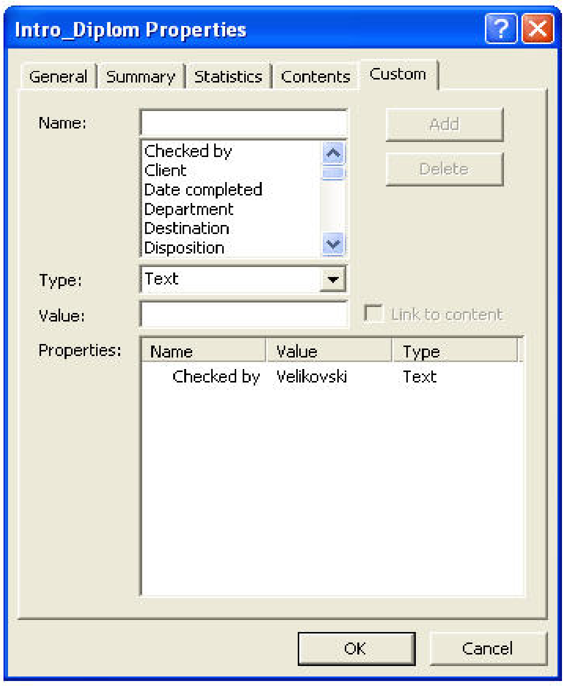
Figure 1.4 Last saved by, Revision number, Total editing time, Statistics (Slides, Paragraphs, Words, Bytes, Notes, Hidden slides, Multimedia clips, Presentation format)



Document contents:

Fonts Used
    Arial
    Times New Roman
Design Template
    Standarddesign
Slide Titles
    Slide 1
    Slide 2

Figure 1.5 Document contents (Fonts Used, Design Template, Slide Titles)



Intro_Diplom Properties

Name:

Checked by
Client
Date completed
Department
Destination
Disposition

Type: Text

Value:

Properties:

Name	Value	Type
Checked by	Velikovski	Text

Figure 1.6 Custom meta data (Checked by, Client, Date completed…)

The possibility of revealing all that meta data to a different than Microsoft® Office application will be a bigger step toward application integration, largely propagated in the current time. This meta data will be integrated into the metaXXXX.xml files, which the plug-in application will automatically generate and which can be further used for indexing and searching procedures. The actual content of the PowerPoint® slide (text, speaker's notes, and slide title) will be integrated into the description section "Notebook" of the Learning Objects (see figure 2.12 Meta indexing of a Learning Object). This will to some extend optimize the manual preparing of the Learning Objects (more information in Chapter 3).

A brief review of the technological stack will be taken in Chapter 2 and then a decision will be made, which elements of this stack will be used for building the plug-in. The actual programming will take place in Chapter 3. Installation and user manual will be explained in Chapter 4.

2 Technology Concept and Relevance

2.1 User Interface

2.1.1 Introduction

In this chapter, a discussion about the “face of a program”, the part that everyone sees - the user interface, will be held. Many programmers do not invest much time in the user interface design, meaning the code is what matters in an application and deserves the most of their attention[10]. However, thoughts about fonts, screens and form elements should be made in advance. The users do not pay much attention about the code because they do not see it. The only thing that can catch their eyes is the user interface with all its advantages and disadvantages. This chapter describes a set of guidelines and some examples that will help to explain the most of the user interface paradigm.

2.1.2 Understanding Human Cognition Processes

Cognition is what goes on in the heads of people when they carry out their everyday activities.[11] It engages cognitive processes, like thinking, reading, remembering, learning, seeing, writing and talking and many others. Norman[12] explains two general modes: experiential and reflective cognition. The experiential cognition is a state of mind in which people observe, act, and react to events around them effectively and naturally. It requires reaching a certain level of knowledge and commitment. Some examples include riding a bicycle, playing a computer game or reading a book. In contrast reflective cognition involves thinking, comparing, and decision-making. This kind of cognition is what guides to new ideas and creativity. Examples include designing a product, composing a song, and writing a book. Norman points out that both models are crucial for everyday life but that each involves different kinds of technological support.

Cognition has also been explained in terms of specific kinds of processes. These include:

- Attention – the process of concentration on selected things, at a point in time, from a range of possibilities available.

[10] See Siller et al. (1998), p.410.
[11] See Preece et al. (2002), p.74.
[12] See Norman (1994) p.14

- Perception and recognition – describe how information is acquired from the environment, via the different sense organs (e.g. eyes, ears, and fingers) and transformed into experiences of tastes, sounds, objects and events.
- Memory – the process of recalling various kinds of knowledge that allow humans to act suitably to the current situation.
- Learning – can be considered in terms of how to use teaching materials to understand and remember a given topic.
- Reading, speaking, and listening – are forms of language processing and have both similar and different properties.
- Problem solving, planning, reasoning, decision making – often involve conscious processes, discussion with others and the use of various kinds of artifacts.

Those cognition processes can be of great importance when transformed in the world of user interface design decisions and practices:

Attention:

- Information must be made outstanding when it needs attending to it.
- Techniques like animated graphics, color, underlining, ordering of items, sequencing of different information, and spacing of items should help to achieve this.
- Cluttering the interface with too much information should be avoided. This especially applies to the use of color, sound and graphics: there is a temptation to use lots of them; resulting in distracting and annoying perception rather than helping the user concentrate to relevant information.
- Interfaces that are plain in their design are easier to understand and use[13].

Perception:

- Graphical elements like icons should enable users to easily differentiate their meaning.
- Sounds and audio should be distinguishable so users realize what they represent.
- Audio text should be clear and distinguishable from the background sounds[14].

[13] See Preece et al. (2002), p.71.
[14] See Preece et al. (2002), p.68.

Memory:

- Users' memories should not be burden with complicated procedures for execution of tasks.
- User Interfaces should be designed to issue *recognition* rather than *recall* by using consistently placed objects, consistently menus and icons[15].

Learning:

- Use interfaces should design to encourage exploration.
- User interfaces should restrain and guide users to select appropriate actions.
- Dynamically linked representations and abstractions[16].

Reading, Speaking and Listening:

- The length of the text message boxes should be kept short and easily to understand;
- The length of speech-based menus or text instruction should be kept to a minimum.
- There should be opportunities for increasing the text on a screen for people who find it hard to read small text (e.g. the elderly generation)[17].

Problem solving, planning, reasoning, decision making:

- The system should provide additional hidden information that is effortless to access for users who wish to know more about how to carry out an activity more affectively[18].

It is important to note that many of these cognitive processes are co-dependent: several may be mixed up for a given activity. For example, when a student tries to learn material for an exam, she need to attend to the material, perceive, and recognize it, read it, think about it, and try to remember it. Therefore, cognition typically involves a series of processes. It is unusual for one to occur in isolation.

[15] See Preece et al. (2002), p.70.
[16] See Preece et al. (2002), p.78.
[17] See Preece et al. (2002), p.78.
[18] See Preece et al. (2002), p.81.

2.1.3 Definition of the Term “User Interface”

> “…An interface is the way to determine what the user can do with the system and how the system response to this interaction…”[19]

Or another one:

> “…The user interface to an interactive product such as software can be defined as the languages through which the user and the product communicate with one another…”[20]

Those simple descriptions gather the meaning of the User Interface (UI) being a mediator between a user and a program. In one direction the user interacts through his input and the system answers through its output. In the time of PC and particularly after the introduction of Apple® Mac OS and Microsoft® Windows® operation systems the era of the Graphical User Interface (GUI) has began. The paradigm of GUI is to accept input via electronic devices such as computer keyboard and mouse and answer with a graphical output on the computer monitor. And this is the widely spread way users interact with the computers nowadays. Although it is a straight forward definition making a good UI is not a trivial task.

“…A user interface is well designed when the program behaves exactly how the user thought it would…”[21] or saying it more comprehensive: “…A user interface is well designed when the program model conforms to the user model…”[22]

It is always preferable to know the user model first before building the program model, because it is easer to design a program model as to teach the user a new model to interact with a given system. It is the evolution way that has more chances of success than the revolution path of bringing a new UI model. But how a software developer knows what the user model looks like. As there is not only one user in this world the question should be how to know the user model of the target user group. And the answer is easy: an open communication with them. Only that kind of collaboration can gather this valuable knowledge at the early design state.

[19] Raskin (2001), p.18.
[20] Mayhew (1999), p.1.
[21] Spolsky (2001), p.8.
[22] Spolsky (2001), p.8.

2.1.4 Usability

The usability is not the only property of a user interface. In the contrary it has multiple components and is traditionally associated with following five usability attributes:[23]

- Learnability: A system should not demand much strength when learn it and an user should be able to swiftly start working with the system.
- Efficiency: A system should guarantee a high level of efficiency during usage, so that once an user has learned the system, advanced productivity is possible.
- Memorability: A system should be easy to keep in mind for a long period of time, so that a normal user is able to work with the system after some period of time, without having to learn it from the beginning.
- Errors: A system should be error resistant, when possible - few errors and easily recover from them. Further, fatal errors must not occur.
- Satisfaction: A system should be overall enjoyable to use.

Later on in 2001 Nielsen introduce an enlarge set of design principles:[24]

- Visibility of system status – users should be informed about what is going on, through providing appropriate feedback (i.e. status messages, percentage message of the work been done, and so on) within reasonable time.
- Equivalence between system and the real world – it is crucial to speak the users' language, using concepts, phrases and words familiar to the them, rather than technical - oriented terms.
- User control and freedom – there should be always possibilities for the users to easily come from places they unpredictably find themselves, by using clearly marked "exits points".
- Consistency and standards – the users should not be taken into position to speculate whether different actions, words or situations mean the same thing.
- Error recognition, diagnose and recover – application designers should uses a simple language to explain problems to the users and suggest ways of solving them.
- Error prevention – where possible errors should be prevented of happening if possible at all.

[23] See Nielsen (1993), p.26.
[24] See Nielsen (2001).

- Recognition rather than recall – options, objects and actions should be made visible.
- Flexibility and efficiency of use – automatons processes that are invisible to beginners, but allow the expert users to carry out task more quickly, should be provided.
- Aesthetic und minimalist design – using information that is irrelevant or rarely needed should be avoided.
- Help and documentation – a set of help information that can be easily searched, should be provided and it should be in a form of concrete steps that can easily be followed.

Design and usability principles have also been refined into even more specific prescriptions called rules.[25] Those are in form of guidelines that should be followed.

The *usability* as a term has so many explanations that try to figure all different aspects of it. That can be to some extend confusing. Those explanations are often used interchangeably and in different combinations. Terms like usability design principles, usability heuristic or design concepts can be heard often but trying to address one and the same thing. The key is in understanding how to use the different levels of guidance (see Table 2.1). Goals refer to the high-level usability aims of the system. Principles refer to general guidance intended to inform the design and evaluation of the system. Rules are particular prescriptions that must be followed. Heuristic is a general term used to refer to design and usability principles when applied to a design problem.

Concept	**Level of guidance**	**Also sometimes called**	**How to use**
Usability goals	General		Setting up usability criteria for assessing the acceptability of a system (e.g. "How long does it take to perform a task?").
User experience goals	General	Pleasure factors	Identifying the important aspects of the user experience (e.g. "How to make the interactive product fun and enjoyable?").
Design principles	General	Heuristic when used in the practice. Design concepts	As reminders of what to provide and what to avoid when designing an interface (e.g. "What kind of feedback are you going to provide at the interface?").
Usability principles	Specific	Heuristic when used in the practice	Assessing the acceptability of interfaces, used during heuristic evaluations (e.g. "Does the system provide clearly marled exits?").

[25] See Preece (2002), p.27.

Rules	Specific		To determine if an interface adheres to a specific rule when being designed and evaluated (e.g. “Always provide a backwards and forwards navigation button on a web browser”).

Table 2.1 Different terms for usability[26]

Problems can arise when trying to apply more than one of the design principles in interaction design so that trade-offs can arise between them. For example, the more a developer tries to constrain an interface, the less visible information the user becomes. Some similar effect can arise when trying to apply a single design principle. Consistency is another design pattern that can be challenging to apply. Trying to design an interface to be consistent with some principals can make it inconsistent with some others. Some time inconsistent interfaces are actually easier to use than consistent interfaces. A trade-off, however, is that it can take longer to learn such an interface but in the long run can make it easier to use.

2.1.5 Designing Effective Forms

Forms are the building blocks of a graphical user interface. With the help of integrated development environment (IDE) tools like Eclipse IDE[27], NetBeans[28], Borland CodeGear[29] or Microsoft® Visual Studio[30], designing a form is simple; doing it well is not so easy. A good form design takes more than just inserting programming events and controls. To make a well-designed form, the developer should understand the form’s purpose, meaning the way it is going to be used, and its relationship with the rest of the program.
As the number of controls, situated on a form increase, the more important it is to keep them organized. An example of a bad form design is shown in Figure 2.1.

[26] Table taken from Preece (2002), p.28.
[27] See Eclipse - an open development platform [Last visited 17.08.2007]
[28] See NetBeans IDE [Last visited 17.08.2007]
[29] See CodeGear [Last visited 17.08.2007]
[30] See Visual Studio 2005 [Last visited 17.08.2007]

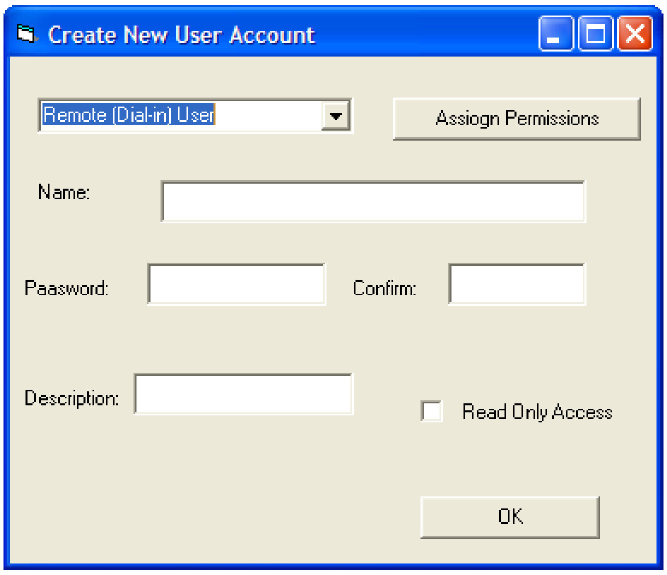


Figure 2.1 A messy form design[31]

It looks as if the controls are chaotically placed on the form. They are not labeled, lined up, or consistently sized. Much better approach is shown in Figure 2.2. Every GUI object like labels, frames, and lines are added to group semantically related controls.

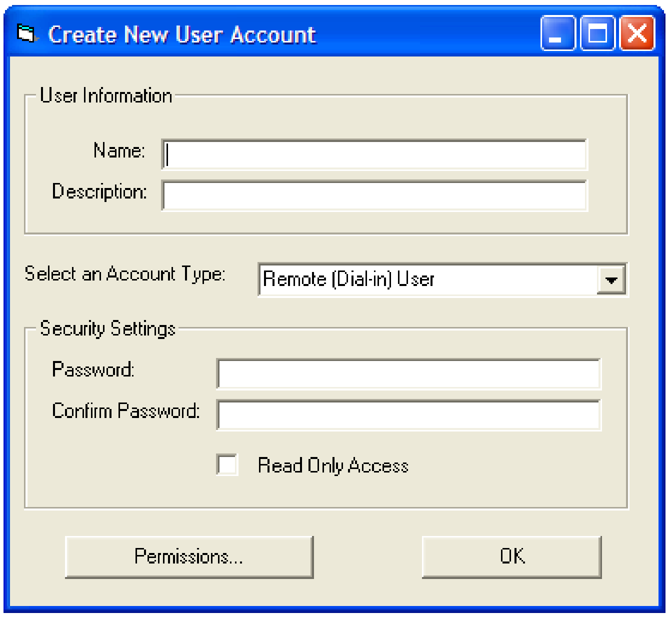


Figure 2.2 A better design for the same form[32]

[31] See figure in Siler et al. (1998), p.410.
[32] See figure in Siler et al. (1998), p.411.

Both figures show “working” applications, but the second one has a more visually organized overall appearance. To relieve excessive elements inconsistency many user interface designers use bounding boxes to more strongly group subsets of the window content.

2.1.6 Software-engineering tools

Experienced programmers sometimes build user interfaces with programming languages such as Java, C#, or C++, but this approach is giving a way to using programming tools that are specialized in developing user interfaces or web access. Choosing among them is sometimes a complex and confusing task, due to the lack of uniform technology used to describe the tools and their features.[33]

There are a large number of tools available for building UI. Table 2.2 lists the four software layers that can be used to build a UI and their associated interactive tools.

Software Layers		**Visual Tools**	**Examples**
1	Application	Model-Based Building Tools	Microsoft® Access, Sybase PowerDesigner
2	Application Framework/ Specialized Language	Conceptual Building Tools	Macromedia Director, Tlc/Tk, Microsoft® MFC
3	GUI Toolkit	Interface Builder	Eclipse, Borland JBuilder, Microsoft® Visual Studio
4	Windowing System	Resources Editor	Microsoft® Win32/GDI+, Apple Quartz, X11 Windowing System

Table 2.2 The four software layers available to build a user interface[34]

The higher levels tools (layer-4) are interface generators, sometimes called user-interface management systems or model-based building tools. Most if not all of the applications can be built quickly using these visual tools. Unfortunately, these tools are currently available only for a small class of application, such as database front-ends (Microsoft® Access[35], Sybase PowerDesigner[36]).

[33] See Shneiderman et al. (2005), p.189.
[34] Schneiderman et al. (2005), p.190.
[35] Microsoft® Office [Last visited 17.08.2007]
[36] PowerDesigner [Last visited 17.08.2007]

Layer-3 tools include specialized languages or application frameworks. These are software architecture tools specially designed for building graphical user interfaces (GUIs). Compared to layer-4 tools, they provide almost no support for the nongraphical part of the application. At this layer, a key distinction is how extensively the software-engineering tool uses convenient visual programming, a relatively simple scripting language (event or object oriented), or a more powerful general-purpose programming language.

The terminology for GUI toolkits (layer-2) varies depending on the vendor. Popular terms for these toolkits include Rapid Prototyper, Rapid Application Developer, User Interface Builder, and User Interface Development Environment. This layer provides software libraries and widgets as building blocks but requires extensive programming to connect these components to each other and to the non-GUI part of the application.

The layer-1 windowing system tools require extensive programming by experienced software engineers and offer little support from interactive tools.

Given this list of layers, the obvious recommendation is to use the highest available. Together with the increased support through the developing tool also more constraints are coming: automatic application generators will quickly build stereotyped applications that are cheaper and easy to developed but offer very little variety or adaptability. Finding the right tool is a trade-off between six main criteria:

- Part of the application built using the tool. Some tools only support building the presentation part of the application; others also help with low-level interaction, and some support general programming mechanism usable in other parts of the application as well.
- Learning time. The time regarded to learn the tool varies.
- Building time. The time required to build a user interface using the tool varies.
- Methodology imposed or advised. Some tools strongly recommend a specific methodology for building the application, such as building the visual part first and connecting it to the remainder of the application afterwards, whereas other tools are more flexible.
- Communication with other subsystems. Applications frequently use databases, files located on the Web, or other resources that, when supported by the building tool, simplify the development.

- Extensibility and modularity. Applications evolve, and new applications may want to reuse parts of existing applications. Supporting the evolution and reuse of software remains a challenge.

Level-4 tools and application frameworks inherently promote good software organization, but the others usually lead to poor extensibility and modularity. It is also important to mention the two possible scenarios that can come in a software project. High percent of all projects are to implement additional features to existing applications and only few are for developing a completely new piece of software. In those cases there is strong presser to use the tools that the old application has been developed with. To some extend that makes sense but at the other hand this is an obstacle to the know-how building of the IT industry. The one that can help is the reengineering paradigm which also has its strong and weakness points.

In the next chapter the problem of choosing the right programming language for a given project will be discussed.

In this chapter a brief overlook of interaction design an in particular user interface design was introduced. It was pointed out how the idea of usability is essential to the idea of interaction design. This was explained in some detail, describing what it is and how it is operationalized to access the suitability, effectiveness, and quality of interactive products, in particular software applications. A number of high-level design principles were also discussed that present diverse forms of guidance for interaction design.

2.2 Decision about the Programming Language

2.2.1 Introduction

The choice of a programming language for building a plug-in for an existing application depends mostly on the application itself. As eduWEAVER (respectively ADVISOR®) installation is only available for the Windows® platform, the programming language to be used should also come from the Microsoft® programming language family (Visual Basic, VBA, C# etc.). The fact that an automation process (PowerPoint transformation) will be started, speaks also for a language from the Visual Basic family, like Visual Basic for Application (VBA).

Visual Basic for Applications (VBA) is a hosted language and part of the Visual Basic (VB) family of development tools.[37] When hosted in other applications such as Word® or PowerPoint®, VBA is using a technology called *automation*, in sense of interacting with the host application's object model, which implements the VBA interface.

To solve the problem of customizing complex applications such as Excel®, Word®, Access®, PowerPoint® and an increasing number of many other applications from Microsoft® or other software firms, "…VBA allows the developer to provide solutions that take advantage of sophisticated components that have been tried and tested…"[38]. The VBA is a language that collaborates with the many different objects via the host application's object model. VBA makes applications extensible. Its support for OLE (see chapter 2.2.4) automation makes VBA an outstanding tool for developing plug-ins application for already existing ones.

In the following sub-chapters a quick discussion of the technologies and paradigms in the world of Microsoft programming environment will take place. Terms like COM, OLE and Microsoft® Office object model will be explained. This chapter will end with a short summary of the differences between VB and VBA to make a bridge to the next chapter where the Microsoft® PowerPoint® Application Object and its accessibility through VBA will be discussed.

[37] See Lomax (1998), p.3.
[38] Lomax (1998), p.3.

2.2.2 The Component Object Model (COM)

The Component Object Model (COM) is Microsoft's contribution to the object model world. In addition it is also a Microsoft specification that describes how to create reusable objects in the Win32 programming environment.[39] COM is binary standard. That makes it possible for any programming language (having the proper facilities) to create COM objects. It provides the foundation for OLE (see chapter 2.2.4).
COM is also an industry standard object model that specifies the following:

- Object definition – defining the objects structure and the memory management of the objects.
- Life cycle management – defining how objects are created and destroyed.
- Inter-object communication protocols – defining the component interoperability mechanism, particularly how the objects communicate and expose their functionality.

COM is a strict set of rules governing basic object structure and semantics. COM is the object model on which the OLE technologies stand. Object Linking and Embedding (OLE) technologies are set of reach system-level services supplied by objects that adhere to COM rules.

OLE is a set of services supplied by objects that conform to the COM specification. The primary responsibility of the COM is to ensure that software components behave in a well-known and consistent manner without constricting how programmers implement different components. The COM accomplishes this by defining a binary interface for objects that is independent of any programming language. Objects confirming to the COM are collectively called component objects. Every feature of OLE depends on the COM to provide basic inter-object communication.
Since OLE is built on top of COM, all of OLE's services are supplied by objects that adhere to this specification. When a program accesses any of OLE's services, a communication with a specific COM object (or set of COM objects) is carried out.

[39] See Freeze (2000), p.4.

2.2.3 A COM Client-Server Model

One of the possibilities of COM is to support a simple client-server model. Objects called *servers*[40] provide functionality to objects called clients. In that software architecture *servers* are always COM objects, i.e., they are objects that conform to the Component Object Model (see figure 2.3).

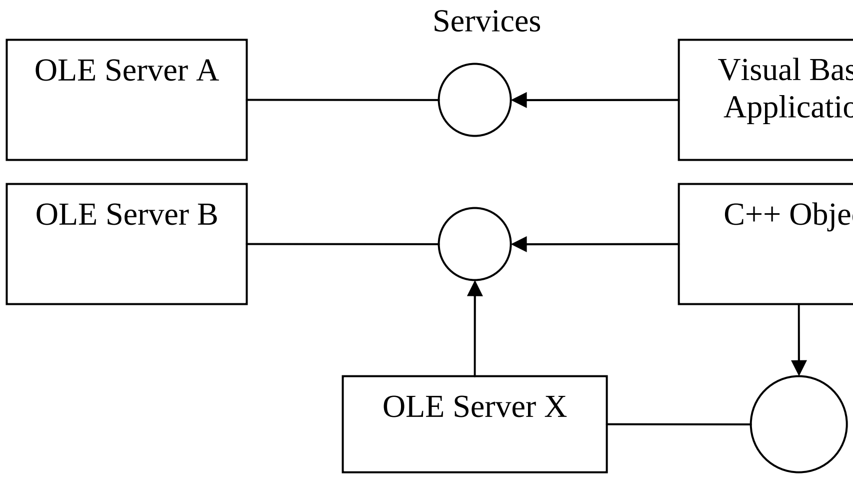


Figure 2.3 A COM Client- Server Model[41]

Clients, on the contrary, have the choice to be implemented with or without the help of COM objects. This means that some clients may be simple Java Beans, C++ objects, Visual Basic Applications, C# objects, etc.
A COM object may be a client and a server at the same time. It only depends on the relations between the objects under consideration.

2.2.4 OLE Architecture and Automation

Object Linking and Embedding (OLE) is an architectural framework, built on top of the Component Object Model (COM), which supplies a set of object-based services to clients.[42]
OLE is also a system-level, object-based, unifying technology that helps to implement application integration through a set of coordinating system libraries (DLLs). OLE includes

[40] See Puopolo (1997), p.17.
[41] Figure taken from Puopolo (1997), p.17.
[42] See Puopolo (1997), p.57.

several distinct technologies that use each other's services to provide an object-enabling system. The real power of OLE comes from the synergy those useful services.

Automation is one of the most powerful features of OLE. It is the mechanism through which an object exposes its methods (functionality) and properties (characteristics) to other objects and applications. An object that exposes its functionality is called an *automation server*.[43]

An *automation controller* is an object or application that uses or directs the actions of the automation server. *Automation servers* exist to service automation controllers' requests and directions. IDispatch is a standard COM interface that specifies automation functionality (see figure 2.4).

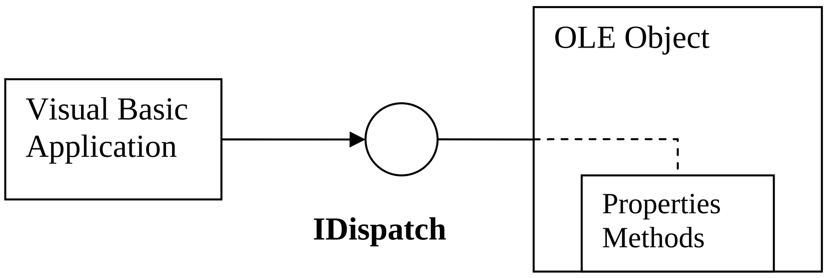


Figure 2.4 The mediator role of the IDispatch[44]

Automation servers implement the IDispatch interface. [45] *Automation controllers*, at the other hand, communicate with the *automation server* through its IDispatch interface. An automation controller does not directly call the functions implemented by the *automation server*. On the contrary, the *automation controller* uses member functions in the IDispatch interface to indirectly call functions in the *automation server*. The IDispatch interface, like all of *Automation*, was developed as part of Visual Basic so that it could be used specifically to automate applications such as Microsoft Office family.

Automation servers[46] are mostly written in C or C++, so they are usually fast, powerful objects that supply robust functionality. *Automation controllers* use the services provided by the *automation servers*. That is why they do not need to be as fast or robust as their associated servers and are generally developed in languages like LotusScript and Visual Basic, where ease of development and flexibility are most important.

[43] See Puopolo (1997), p.65.
[44] Figure taken from Puopolo (1997), p.65.
[45] See Rogerson (1997), p.279.
[46] See Puopolo (1997), p.66.

2.2.5 Differences between VBA and VB

There are some major differences that focus on usage of both programming languages. This is because VB is a complete Rapid application development (RAD) environment (Microsoft® Visual Studio for Visual Basic) that features a set of user interface components and relies on VB as its programming language.

Some cardinal differences between the two programming languages are though presented:

• VB applications are normally compiled into native code executables (*.exe), whereas VBA applications always depend on their host application for execution and are always interpreted.

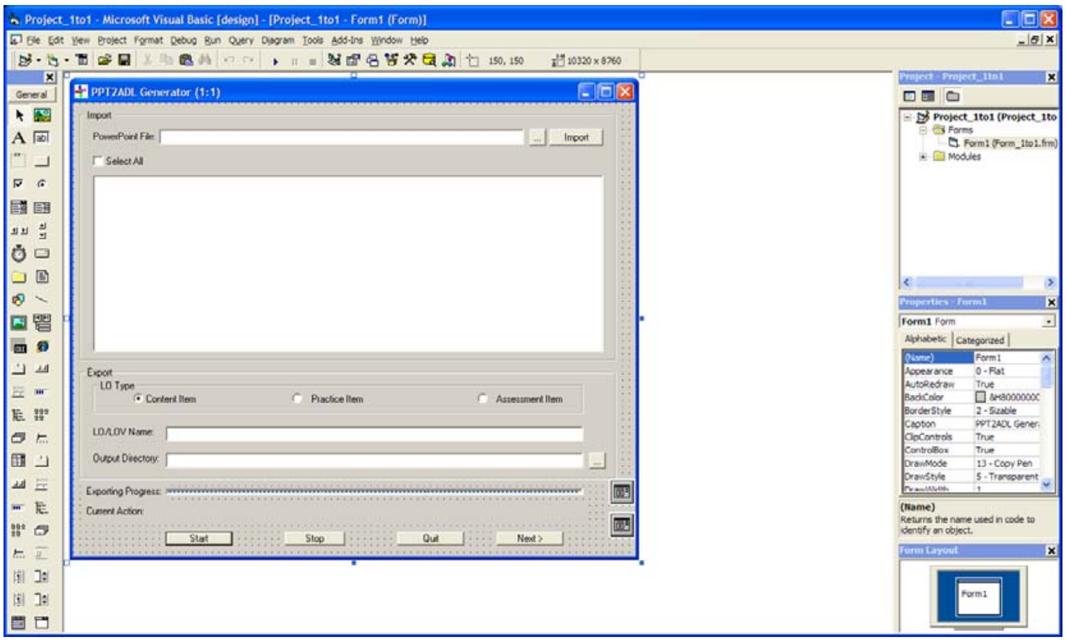
• VBA applications are that's way slower than VB ones because of the missing true compiler.[47]

• The programs created by the two products fulfil different requirements. VB helps creating standalone applications. On the contrary VBA is used to create applications that enhanced the possibility of the host applications (e.g. Access® or PowerPoint®).

• Other than VB, VBA allows the programmer to write code for multiple platforms. Versions of VBA are available not only for Windows® platforms but also for Apple Macintosh, Alpha RISC and many others.

Taking these differences into consideration, particularly the possibility to use PowerPoint® features from outside the Microsoft® Office environment, for building the eduWEAVER Plug-in a pure Visual Basic environment will be used as a developing platform (Microsoft® Visual Studio for VB6 see figure 2.5).

[47] See Doberenz et al. (1999), p.60.



(Name)	Form1
Appearance	0 - Flat
AutoRedraw	True
BackColor	&H8000000C
BorderStyle	2 - Sizable
Caption	PPT2ADL Gener
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	5 - Transparent

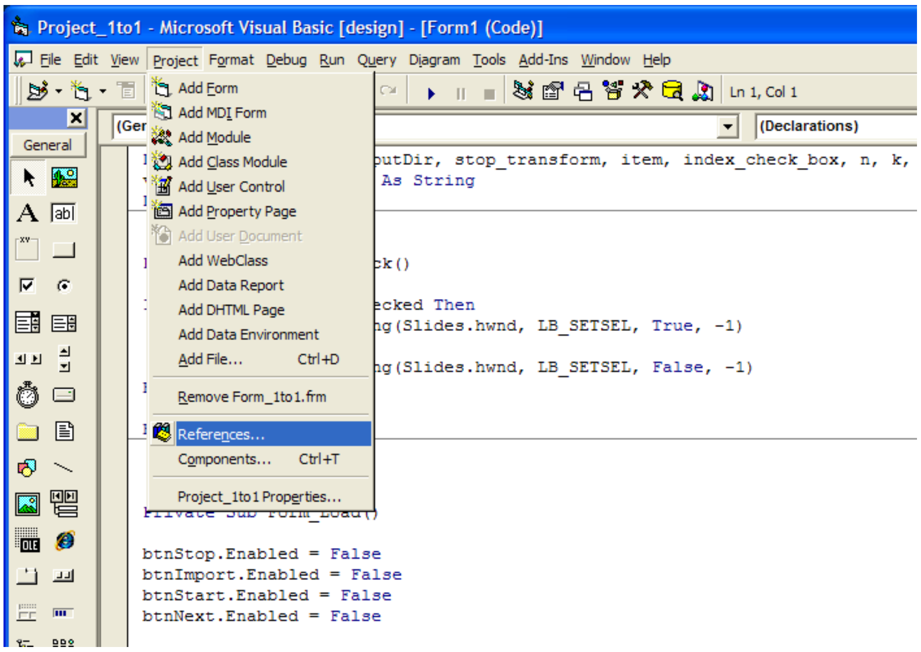
(Name)
Returns the name used in code to identify an object.

Figure 2.5 Microsoft® Visual Studio for VB6

The PowerPoint application objects will be communicated through VBA code embedded in the VB application. The output will be a compiled .exe, which purpose will be to enable eduWEAVER interact with PowerPoint® application on automation base.

The problem is that a Power Point Application should be installed on the target computer in order for the eduWEAVER plug-in to interact with it. But that should not be a critical point as all the teaching staff use the Microsoft® Office family and the plug-in application will be only used to transform PowerPoint® presentations into e-Learning materials. But how the plug-in will know this fact as it is an autonomic application. The answer is “early binding”.

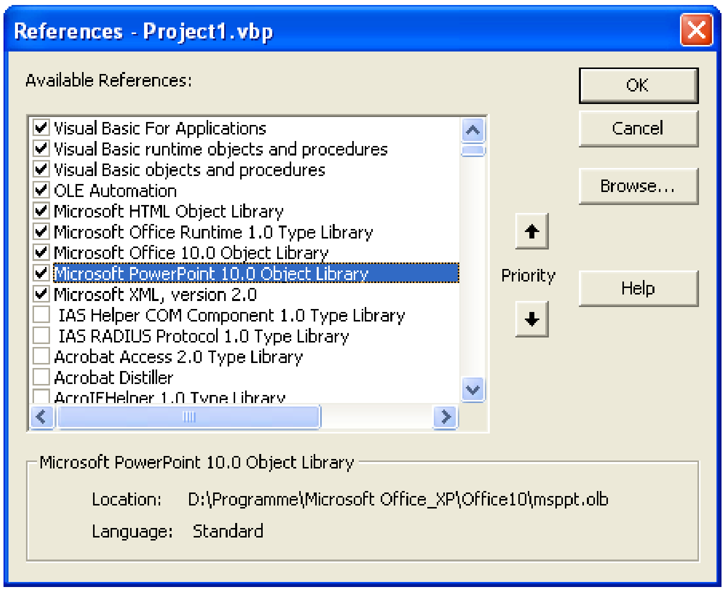
The term “early binding” refers to the procedure of adding an object reference to the VB project by using the References dialog (in a stand alone Visual Basic IDE or in a Visual Basic Editor in any Microsoft® Office Application), which is shown in figure 2.6. The References Dialog can be reached by selecting Project -> References option from the menu.



```
putDir, stop_transform, item, index_check_box, n, k,
 As String

ck()

ecked Then
g(Slides.hwnd, LB_SETSEL, True, -1)

g(Slides.hwnd, LB_SETSEL, False, -1)

Private Sub Form_Load()

btnStop.Enabled = False
btnImport.Enabled = False
btnStart.Enabled = False
btnNext.Enabled = False
```

Figure 2.6 Selecting the References option from the Project Menu

After that all OLE Automation Servers registered on the machine are shown in the list. (See figure 2.7)



Available References:

- [✓] Visual Basic For Applications
- [✓] Visual Basic runtime objects and procedures
- [✓] Visual Basic objects and procedures
- [✓] OLE Automation
- [✓] Microsoft HTML Object Library
- [✓] Microsoft Office Runtime 1.0 Type Library
- [✓] Microsoft Office 10.0 Object Library
- [✓] Microsoft PowerPoint 10.0 Object Library
- [✓] Microsoft XML, version 2.0
- [ ] IAS Helper COM Component 1.0 Type Library
- [ ] IAS RADIUS Protocol 1.0 Type Library
- [ ] Acrobat Access 2.0 Type Library
- [ ] Acrobat Distiller
- [ ] AcroIEHelper 1.0 Type Library

Microsoft PowerPoint 10.0 Object Library

Location:	D:\Programme\Microsoft Office_XP\Office10\msppt.olb
Language:	Standard

Figure 2.7 The References dialog

After a reference is added to the project, an early bound interface can be created by using the *Private*, *Public*, *Friend*, or *Dim*. Then the *Set* statement is to be used to set an object reference to the local variable (see listing 2.1).

```
Dim objPres as Presentation
Set objPres = PowerPoint.Application.ActivePresentation
```

Listing 2.1 Early Binding

The *New* statement (see next chapter) can be used with the *Private*, *Public*, *Friend*, or *Dim* statement (see listing 2.2).

```
Dim objPres As New Presentation
```

Listing 2.2 The "new" statement

Because the project has a reference to the object library in advance, binding to the object can be arranged at "early" compilation time. That makes the developing process more comfortable and efficient.

2.2.6 VBA and the Microsoft® PowerPoint® Application Object

As mentioned above, when a Microsoft® Visual Basic® for Applications (VBA) or pure Visual Basic (VB) code is written to work with Microsoft® PowerPoint®, the Application object[48] is to be referenced at first. In case VBA code is written within PowerPoint®, the Application object is automatically created. If automating PowerPoint® from some other application via VB, a PowerPoint® *Application* object variable should be created and then an instance of PowerPoint® should be started. This instance of the *Application* object can include any number of open *Presentation* objects. Figure 2.8 shows a part of the PowerPoint® application object that is relevant for the development of the eduWEAVER plug-in application. This model should be used to extract all the information that is relevant in the process of e-Learning.

[48] See Microsoft Corporation (2001), p.141.

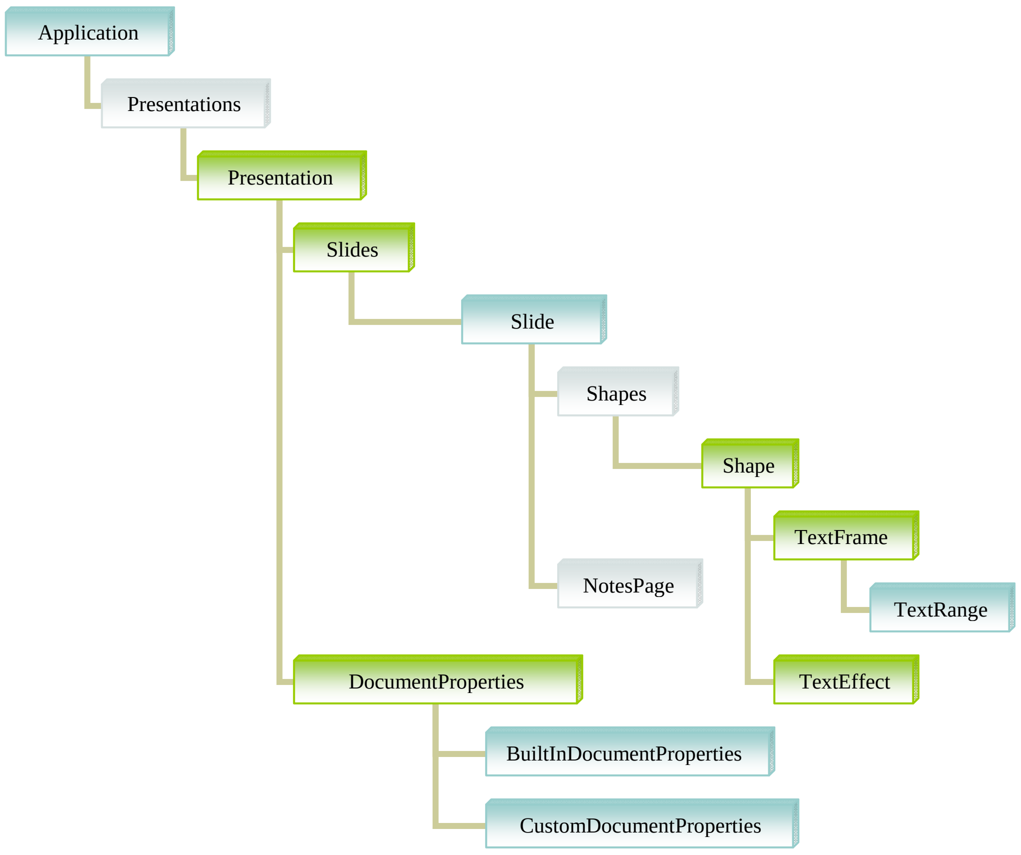


Figure 2.8 A relevant part of the PowerPoint® Application Object Model

2.2.6.1 Working with the *Application* Object

Like in other Microsoft® Office applications is the *Application* object the most important one[49]. It is the place where the VB/VBA starts interacting with the whole Microsoft® Office application family. It is so important like starting PowerPoint® or any other Microsoft® Office application on the PC using its application icon. For example a new PowerPoint® application can be invoked through the following code (see listing 2.3):

```
Dim ppApp      As PowerPoint.Application
Set ppApp = New PowerPoint.Application
```

Listing 2.3 A new PowerPoint application

[49] See Seelhofer (2003), p.465.

Some of the properties of the application object are: *ActivePresentation*, *ActiveWindow*, *AddIns*, *CommandBars*, *FileSearch*, *Windows* and many others.
Some of the properties are *Visible*, *WindowState*, *Name*, *Path*, *Caption* and so on.
Some of the important methods are *Quit* and *Run*.
One of the most important properties of the *Application* – Object is the *Presentations* - Object which is revealed in the next chapter.

2.2.6.2 Working with the *Presentation* Object

The *Presentation* object and the slides it contains are some of the most important objects when working with PowerPoint® object model through VB/VBA. They are the placeholder for the content being representing through a PowerPoint® presentation. There are four ways to access a *Presentation* object:[50]

- By referencing to an open and active presentation.
- By using the file name of an open presentation.
- By using the *Caption* setting of the PowerPoint® Application Window object that contains an open presentation.
- By using the value of the index of an open presentation.

The examples below illustrate the four different ways to set a reference to an already open presentation (see listing 2.4).

```
Dim prsPres As PowerPoint.Presentation
      '1 referencing to an open and active presentation.
Set prsPres = ActivePresentation
      '2 using the file name of an open presentation.
Set prsPres = Presentations("test.ppt")
      '3 using the Caption setting of the PowerPoint® Application Window
      object that contains an open presentation.
Set prsPres = Presentations("test")
      '4 using the index of an opened presentation
Set prsPres =Presentations(1)
```

Listing 2.4 Set a reference to an open presentation

[50] See Microsoft Corporation (2001), p.141.

As shown above there are methods to reference an open presentation. There is also a method to open a presentation already saved to disk and at the same time to create a reference to that presentation. That is done through a *Presentations* collection's *Open* method (see listing 2.5).[51]

```
Dim ppApp As PowerPoint.Application
Dim prsPres As PowerPoint.Presentation

Set ppApp = New PowerPoint.Application
Set prsPres = ppApp.Presentations.Open("C:\test.ppt")
With prsPres
   'Some coding...
End With
```

Listing 2.5 Open the PowerPoint.ppt presentation

The most important property of the *Presentaion* object is the *Presentaion.Slides property*, which will be reviewed in the next chapter.

2.2.6.3 Working with the *Slide* Object

Normally a PowerPoint® presentation is a set of slides. Every one of them usually contains text followed by graphics or some other visual effects like transformation animations. The Presentation object has a *Presentation.Slides* property that returns the *Slides* collection.[52] It can be used to access an existing slide or add new slides to a given presentation. In the collection every individual slide is represented by a *Slide* object. PowerPoint® automatically automatic naming convention follows the pattern: <*Slide*> *X*, where *X* is a number indicating the location of the slide in the presentation at the time point it was added to the Slides collection. In addition a particular name for a slide can be specified by using the *Slides(X).Name* property.

There are again four possible ways to access a particular *Slide* object in a *Slides* collection:[53]

[51] See Microsoft Corporation (2001), p.142.
[52] See Microsoft Corporation (2001), p.146.
[53] See Microsoft Corporation (2001), p.147.

- By using an index value, indicating the location of the slide in the *Slides* collection.
- By using the slide name.
- By the slide's *SlideID* property using Slides collection's *FindBySlideID* method.
- By using the *SlideRange.SlideIndex* property (indicating the currently selected slide).

Some of the *Slide* object methods are: Add(), Delete(), Cut(), Copy(), Paste() and so on. The most important property of the *Slide* object are the *Shapes* objects, reveal in the next chapter.

2.2.6.4 Working with the *Shape* Object

Every slide in a presentation consists of one or more *Shape* objects. Every artefact like a picture, a title, text, a table, an AutoShape, or other content, everything that can be put on the slide is represented by a *Shape* object.

A *Shape* object on a slide can be referenced in two ways:[54]

- By using the shape's index in the shape’s collection.
- By using the name of the shape.

In case that many shapes on a slide have to be referenced to, a good choice is to use the *Shapes.Range* method of the *Shapes* collection, which returns a *ShapeRange* object. It consists of the shapes specified in the *ShapeRange.Index* method's argument. By means of not specifying any *Index*, the *Range* method returns a collection of all the shapes on a slide.

2.2.6.5 Working with *TextFrame* Object

All *Shape* objects on a slide that support text have a *TextFrame* property, which can be used to return a *TextFrame* object. To determine if a shape supports the use of a text frame, the *Shape.HasTextFrame* property can be used. Analogically each *TextFrame* object has a *HasText* property, which can be used to determine if the text frame contains text.

The *TextFrame* object has also a *TextRange* property, which can be used to return a *TextRange* object.[55] *TextFrame.TextRange.Text* property can be used to specify, determine and modify the text associated with a *Shape* object.

[54] See Microsoft Corporation (2001), p.149.

There is one *Shape* object that can not be referenced directly by using the *TextFrame* or *TextRange* objects. The *TextEffectFormat* object is returned by *TextEffect* property and contains the methods to work with *WordArt* shapes. To read the text in a *WordArt* shape, the *TextEffectFormat.Text* property is to be used. For example, the following code (see listing 2.6) reads the text of an particular *WordArt* shape on the fifth slide of the current presentation and put it out in a message box.

```
With ActivePresentation
   strExistingText =.Slides(5).TextEffect.Text
 MsgBox strExistingText
End With
```

Listing 2.6 Working with the TextEffect property of the Shape object

Both *TextRange.Text* and *TextEffect.Text* properties are necessary for programming the plug-in application, as being one of its main features of extracting text from a PowerPoint® presentation and writing it in an .xml and .adl files.

The next main chapter will reveal why and how XML can help dividing content from presentation and at the same time wrapping the content with context (meta data).

In this chapter a brief overlook of the possible programming language for developing the plug-in application was introduced. Technologies like COM, OLE and programming languages like Visual Basic and Visual Basic for Application were checked about their relevance for the developing process. The main idea of this master thesis was to program a plug-in that should operate in Windows® environment, like eduWEAVER and PowerPoint® do. It was a clear decision to use all technologies described above as they were developed and maintained by the Microsoft® Corporation and fully integrated in the Windows® world.

The main subject of the next chapter will be the introduction of XML and its technological relevance for developing the eduWEAVER plug-in application.

[55] See Microsoft Corporation (2001), p.152.

2.3 XML – Dividing Content from Presentation

2.3.1 Introduction

One of the main purposes of the “backloading” process of the plug-in application will be to extract all textual informational artefacts of a Power Point® presentation. The second process will be to save that information in generic way not bothering about how its representation will look like. Talking about that kind of separation is inevitable to talk about XML. The following chapter will make a brief review of the XML standard and its implementation possibilities.

2.3.2 XML (eXtensibleMarkup Language)

XML, the eXtensible Markup Language[56], is a World Wide Web Consortium[57] (W3C) standard for document markup. It classifies a generic syntax used to mark up data with simple, human-readable tags, in contrary to the Hyper Text Markup Language[58] (HTML) of the Internet. It provides a standard format for computer documents. XML “…is flexible enough to be customized for domains as diverse as web sites, electronic data interchange, vector graphics, genealogy, real estate listing, object serialization, remote procedure call, remote procedure calls, voice-mail systems, and more…”.[59]

XML is also a *meta* markup language. “…It is a language used to describe and define other languages…”.[60] Data is included in XML documents in form of strings. The data is surrounded by a human readable text markup, indicating the data inside the XML file. This basic set of data and markup is defined as an element. The XML specification sets the exact syntax the XML markup must abbey. Some of the rules determine, which elements are delimited by tags, what these tags look like, what names are acceptable for elements, where attributes are placed, etc. The fact that a specific grammar is being defined allows the development of XML parsers, which can universally read any XML file. Documents that

[56] See XML.[Last visited 16.08.2007]
[57] See World Wide Web Consortium.[Last visited 16.08.2007]
[58] See HTML 4.01 Specification.[Last visited 16.08.2007]
[59] Harold (2002), p.3.
[60] Moseley (2007), p.53.

satisfy the XML grammar are defined as being well-formed. Documents, witch are not well-formed will not be accepted by any XML processors.

XML is a *meta* markup language, meaning it does not have a fixed set of tags like HTML does. On contrary, XML allows developers to classify the elements they need and in this way *extend* the language vocabulary. The letter X in XML stands for *eXtensible*. This means that the XML language can be extended and it is not stacked to particular markup language syntaxes. It is possible that every industry can issued its own set of mark-up which best suites its needs.

To guarantee interoperability, organizations involved in XML business usage can agree to use only certain set of tags, defined as *XML applications*. An *XML application* is the usage of XML in a particular domain like financial data or vector graphics.

The XML markup in a document describes the structure of the document. It shows the relation between the elements of the document. In a well-designed XML document, the markup also expresses the document's semantics. For example, the markup can indicate an element as a price or a spare part or a book title and so on. In well-designed XML applications, the markup should not include any kind of information about how the document should be displayed. As said before, XML is a markup language that describes the structure and semantic of the data and not its presentation. That is why XML is the best choice when unwrapping content from its presentation.

XML markup have the freedom to be extended, but this fact leads to some constrains. A strong rule without compromise is that every XML document must be well-formed. Among many others there are a number of rules, including the following: [61]

- Every opening start-tag must have its matching closing end-tag.
- Elements may be nested, but can not overlap.
- There must be exactly one root element.
- Attribute values must be quoted.
- An element may not have two attributes with the same name.

[61] See Harold (2002), p.24.

- Comments and processing instructions may not appear inside tags.
- Unescaped “<” or “&” signs must not occur in the character data of an element or attribute.

Well- formedness errors should be reported by every XML parser reading an XML document. The parser is not allowed to try to fix the document, because it can not guess what the author of the XML document really meant. It should only return an error. It is good programming practice, before publishing an XML document to check it for well-formedness.

An XML document instance can be documented in a schema and lately compared to that schema. If it matches the schema it is said to be a valid XML document or respectively invalid if it do not match the schema. So it is obviously that validity depends on the schema on which the XML document is compared to. The validation is an option. For many purposes it is enough for an XML document to be simply well-formed.

The mostly used schema language that is defined by the XML 1.0 specification is Document Type Definition (DTD). A DTD is organized as a list of the entire legal markup that can be included in a document. It also specifies where and how the XML elements (node, sub-node, attributes) may be included. It also represents the model of the document it is attached to. DTDs are optional in XML applicatons. These DTDs can be included in the XML document it describes, or alternatively an XML document can referenced to it at an external URL.[62] Those external DTDs can be shared by different organisations. This practice brings advantages and disadvantages to the using of DTDs. If the DTD is being only referenced, changes made to the original DTD are automatically available to all documents referencing to that DTD. On the other hand, backward compatibility is not guaranteed when a DTD modification. Incompatible changes can make XML documents invalid.

2.3.3 DOM and SAX

DOM is the standard representation for HTML and XML documents. This has been a W3C Recommendation since 1997 and is well established. The Recommendation declares that the DOM is an API that defines the logical structure of documents and the way a document is

[62] See Harold (2001), p.211.

accessed and manipulated programmatically.[63] The main idea after DOM is to define generic API and leave the implementation to the developers of parsers. Application developers who use those parsers will expect them to enable the access to the documents structure following the DOM standard. DOM gives an access to the information stored in an XML document by a hierarchical object model. That means that DOM creates a tree of nodes (based on the structure of the XML document). The DOM standard is not limited only to XML. It can also be used to save the structure of a HTML document.[64] The information within can be then accessed through this tree of nodes. Figure 2.9 illustrates this process.

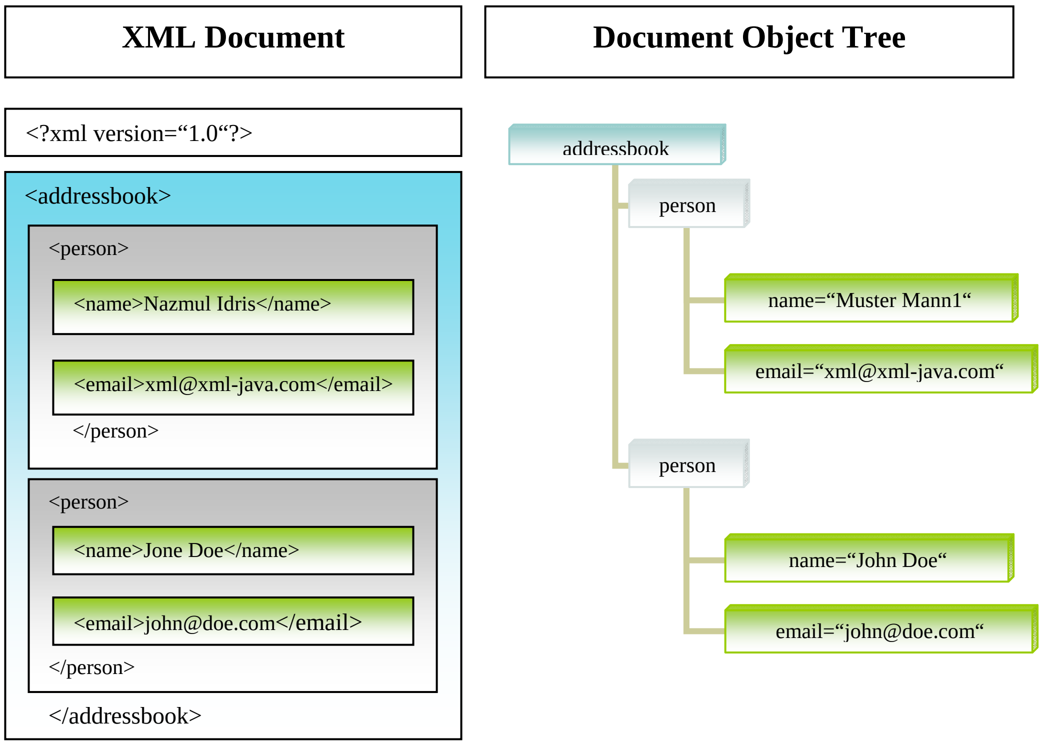


Figure 2.9 Hierarchical structure of a document object (DOM)[65]

DOM creates a hierarchical tree of nodes when creating a Document object for a particular XML document. An average DOM parser does almost everything automatically like: reading the entire XML document in the memory. After that follows the creation of a programming object model on top of it and then giving a reference to this object model for further manipulation.

[63] See Bates (2003), p.276.
[64] See Knobloch et al. (2001), p.38.
[65] Figure taken from Idris. [last visited 04.05.2007]

Alternatives to the DOM approach are in some cases needed, particularly in cases of huge data transfer and manipulation. This is the reason why Simple API for XML (SAX) was developed.[66] SAX was created by Dave Megginson and his fellow subscribers to an email list called XML-DEV[67]. It is important to mention that SAX is neither an official standard nor a W3C Recommendation like DOM but it has become a de facto standard since so many XML products support it.

SAX chooses an alternative way to access the information in an XML document, not as a tree of nodes, but as a sequence of events. SAX chooses not to create a generic programming object model on top of an XML document (like DOM does). This makes SAX faster, but at the same time requires extra programming like creation of an own custom object model and a class that listen to SAX events and properly instantiating an object of that model.

DOM does not need all this extra programming, because it already creates an object model at beginning of the XML parsing (which represents the information as a tree of nodes).

On the other hand, SAX does not depend on the parser to do much of the work. All that SAX requires is that the parser should read the XML document, and fire a bunch of events depending on what tags it comes across in a given XML document. SAX will fire an event for every open and every close tag. The SAX document handler will have to interpret these events. Additionally it must interpret the sequence in which these events are fired. SAX allows quickly creating a handler class, which can create instances of object models based on the data in an XML documents. An example of this approach is shown on figure 2.9. A SAX document handler reads an XML document that contains the address book and as a result creates the *AddressBook* class. The address book XML document contains <person> elements (nodes), which contain <name> and <email> sub-elements (sub-nodes). The *AddressBook* object model contains the following classes:[68]

- *AddressBook* class, capsulating the *Person* objects.
- *Person* class, capsulating the name and email *String* objects.

[66] See Bates (2003), p.290.
[67] See XML-DEV. [last visited 04.05.2007]
[68] See Idris. [last visited 04.05.2007]

Following this schema, the SAX document handler is responsible for turning <person> into *Person* objects, <name> and <email> elements into *String* objects and then storing them in an *AddressBook* object. So the SAX document handler does element to object mapping.

2.3.4 Parsing XML Data with MSXML

Starting with Internet Explorer 4.01, Microsoft® have issued their own XML parser – MSXML (a COM component).[69] The MSXML library provides a set of objects that abstract the complexity of XML.[70] Most of that complexity is beyond the scope of this diploma thesis. There are only four important objects needed for the building of the plug-in application:[71]

- **XMLDocument** object represents an entire XML document.
- **IXMLDOMNode** object represents a single entity in the tree.
- **IXMLDOMNodeList** object represents a collection of child nodes for a particular entity.
- **IXMLDOMNamedNodeMap** object represents a collection of attributes for an entity.

The naming convention of those objects follows the rule - DOM stands for Document Object Model and the "I" stands for interface (COM interface).

Adding a Microsoft® XML (2.0 - 4.0) object reference to the VB project is by using the References dialog (in a stand alone Visual Basic IDE or in a Visual Basic Editor in any Microsoft® Office Application), see figure 2.6. The References Dialog can be accessed by selecting the Project Menu - > References.[72] (See figure 2.7). The XML DOM objects can than be created using the CreateObject() syntax like[73] in Listing 2.7.

```
Dim objXmlDoc
Set objXmlDoc = CreateObject(“Microsoft.XMLDOM”)
```

Listing 2.7 Creating an DOM Object

[69] See Gunderloy. [Last visited 04.05.2007]
[70] See Gunderloy. [Last visited 04.05.2007]
[71] See Gunderloy. [Last visited 04.05.2007]
[72] See Geese (2001), p.68.
[73] See Wilson (2000), p.79.

Lots of people nowadays use XML to save their data. Applications can also use XML to save configurations and almost any other type of information, from chemical formulas till data archives.[74] In the case of the plug-in application the information saved in the automatic generated .xml files can be used from other applications like search or indexing machines to enhance the process of managing e-Learning materials.

In the next chapter a brief introduction to the eduWEAVER managing platform and the proprietary ADONIS® Definition Language (ADL) and AdoScript language will be presented.

[74] See Ray (2001), p.279.

2.4 eduWEAVER Modelling Library

2.4.1 Introduction

The general offers of e-learning management solutions exceed to some extend the demand for those products. This fact does not help solving the problem most of the teaching staff faces nowadays. Using these tools to managed courses (e-learning or classroom attended) is not yet a trivial task. Most of the pre-issuing process like: digitising and multimedializing of the teaching materials require technical knowledge, which most of the teaching staff not yet possess. This situation helps to explain the fact that most of the instructor staff does neglect the possibilities an e-learning platform so-called "Learning Management Systems (LMS)"[75] brings to the knowledge transfer process.

The reusing of contents is extremely important in this field, where budgets are getting tighter every day. The development of reusable courseware is being supported by several process models, but user tools that implement these models are still not brightly accessible.

Within the eduBITE project (Educating Business and Information Technologies)[76] a modelling tool called eduWEAVER was developed to overcome the problems mentioned before.

eduWEAVER tool is based on the meta-modelling platform ADVISOR® and was developed as a specific method of ADVISOR®.[77] ADVISOR® itself is an open meta-modelling platform, developed by the BOC Information Technologies Consulting GmbH[78] in connection with the ESPRIT project of the European Union between 1998 and 2000.[79] ADVISOR® additionally offers supportive features for the training of employees in an e-learning environment. eduWEAVER serves as a learning object repository (learning object pool), which manages metadata-indexed learning objects (LOs), which are reusable content chunks with a high degree of cohesion.[80] The thought behind this functionality of eduWEAVER is to gather reusable multimedia content and empower the process of engineering and re-engineering of new and old on/offline instructional courses. These LOs can then be retrieved using metadata and reused by other members of the teaching staff. As being ADVISOR®'s,

[75] Bajnai (2003) [Last visited 04.05.2007]
[76] eduBITE [Last visited 16.08.2007]
[77] eduWEAVER [Last visited 04.05.2007]
[78] BOC Information Technologies Consulting AG [Last visited 16.08.2007]
[79] See Akogrimo Consortium (2005).
[80] Steinberger (2002)

the eduWEAVER modelling library inherits all of the possibilities offered by this modern instruction process modelling tool. Teachers can use the functionality of a kind of CASE tool that supports courseware design and generation. This process is being supported through a graphical user interface (GUI) It helps in a graphical way combine LOs to set of lessons, modules and courses and export the results via IMS[81] packages to an LMS of choice. This process can end with the import of those IMS packages in a Learning Management System of choice.

2.4.2 Elements of eduWEAVER

LOs are defined as "…self-contained reusable multimedia materials describing a homogenous chunk of content to be taught…"[82]. The base of eduWEAVER LO is the concept of Reusable Learning Objects[83]. Normally a LO should not exceed 10 – 30 minutes instructional time and be thought after special didactical and technical guidelines. A LO has three different elements: *Content Item*, *Practice Item* and *Assessment Item*. *Content Item* part represents the content of the course, *Practice Item* is the basis for exercises and the *Assessment Item* is to examine the learning process (see figure 2.10).

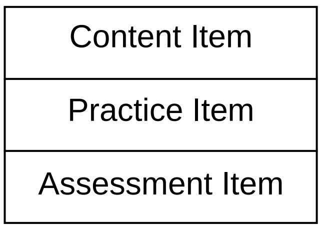


Figure 2.10 General concept of a learning object (LO)[84]

Different multimedia authoring tools like Microsoft® Office, Dreamweaver, Macromedia Flash, Authorware, Macromedia Director, SAP iTutor, and many others can be used in the process of LO creating. Each ready LO (e.g. LO 'What is an ERP?') is finally stored in the repository whereas special metadata have to be specified by the author (see also figure 2. 18).

[81] See IMS. [Last visited 04.05.2007]
[82] Bajnai (2003) [Last visited 04.05.2007]
[83] See Cicso Systems Inc. (2000).
[84] See Cicso Systems Inc. (2000).

In the role of a course designer, the teaching staff can access the eduWEAVER LO and graphically model a group of LOs to build the demanded course. In this process of course design the metadata of the LOs is used. This is how the LOs but also larger models (lessons, modules) stay highly reusable using eduWEAVER.

A ready designed course can be exported via an IMS-package and can be offered online to students via a LMS.

A member of the teaching staff can use eduWEAVER in three different ways, best situating her/ his role:

- As a content author - to compose LOs and publish them in the LO pool.
- As a modeller - to take LOs from the pool and use them in way to compose and model new lessons, modules and courses with the help of the eduWEAVER GUI.

As a coach - to export the LOs as IMS content packages in order to provide the instructional courses through any IMS-compliant LMS of choice.

2.4.3 The Metamodels of eduWEAVER

2.4.3.1 The Educational Metamodel of eduWEAVER

eduWEAVER modelling tool uses on one hand a hierarchical and on the other hand a process oriented way of course design. Those two ways reflect in the courseware modelling use case (see figure 2.11) with three basic user roles and four interconnected model types (see figure 2.13 - 2.16).

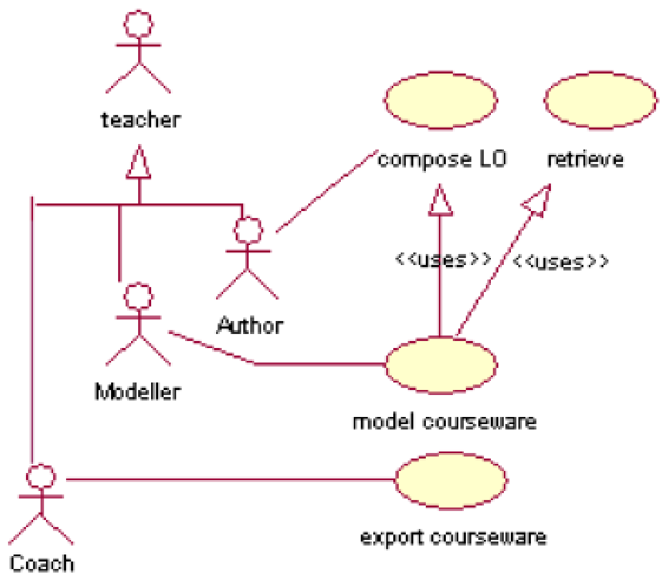


Figure 2.11 The eduWEAVER use cases[85]

A LO content author can use eduWEAVER to compose multimedia LOs and referenced them in the LO pool. In addition she/he can meta-indexing them. As course modeller she/he can take LOs from the Learning Object Pool and use them in order to model new lessons, modules and courses. In a role of a coach, the teacher can make use of IMS content packages export functionality of the eduWEAVER and import the ready courses in any IMS-compliant LMS.

The educational meta model of eduWEAVER can be defined in four modelling levels. On modelling level-1, *courses* can be defined as separated and in that sense as independent units. This is the most abstract level, where every single course has no connection to other learning units(see figure 2.12).

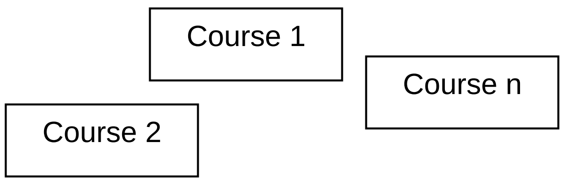


Figure 2.12 Courses on model level 1 of eduWEAVER [86]

[85] Figure taken from Bajnai (2003).[Last visited 03.05.2007]
[86] Figure taken from Bajnai (2003).[Last visited 03.05.2007]

On level-2, a course can be divided into *modules*, representing thematically united learning units. They can follow a consequential order, building a learning transfer process with the possibility to choose several different paths (see figure 2.13).

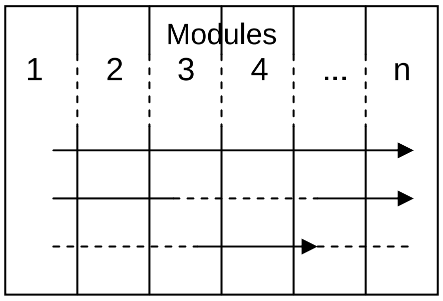


Figure 2.13 Possible paths on modelling level 2 of eduWEAVER [87]

On level-3, a *module* is divided into *lessons.* Typical duration of a unit on that level is about 45-90 minutes of instruction time. Here the course designer can use a hierarchical or sequential path in the knowledge transfer process. The target user previous knowledge level determines the actual process paths like: alternative paths, loops or ones (see figure 2.14).

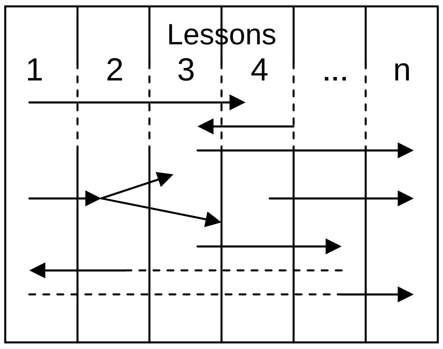


Figure 2.14 Possible paths on modelling level 3[88]

The smallest possible units in the educational metamodel of eduWEAVER resides on the modelling level-4, the level of *learning object use (LOV – abbreviation from German "**L**ern **O**bjekt **V**erwendung").* On that level the LOs can be arrange in a chain or using parallel paths and in that way made part (*used)* of the teaching process. The course designer defines references to the (*unused*) learning objects, which reside in the Learning Object Pool

[87] Figure taken from Bajnai (2003).[Last visited 03.05.2007]
[88] Figure taken from Bajnai (2003).[Last visited 03.05.2007]

(repository). An average lesson should approximately gather a set of about 7 *learning objects* beginning with an overview part and ending with a summary (see figure 2.15).

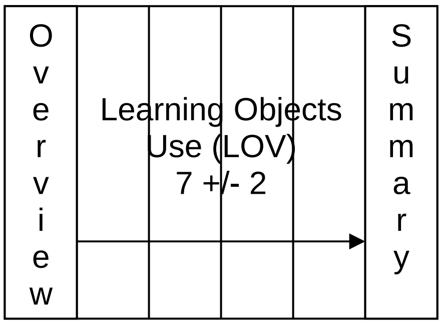


Figure 2.15 LOV on modelling level 4 of eduWEAVER [89]

Course modelling with eduWEAVER can be fulfilled in two vertical directions. The *top-down* approach was already presented above. Course designers can also start a *bottom-up* approach by searching for possible learning objects in the *Learning Object Pool* or creating new ones using an external content creation application. They can then referenced them in a LOV model respectively importing the new ones in a Learning Object Pool and then setting a reference to them in the target LOV model. Next steps are grouping them into a lesson, defining several lessons as a module and finally building a whole course. Figure 2.16 represents the both approaches.

[89] See Cicso Systems Inc. (2000).

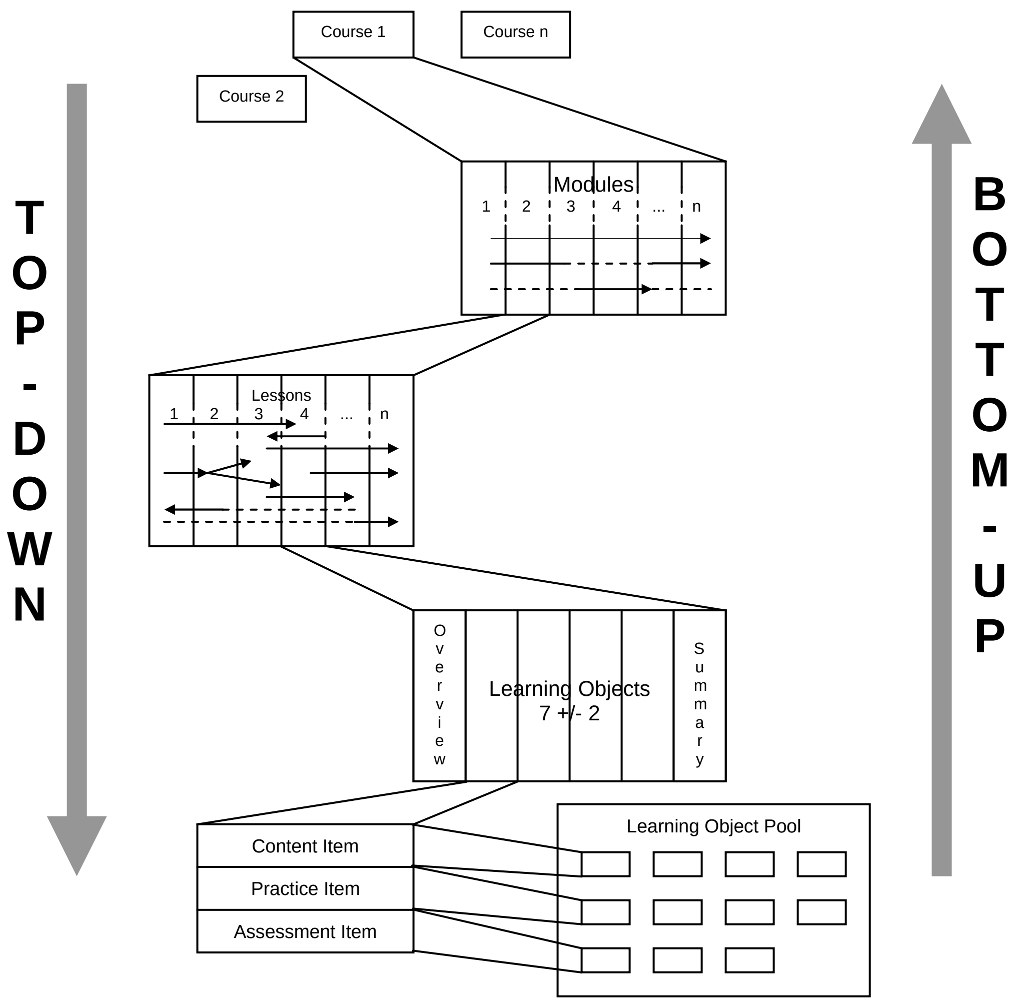


Figure 2.16 Top-down and bottom-up approaches in eduWEAVER [90]

2.4.3.2 The Programming Metamodel of eduWEAVER

The programming metamodel of eduWEAVER modelling library is based on the metamodelling platform ADONIS®.[91] ADONIS® is a generic process modelling platform that lets the usage of newly defined modelling methods. Figure 2.17 represents the meta model that was chosen by the developers of eduWEAVER. Some of the functionalities implemented in the ADONIS® like the search of LOs, analysis, definition, target group simulation, export function, etc. are fully inherit by the eduWEAVER.

[90] Figure taken from Bajnai (2003).[Last visited 03.05.2007]
[91] See BOC Information Technologies Ltd. [Last visited 04.05.2007]

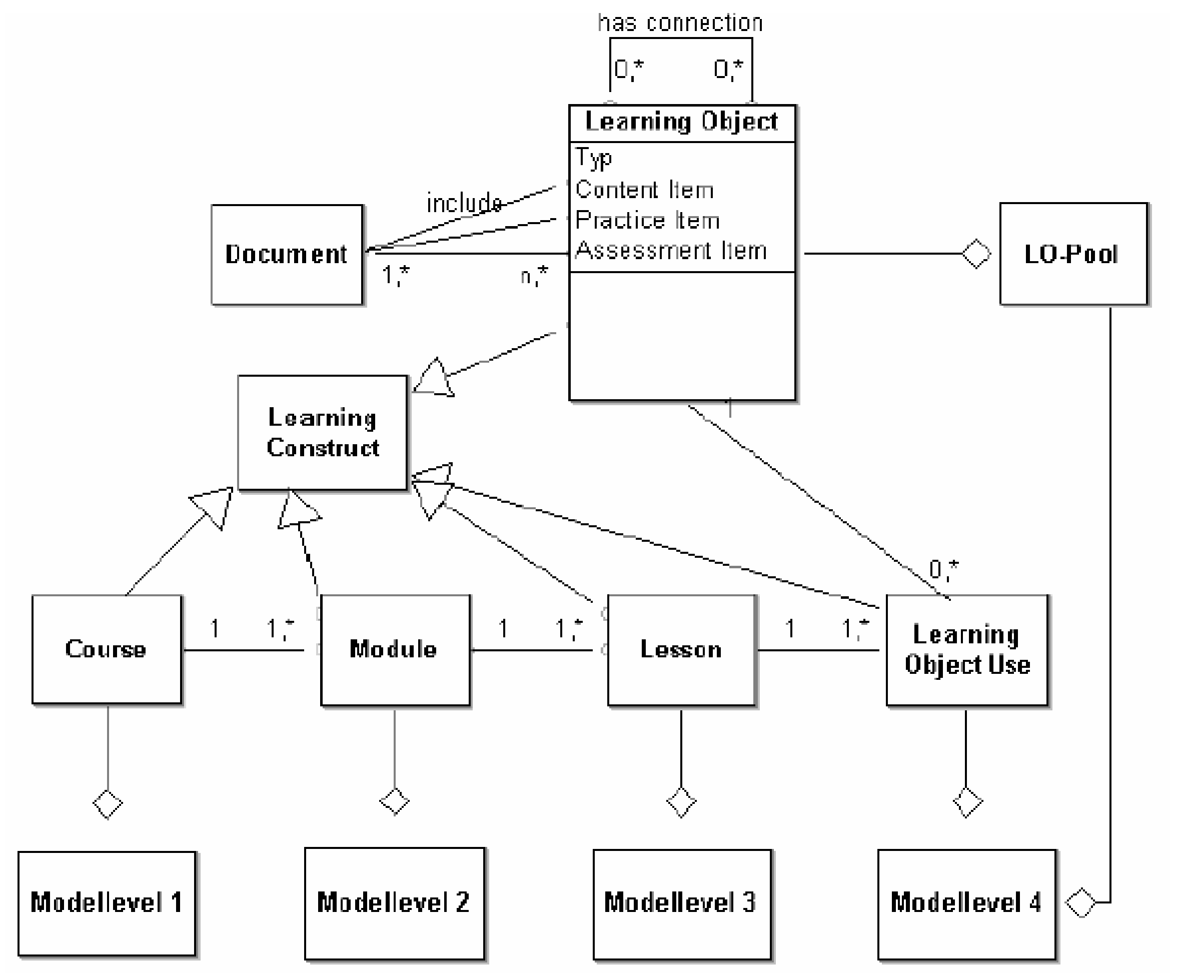
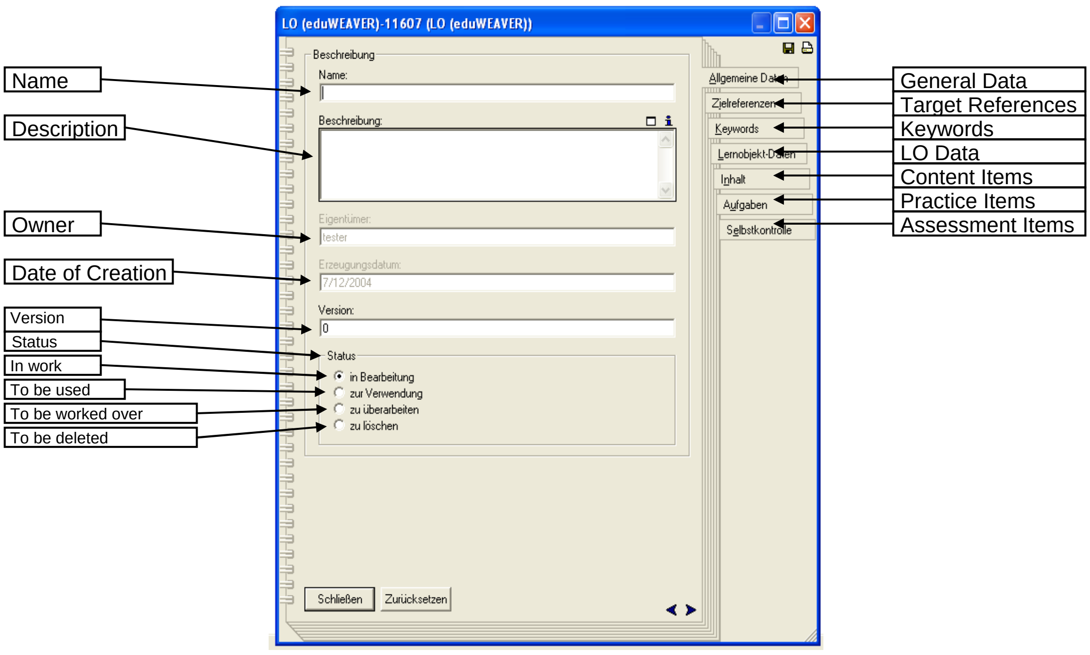


Figure 2.17 Simplified metamodel of eduWEAVER[92]

There are four model levels: *Course*, *Module*, *Lesson* or *Learning Object Use*. The model types are instantiated by a learning constructor and hierarchically linked to each other by internal references. On level-4 there exist a direct link between the *Learning Object Use* (LOs in a model) and the *Learning Object Pool* elements (LOs in a repository). The LOs in the pool are referenced to multimedia content and are additionally meta-indexed by their authors as shown in figure 2.18**.** On the Graphical User Interface of eduWEAVER all the different metadata are represented into different tabs covering selected parts of the IMS standard.[93]
The meta information to a particular Learning Object are divided into *General Data*, *Target References*, *Keywords* and *LO Data* give the user the possibility to input all the meta data that corresponds to the IMS standard.

[92] See Kühn (1999), pp.173-174.
[93] See IMS. [Last visited 04.05.2007]

In chapters Content Items, Practice Items and Assessment Items, representing the concept of a reusable learning object, the multimedia documents can be referenced. These documents can have any of the usual multimedia formats like HTML, PDF, PowerPoint®, Flash-Movie, and so on.


LO (eduWEAVER)-11607 (LO (eduWEAVER))

Beschreibung

Name:

Beschreibung:

Eigentümer: tester

Erzeugungsdatum: 7/12/2004

Version: 0

Status: [✓ in Bearbeitung] zur Verwendung zu überarbeiten zu löschen

Figure 2.18 Metadata indexing[94]

2.4.4 Course Modelling with eduWEAVER

The process of course modelling every time begins with providing of LOs to the *Learning Object Pool*. They can be referenced to self developed multimedia materials or to materials developed by experts in multimedia design.

The short example bellow shows the bottom-up approach of course modelling with eduWEAVER.

This exemplary lesson is represented by three LOs and the teacher can choose two different parallel paths. The theme of the lesson is about process modelling starting with an introductory. Then the teacher can choose to follow either with a theoretical part or a practical example (see figure 2.19).

[94] Figure taken from Bajnai (2003).[Last visited 03.05.2007]

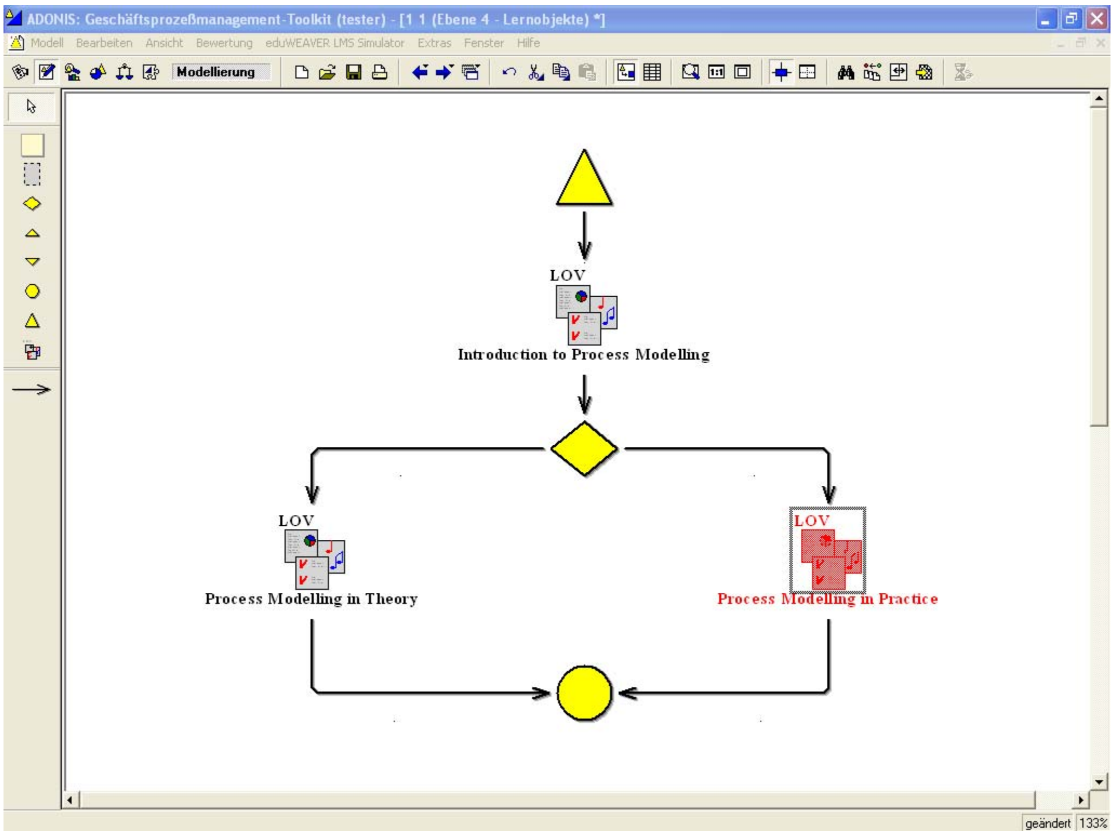

ADONIS: Geschäftsprozeßmanagement-Toolkit (tester) - [1 1 (Ebene 4 - Lernobjekte) *]
Modell Bearbeiten Ansicht Bewertung eduWEAVER LMS Simulator Extras Fenster Hilfe
Modellierung
LOV
Introduction to Process Modelling
LOV
Process Modelling in Theory
LOV
Process Modelling in Practice
geändert 133%

Figure 2.19 Example for modelling level 4[95]

This example shows the graphical approach of eduWEAVER in the process of course modelling.

2.4.5 ADL (ADONIS® Definition Language)

The developer of ADONIS introduced a markup like language named ADL (ADONIS® Definition Language) which purpose is to save the existing models for export in another ADONIS instalation. Its syntax is very close to this of XML. In listing 2.9 an example of the ADL syntax is shown. At the beginning, the TYPE<> element indicates the type of the model and in addition includes attributes for describing the model specific information like: author of the model, time of creation, settings for the visual representation, etc. The INSTANCE<> elements define concrete instances of the modelling elements specified in the metamodel with attributes specific to the ADONIS® modelling environment. With the RELATION<> element

[95] Figure taken from Bajnai (2003).[Last visited 03.05.2007]

instances of the relation classes are defined including the*'Is From Class'* and *'Is To Class'* relationships.[96]

```
//...
TYPE <Ebene 4 - Lernobjekte>
      ATTRIBUTE <Autor>
      VALUE "tester"

      ATTRIBUTE <Angelegt am>
      VALUE "07.12.2004, 19:03"

      ATTRIBUTE <Letzte Änderung am>
      VALUE "07.12.2004, 19:48"

      ATTRIBUTE <Letzter Bearbeiter>
      VALUE "tester"
//...
INSTANCE <Introduction to Process Modelling> : <LOV (eduWEAVER)>

      ATTRIBUTE <Position>
      VALUE "NODE x:9.50cm y:4.00cm w:1.23cm h:1.45cm index:1"
//...
RELATION <Nachfolger>
      FROM <Introduction to Process Modelling> : <LOV (eduWEAVER)>
      TO <Entscheidung (eduWEAVER)-11665> : <Entscheidung (eduWEAVER)>
 //...
ATTRIBUTE <Positions>
      VALUE "EDGE 0 index:5"
//...
```

Listing 2.9 Excerpt of the ADL code

After this introduction of the relevant technical aspects of the plug-in application a programming process has to be started. In the next chapter the application's work flow is to be determined.

[96] See Fill. [Last visited 25.04.2006]

2.4.6 AdoScript

For the purpose of programming add-on or plug-in for Adonis and all its derivates a script language AdoScript is introduced. AdoScript offers commands for interacting with the program functionality (loading a model, selecting an object, ADL – Export), as well as manipulation of models and definition of extra menus or menu items. These menu items can execute an external programs or functions (DLLs) which can interact through open APIs with the Adonis models. Exactly those possibilities will be used by the eduWEAVER plug-in, which will implement the communication between the external plug-in execution program and the eduWEAVER.

3 Development of the eduWEAVER Plug-in

3.1 Application Structure

The plug-in application will enable the teaching stuff to transform externally made content to a form that can be managed by eduWEAVER. The plug-in will have two versions for the “backloading” process.

The first version will transform every slide of the presentation into a LO of a Learning Object Pool Model and reference them to a LOV of a Learning Object Use Model (named “PPT2ADL_one2one.exe”). The program will then ask the user if he/she wants to import the new generated models into eduWEAVER.

The second version will transform the whole presentation into one LO and its corresponding LOV (named “PPT2ADL_many2one.exe”). It will again ask the user if he/she wants to import that LOV into an existing LOV in one step.

The plug-in can have a possible application structure as follow (see figures 3.1-3.2).

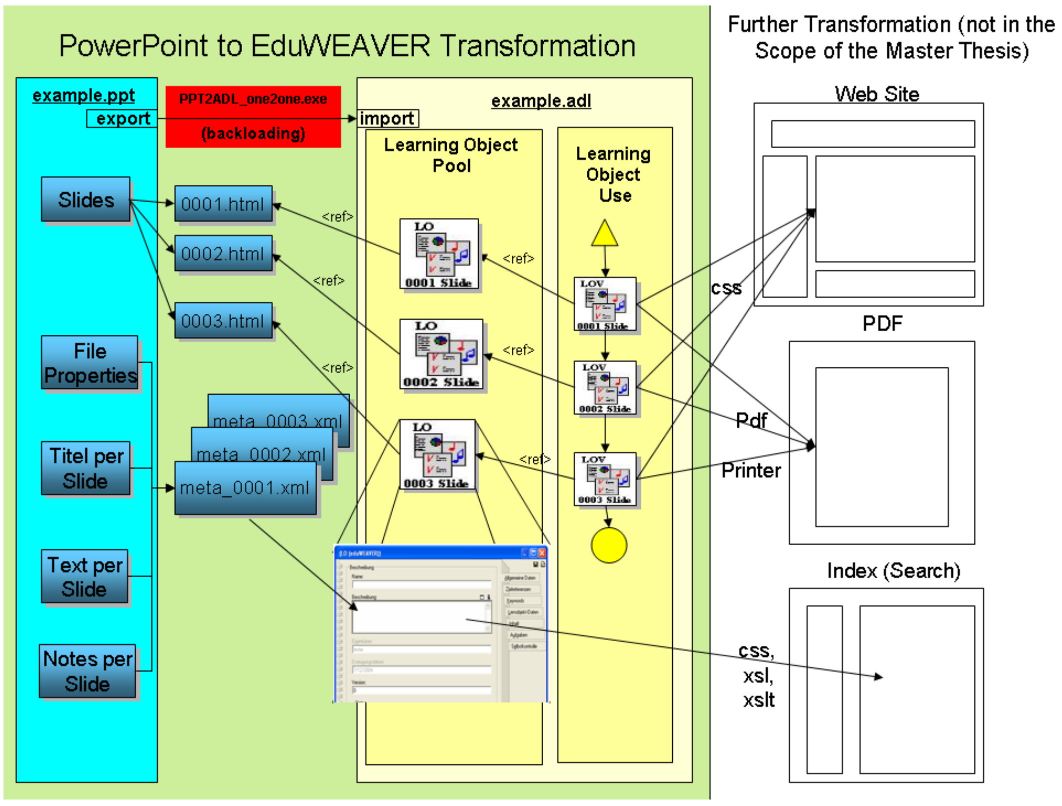


Figure 3.1 The application structure for Scenario 1 (PPT2ADL_one2one.exe)

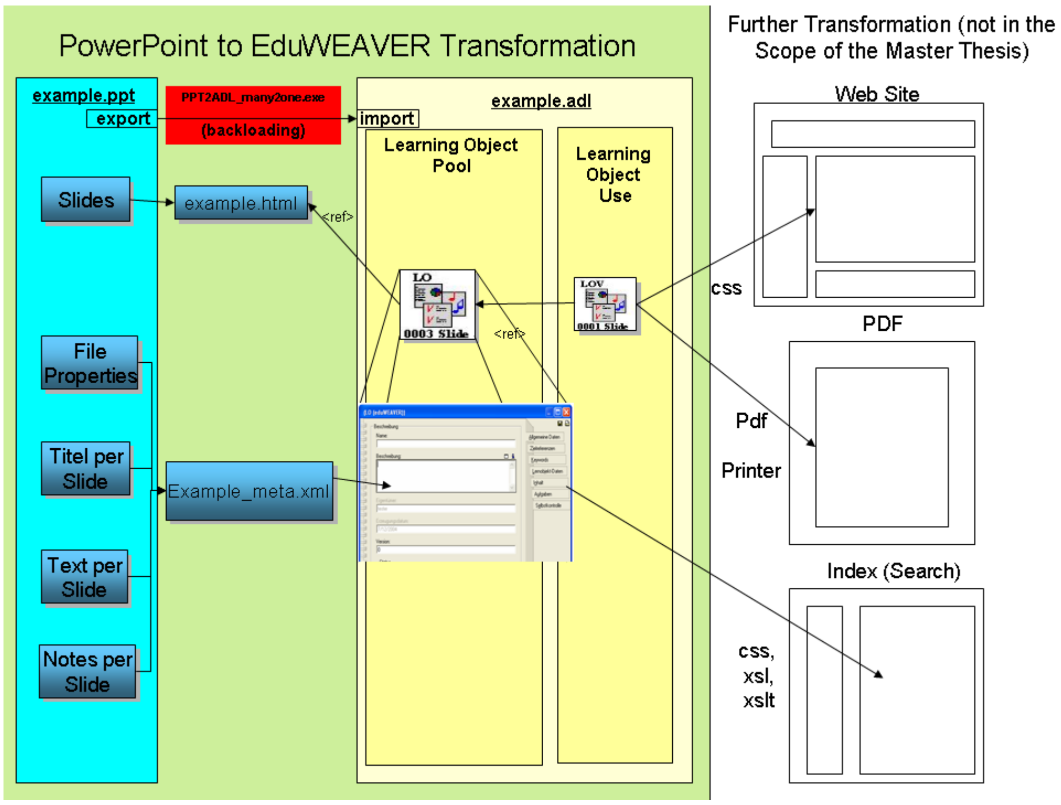


Figure 3.2 The application structure Scenario 2 (PPT2ADL_many2one.exe)

It will operate at the Office application model level, instantiating all relevant PowerPoint® document objects like slides, speaker's notes, slides' titles, texts, file properties. Through a "backload" process all meta data and content in text form for a given slide will be exported to a meta_X.xml (X being a sequential number) file and at the same time the text content will be placed in the description part of the Learning Objects and saved as name_of_the_presentation.adl file. Further more the application will use the PowerPoint® export functionality to save the PowerPoint® presentation in a HTML file format. Each slide (respectively all slides) will be saved as name_of_the_presentation_X.htm file and put into folder named after the presentation's title. The plug-in will also put a reference of each HTML file into an automatically created eduWEAVER's Learning Object. The user will then import the generated .adl file for further transformation into eduWEAVER. The generated .adl file will consist of a Learning Object Use and a Learning Object Pool meta models. The Learning Object Use model (Scenario I) will follow the presentation slides' sequence.

3.2 Application Work Flow

The application workflow of the plug-in will be as follow:

- It will start within the eduWEAVER.
- It will have its own GUI (VB6 Form).
- It will import .ppt files, converting them in background to .html, .xml, naming them as each slide's title if one exists and if not with an ongoing number.
- At the end the user will be able to save .adl file (containing the Learning Object Usage / Learning Object Pool to a directory of choice.
- The user will be then able to import the generated .adl file in eduWEAVER and use the .xml and .html for further transformation.

Figure 3.3 shows a possible program flow chart for the plug-in application. A program flow chart is normally used to describe the flow of data through a particular computer program, showing the exact sequence of operations performed by that program in order to process the data. Different graphic symbols are used to represent data input and output, decisions, branches, and subroutines.

The following table 3.1 explains the meaning of the symbols used in the program flow chart in figure 3.3.

Symbol	**Meaning**
▭	Processing activity
▱	Input / Output operation
◇	Decision to be made
⬭	Start / End
→	Flow Line

Table 3.1 Flow chart symbols explanation

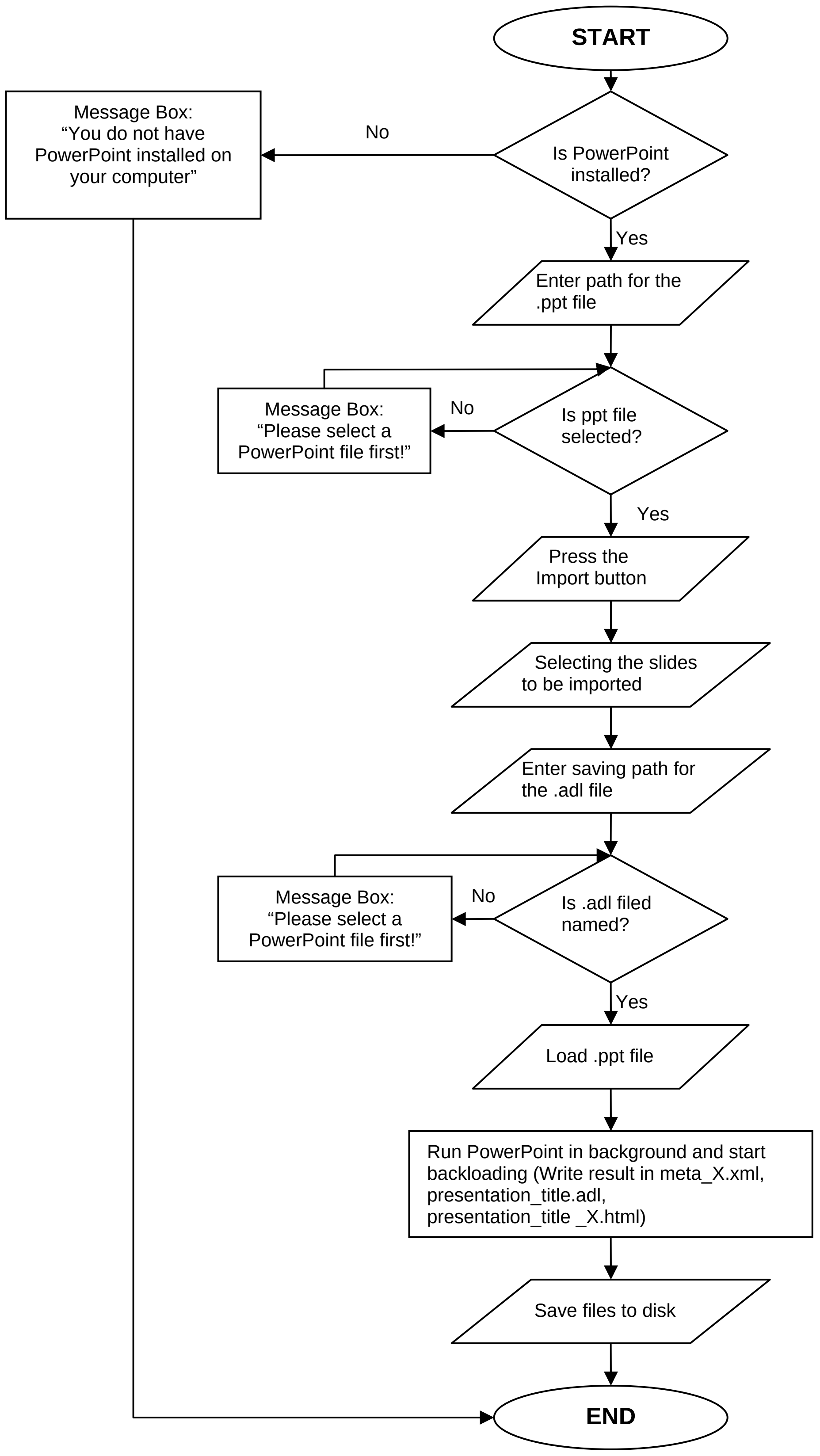

START
Is PowerPoint installed?
No
Message Box: “You do not have PowerPoint installed on your computer”
Yes
Enter path for the .ppt file
Is ppt file selected?
No
Message Box: “Please select a PowerPoint file first!”
Yes
Press the Import button
Selecting the slides to be imported
Enter saving path for the .adl file
Is .adl filed named?
No
Message Box: “Please select a PowerPoint file first!”
Yes
Load .ppt file
Run PowerPoint in background and start backloading (Write result in meta_X.xml, presentation_title.adl, presentation_title _X.html)
Save files to disk
END

Figure 3.3 Program flow chart

The plug-in will use the PowerPoint® installed on the target desktop through the automation that the VBA deliver. So it is important to have PowerPoint® installed there. Therefore the program checks for that installation before running the “backload” procedure. If no installation of PowerPoint® is presented the program will terminate after the “Start” button being clicked, giving a message box alert that PowerPoint® is not yet installed.

The user has two input fields for selecting the PowerPoint .ppt file and for the directory of the output files. The output button brings a “*Save as*” file window with already named (named after the PowerPoint® presentation name) .adl file. As discussed in chapter 2.5.2 Elements of eduWEAVER, the LO can be of type Content, Practice or Assessment Item. The user can choose one of these types by clicking on the radio button of choice. The Content Item is the default value. The user can also give a different name to the LO/LOV in the TextBox below the type check boxes.

After that the user can start the “backloading” process by clicking the “Start” button. During the transformation the user can stop the process by clicking on the “Stop” button and quit the program by clicking the “Quit” button. The result will be written in different files and these files will be saved at a position given by the user in the “S*ave as”* file window. That will be the end of the plug-in task in the scenario 1 (1:1) followed by the ADL Import window of the eduWEAVER. The user can import the generated .adl file to work within eduWEAVER, and the html and xml files for further transformation or presentation. When imported in eduWEAVER the user will find two models – *Learning Object Pool* and *Level 4- Learning Objects*. In chapter *General Data >> Description* the user will find the textual content of the presentation slide. He/she can then reorganise the text for a more accurate description of the LOV. The Learning Objects in the *Level 4* will have the as targets the LOs from the *Learning Object Pool*. The LOs in the pool will have as *Entering Point* in Chapter *“Content Item”* (alternatively *“Practice Item”* or *“Assessment Item”*) the HTML files generated by the plug-in.

The second scenario (N:1) will also automatically invoke the ADL import window of the eduWEAVER but after importing it will prompt for a suggestion in which LOV it should import the generated LOV. The user must choose one LOV that is already presented in the eduWEAVER. The plug-in will put the new LOV in the upper left corner of the graphical presentation of the hosting LOV. In the background the LO will also be imported in the general models area. The LO can be than manually imported into another LO Pool.

4 Installation and User Manual

4.1 Installation

The eduWEAVER plug-in will be distributed in form of two executables (*PPT2ADL_one2one.exe*/*PPT2ADL_many2one.exe*) and an AdoScript *import_into_model.asc* file *(see source code in Appendix)*. The three files should be copy/paste in the main directory of eduWEAVER. The eduWEAVER administrator should customise the layout of the modelling library by the means of external coupling to edit the import/export perspective menu of the eduWEAVER adding a new button (e.g. “backloading” see the Appendix for a sample code). The user can start the plug-in from there choosing the new button article (e.g. “Backloading”) with two sub-articles (e.g. “ADL Import n: 1”/”ADL Import 1:1”) which both will be linked to the plug-in executables.

4.2 User Manual

4.2.1 Scenario I

For starting the plug-in the user must go to the Import/Export perspective of eduWEAVER as shown in figure 4.1.

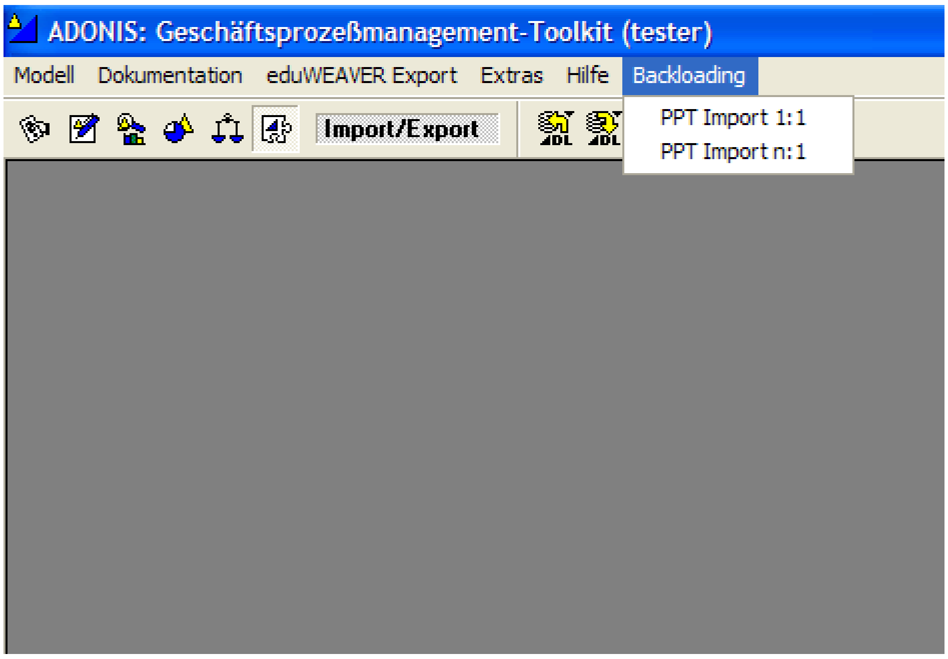


Figure 4.1 Import/Export perspective of eduWEAVER

On the menu area there is an item called “Backloading” with two sub-items “PPT Import n: 1” and “PPT Import 1:1”. The user can choose one of them corresponding to scenario she wants to use.

The steps bellow will demonstrate the Scenario I (PPT Import 1:1):

1. The plug-in application starts in an extra window as shown in figure 4.2.

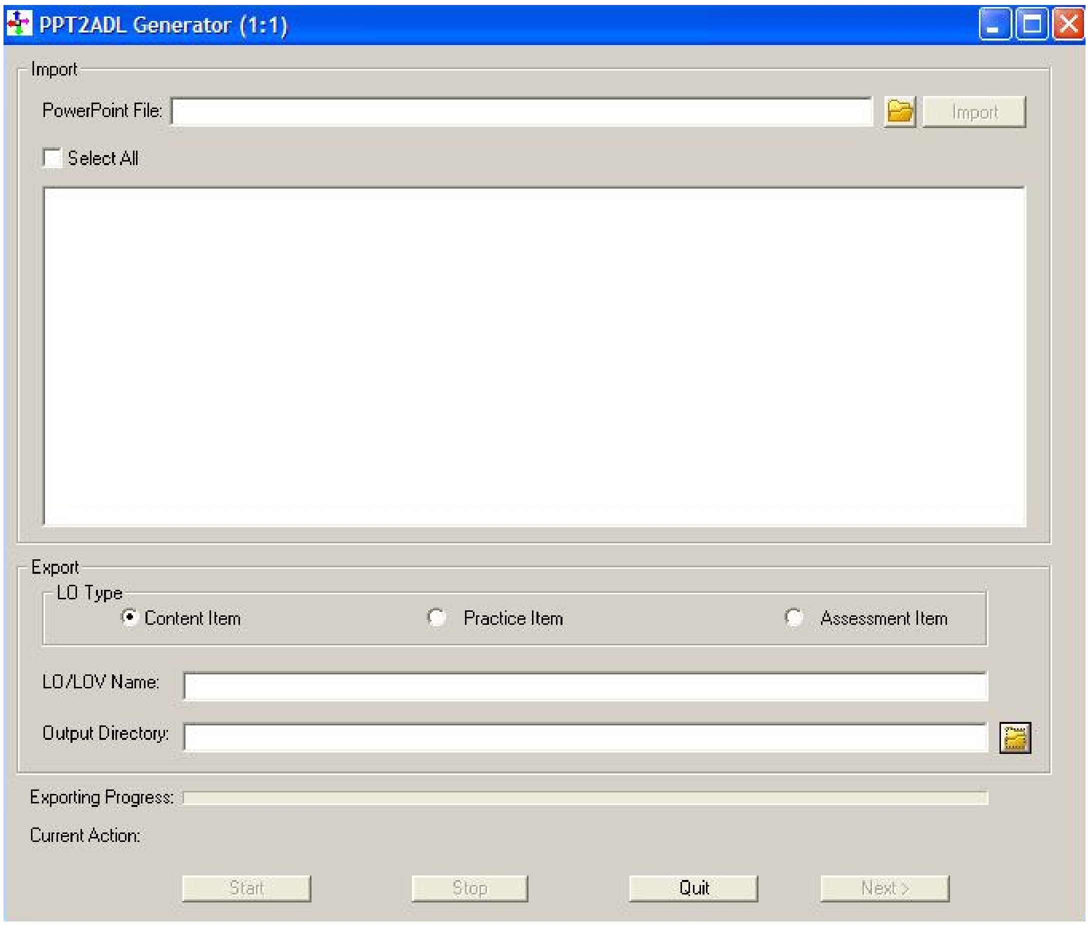


Figure 4.2 The GUI of the plug-in

2. After clicking the button in the "Import Field Area" (at the top of the application) the plug-in opens the Windows® explorer where the user inputs the path to the PowerPoint® file to be imported as shown in figure 4.3.

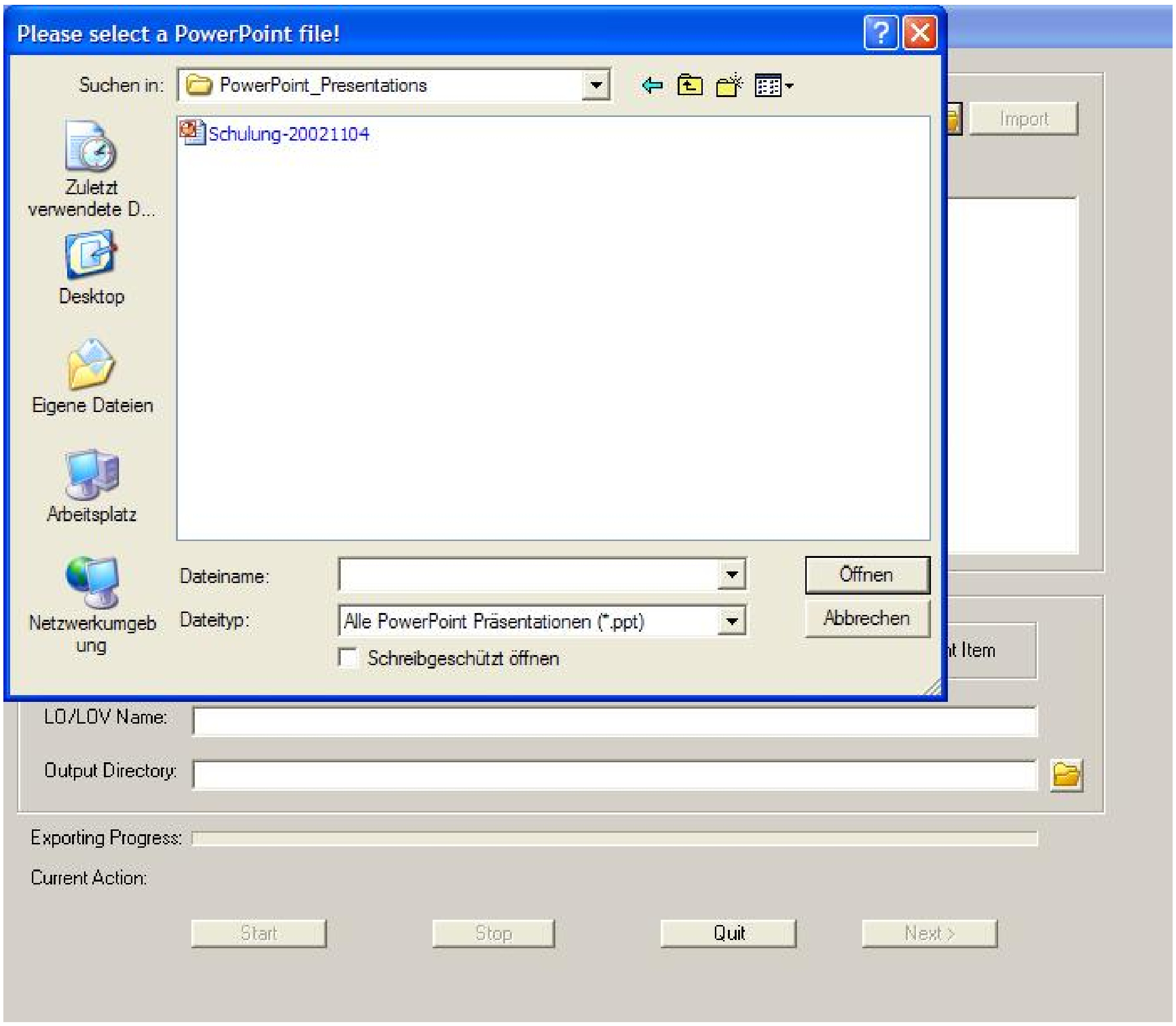
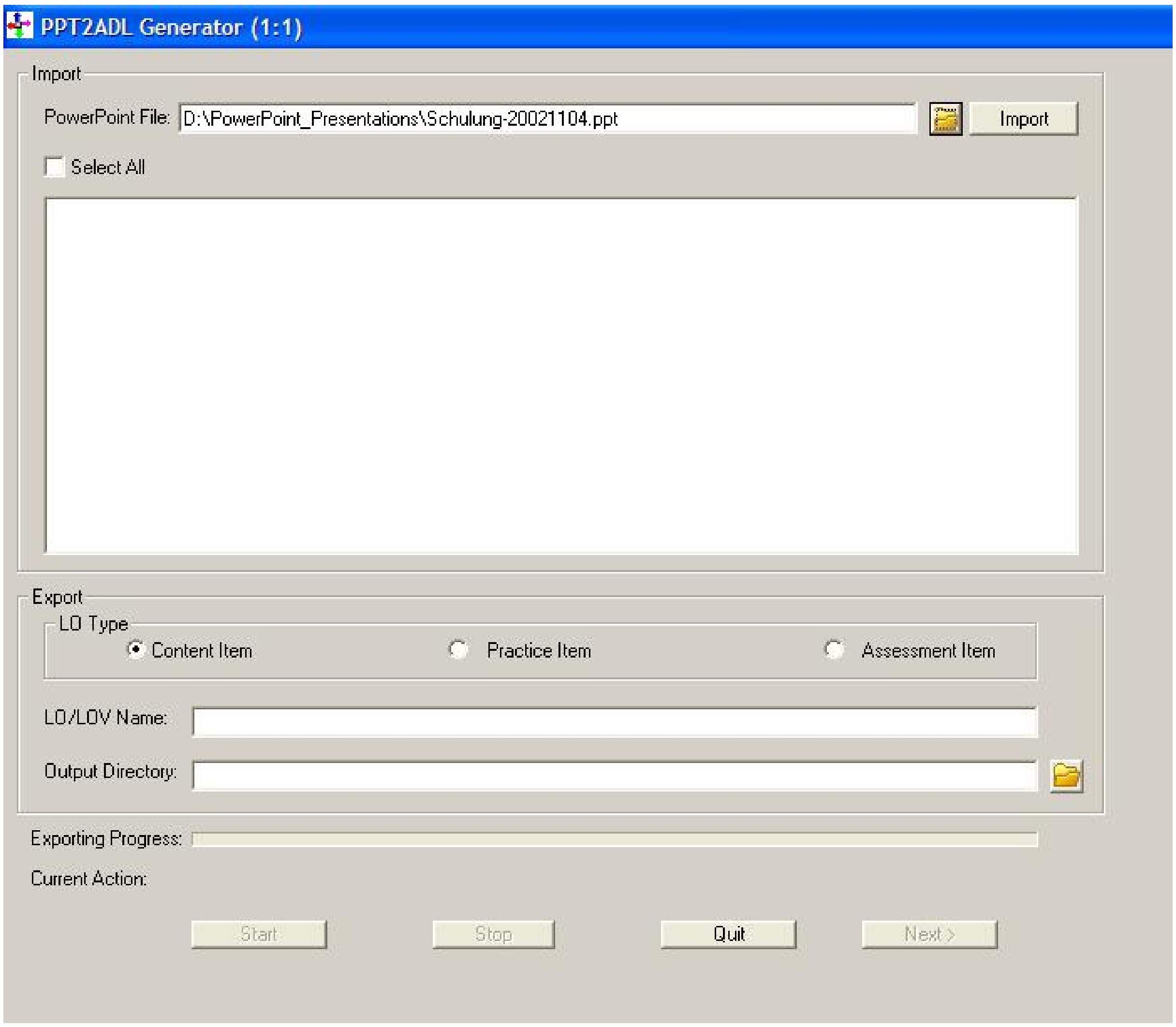


Figure 4.3 The window explorer shows the path to the PowerPoint® file

3. The Import button turns into active state and the user can start the import of the PowerPoint® file as shown in figure 4.4.



PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

LO Type: [✓ Content Item] Practice Item Assessment Item

LO/LOV Name:

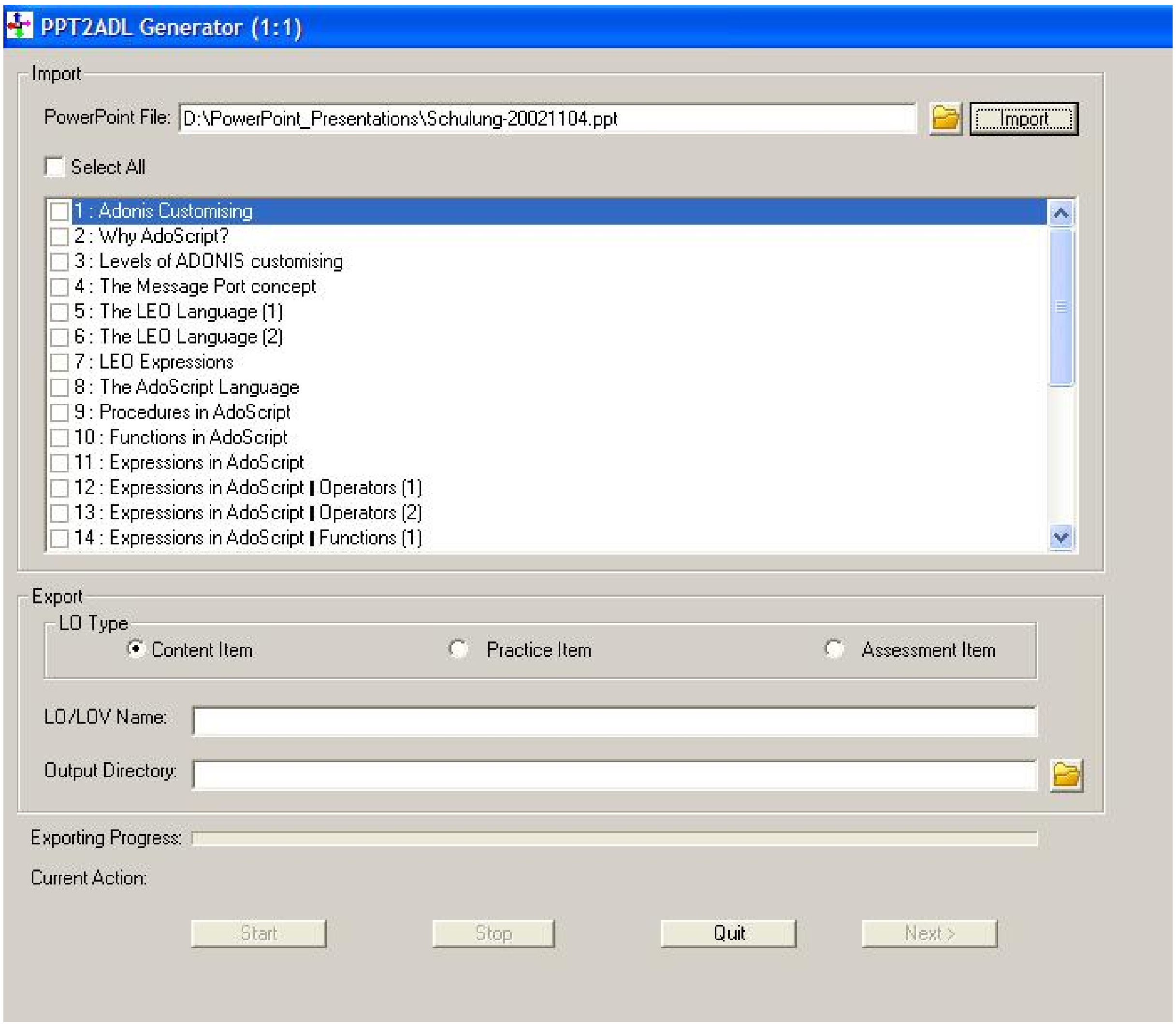
Output Directory:

Exporting Progress:

Current Action:

Figure 4.4 The" Import" button turn into active state

4. After clicking the Import button the plug-in starts to import the PowerPoint® application at loads all slides in the TextBox Area bellow as shown in the figure 4.5.


PPT2ADL Generator (1:1)

Import

PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

☐ Select All

- ☐ 1 : Adonis Customising
- ☐ 2 : Why AdoScript?
- ☐ 3 : Levels of ADONIS customising
- ☐ 4 : The Message Port concept
- ☐ 5 : The LEO Language (1)
- ☐ 6 : The LEO Language (2)
- ☐ 7 : LEO Expressions
- ☐ 8 : The AdoScript Language
- ☐ 9 : Procedures in AdoScript
- ☐ 10 : Functions in AdoScript
- ☐ 11 : Expressions in AdoScript
- ☐ 12 : Expressions in AdoScript | Operators (1)
- ☐ 13 : Expressions in AdoScript | Operators (2)
- ☐ 14 : Expressions in AdoScript | Functions (1)

Export

LO Type: [✓ Content Item] Practice Item Assessment Item

LO/LOV Name:

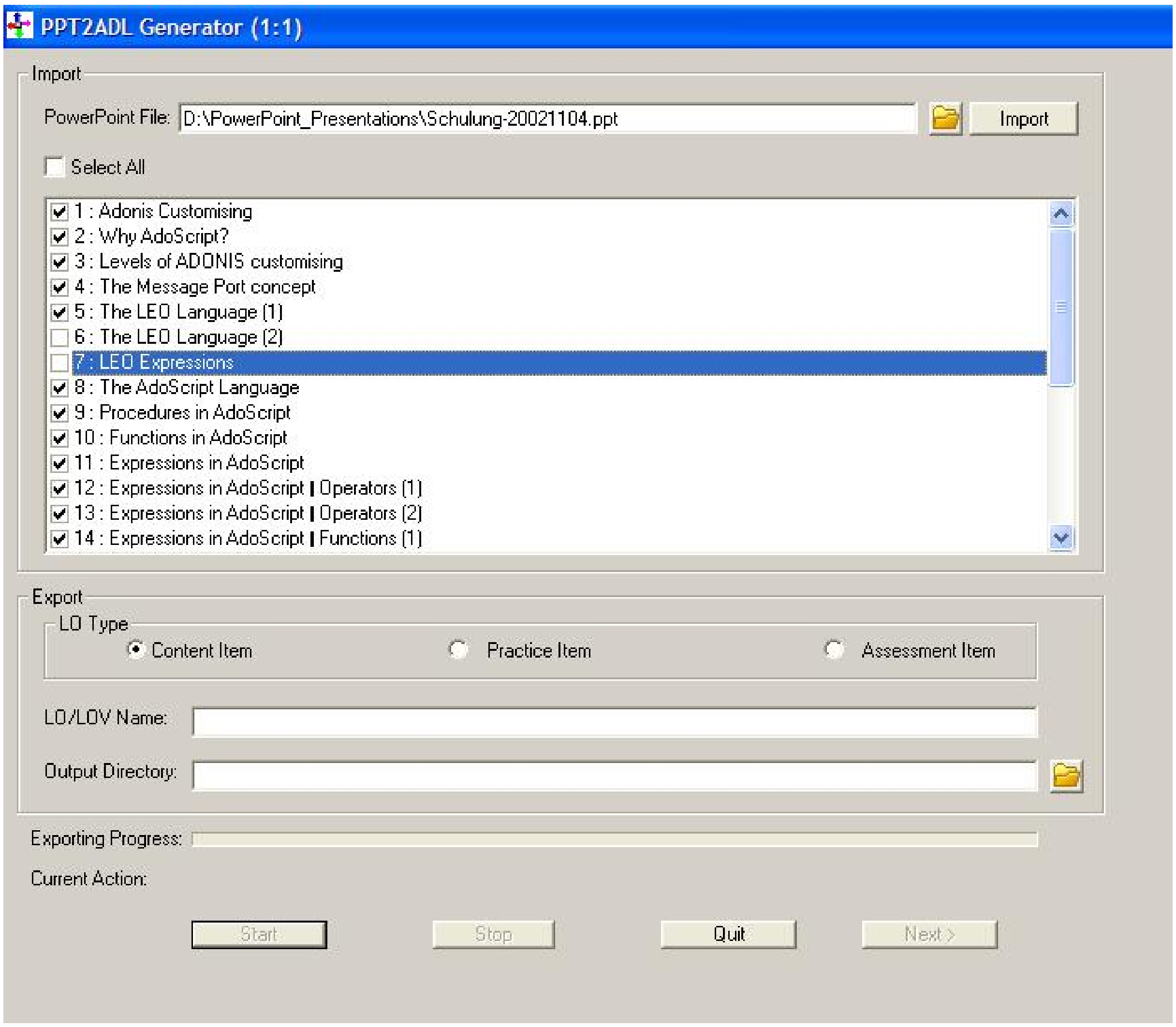
Output Directory:

Exporting Progress:

Current Action:

Figure 4.5 The imported PowerPoint® slides

5. The user can select all the slides or only those that he needs for the LOV/LO as shown in the figure 4.6. (by default all slides will be selected).



PPT2ADL Generator (1:1)

Import

PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

☐ Select All

☑ 1 : Adonis Customising
☑ 2 : Why AdoScript?
☑ 3 : Levels of ADONIS customising
☑ 4 : The Message Port concept
☑ 5 : The LEO Language (1)
☐ 6 : The LEO Language (2)
☐ 7 : LEO Expressions
☑ 8 : The AdoScript Language
☑ 9 : Procedures in AdoScript
☑ 10 : Functions in AdoScript
☑ 11 : Expressions in AdoScript
☑ 12 : Expressions in AdoScript | Operators (1)
☑ 13 : Expressions in AdoScript | Operators (2)
☑ 14 : Expressions in AdoScript | Functions (1)

Export

LO Type: [● Content Item] ○ Practice Item ○ Assessment Item

LO/LOV Name:

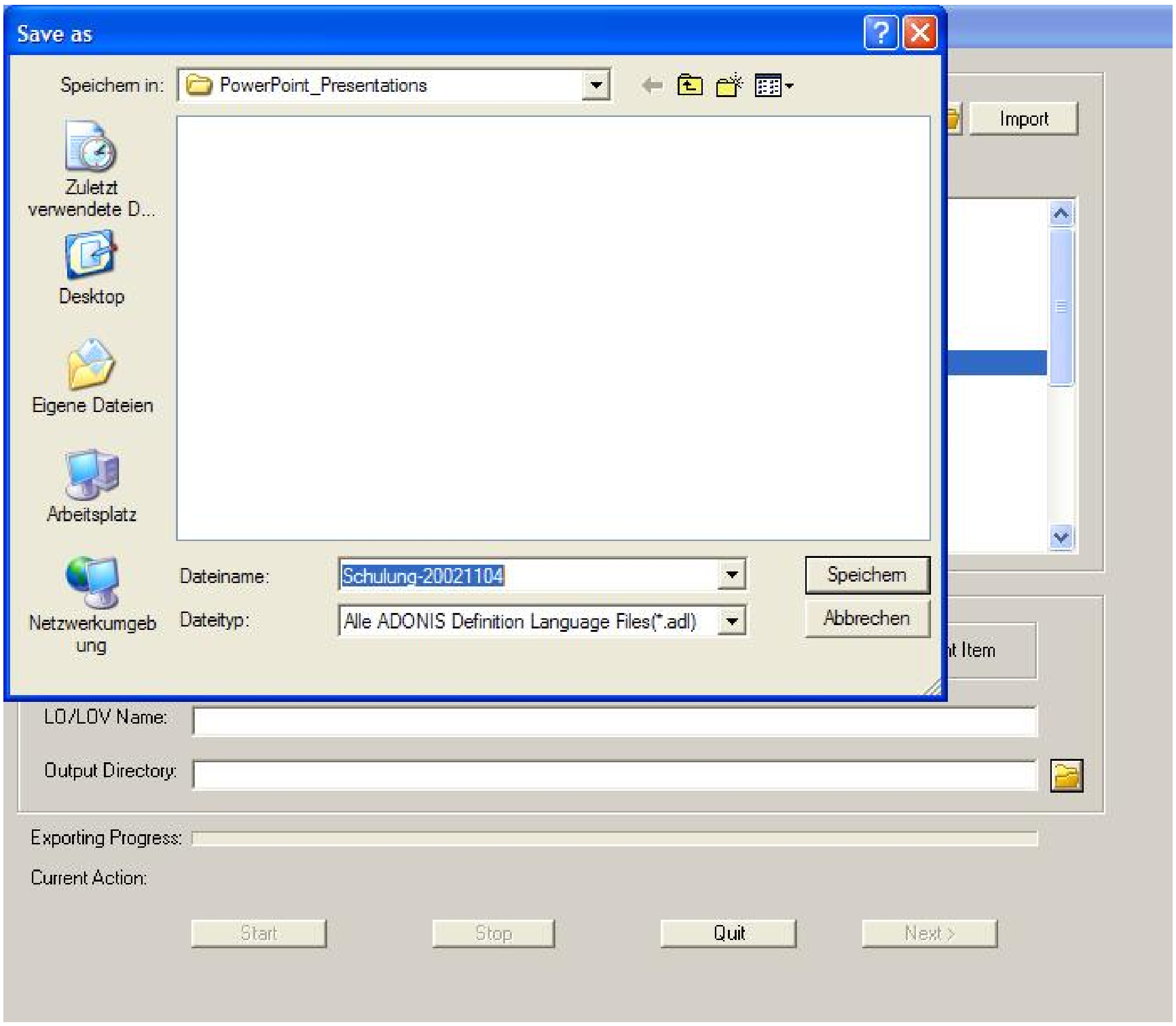
Output Directory:

Exporting Progress:

Current Action:

Figure 4.6 The imported PowerPoint® slides

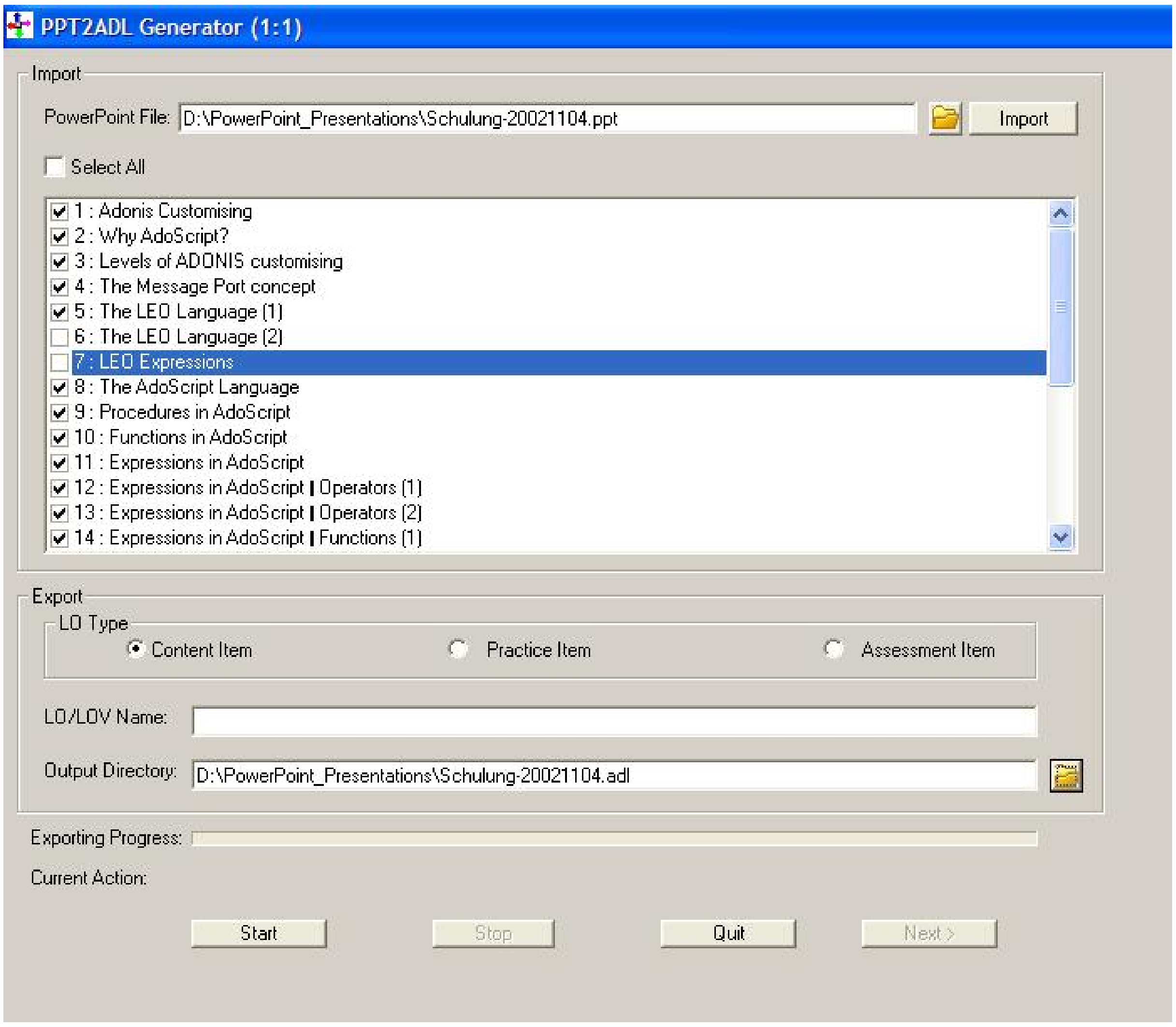
6. After that the user can choose between three types of LO namely “Content Item”, “Practice Item” and “Assessment Item”. (default value is “Content Item”).
7. The user can also give a special name for the LO/LOV (default value is the name of the presentation).
8. Now the user must give the path to the output directory (by clicking the button) where the .adl, .xml. and .html files will be saved as shown in the figure 4.7.



Dateiname: Schulung-20021104
Dateityp: Alle ADONIS Definition Language Files(*.adl)
LO/LOV Name:
Output Directory:
Exporting Progress:
Current Action:

Figure 4.7 The window explorer shows the path to the output directory

9. After that the “Start” button turns into active state as shown in the figure 4.8. The user can now start the transforming (“backloading”) process.


PPT2ADL Generator (1:1)

Import

PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

Select All

- [✓] 1 : Adonis Customising
- [✓] 2 : Why AdoScript?
- [✓] 3 : Levels of ADONIS customising
- [✓] 4 : The Message Port concept
- [✓] 5 : The LEO Language (1)
- [ ] 6 : The LEO Language (2)
- [ ] 7 : LEO Expressions
- [✓] 8 : The AdoScript Language
- [✓] 9 : Procedures in AdoScript
- [✓] 10 : Functions in AdoScript
- [✓] 11 : Expressions in AdoScript
- [✓] 12 : Expressions in AdoScript | Operators (1)
- [✓] 13 : Expressions in AdoScript | Operators (2)
- [✓] 14 : Expressions in AdoScript | Functions (1)

Export

LO Type: [● Content Item] Practice Item Assessment Item

LO/LOV Name:

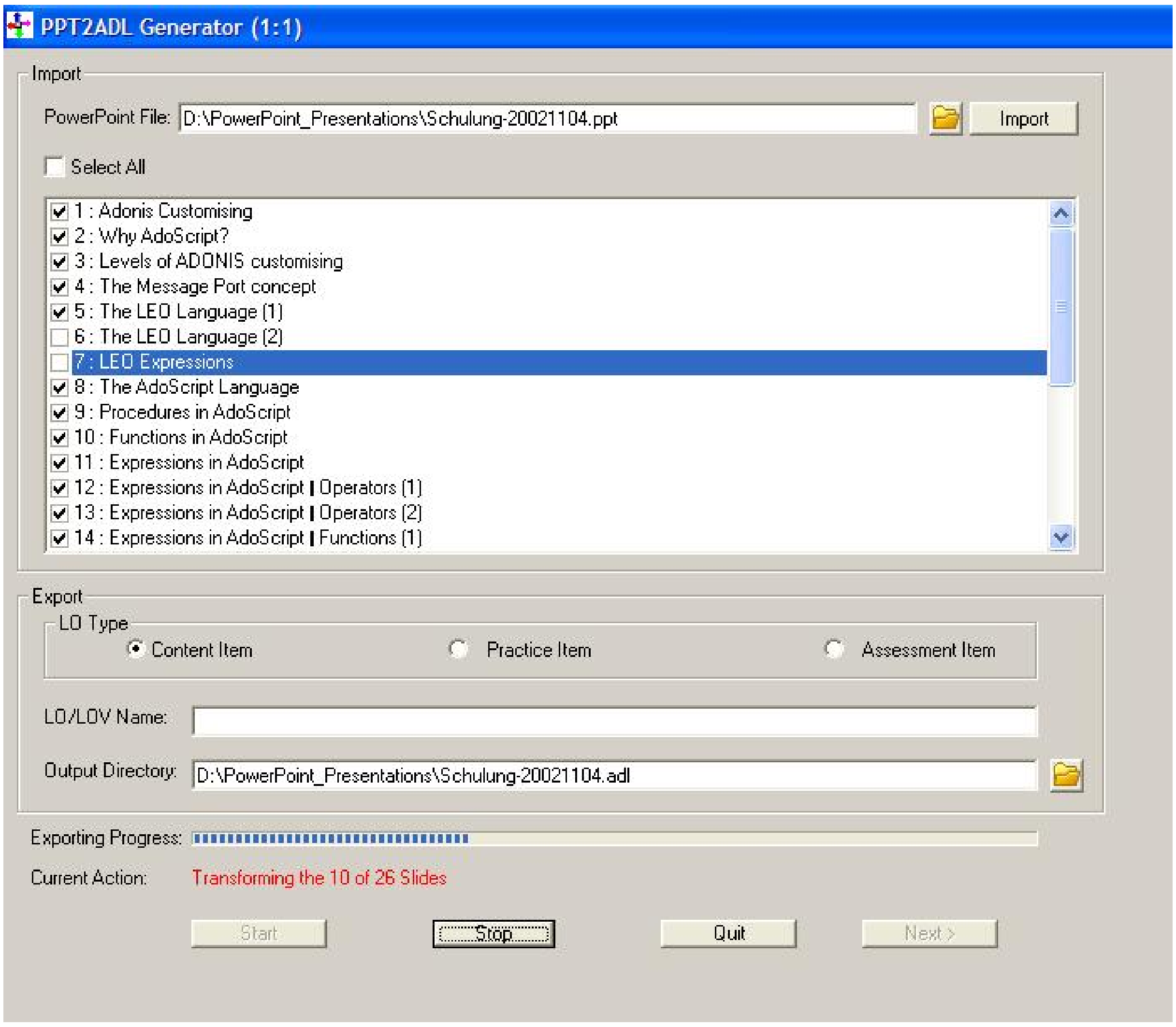
Output Directory: D:\PowerPoint_Presentations\Schulung-20021104.adl

Exporting Progress:

Current Action:

Figure 4.8 The "Start" button turns into active state

10. After starting the transforming process the "Stop" button turns also into active state. The user can stop the process at any time. The "Exporting Progress" shows the current progress of the transformation and the "Current Action" shows the current PowerPoint® slide being transformed as shown in the figure 4.9.


PPT2ADL Generator (1:1)

Import

PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

☐ Select All

- ✓ 1 : Adonis Customising
- ✓ 2 : Why AdoScript?
- ✓ 3 : Levels of ADONIS customising
- ✓ 4 : The Message Port concept
- ✓ 5 : The LEO Language (1)
- ☐ 6 : The LEO Language (2)
- ☐ 7 : LEO Expressions
- ✓ 8 : The AdoScript Language
- ✓ 9 : Procedures in AdoScript
- ✓ 10 : Functions in AdoScript
- ✓ 11 : Expressions in AdoScript
- ✓ 12 : Expressions in AdoScript | Operators (1)
- ✓ 13 : Expressions in AdoScript | Operators (2)
- ✓ 14 : Expressions in AdoScript | Functions (1)

Export

LO Type: [● Content Item] ○ Practice Item ○ Assessment Item

LO/LOV Name:

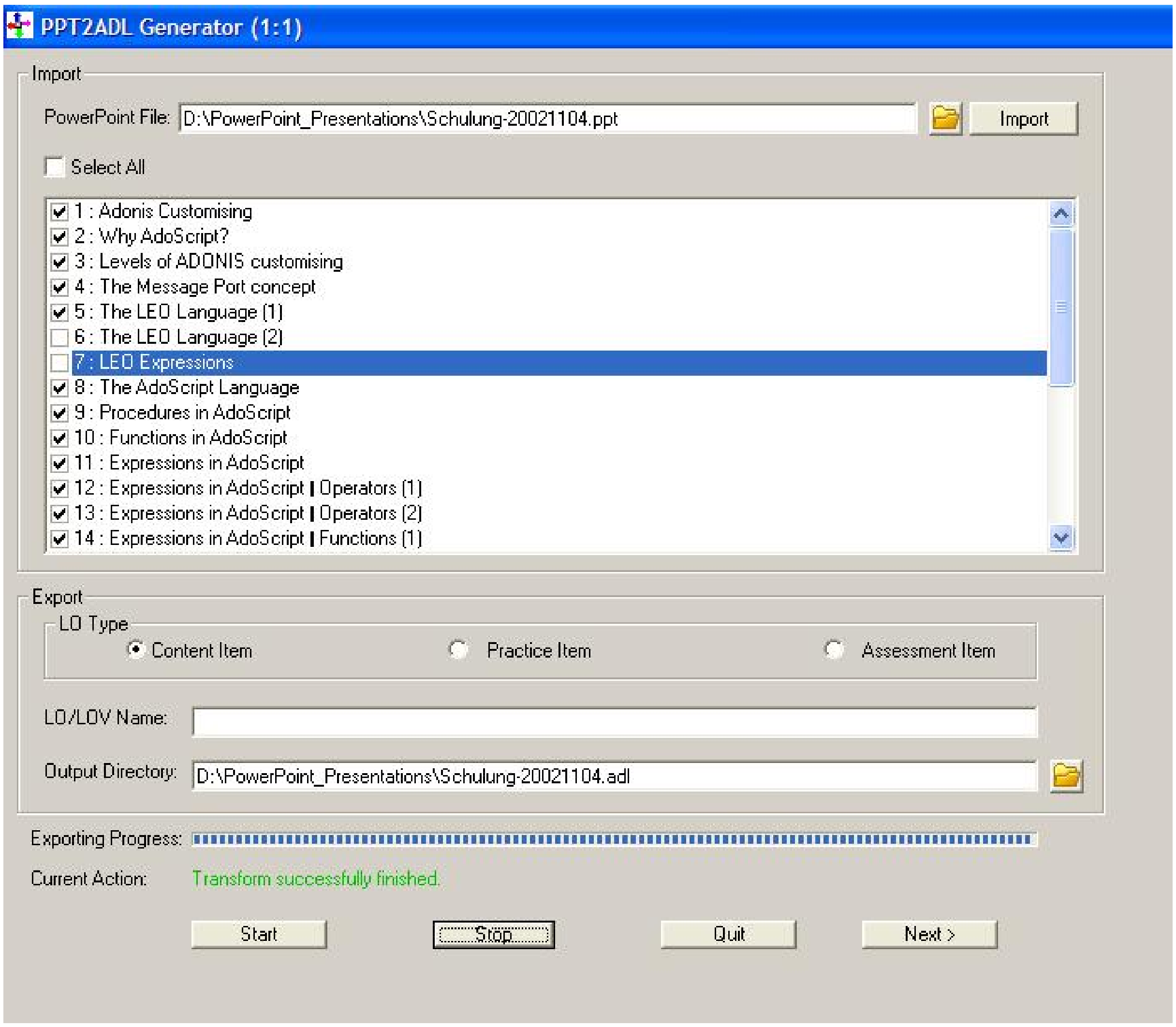
Output Directory: D:\PowerPoint_Presentations\Schulung-20021104.adl

Exporting Progress:

Current Action: Transforming the 10 of 26 Slides

Figure 4.9 The current transform state can be seen

11. After finishing the transform procedure the “Next >” button turns into active state and the user can go on with the importing into eduWEAVER as shown in the figures 4.10-4.11.



PPT2ADL Generator (1:1)

Import

PowerPoint File: D:\PowerPoint_Presentations\Schulung-20021104.ppt

[ ] Select All

[✓] 1 : Adonis Customising
[✓] 2 : Why AdoScript?
[✓] 3 : Levels of ADONIS customising
[✓] 4 : The Message Port concept
[✓] 5 : The LEO Language (1)
[ ] 6 : The LEO Language (2)
[ ] 7 : LEO Expressions
[✓] 8 : The AdoScript Language
[✓] 9 : Procedures in AdoScript
[✓] 10 : Functions in AdoScript
[✓] 11 : Expressions in AdoScript
[✓] 12 : Expressions in AdoScript | Operators (1)
[✓] 13 : Expressions in AdoScript | Operators (2)
[✓] 14 : Expressions in AdoScript | Functions (1)

Export

LO Type: [✓ Content Item] Practice Item Assessment Item

LO/LOV Name:

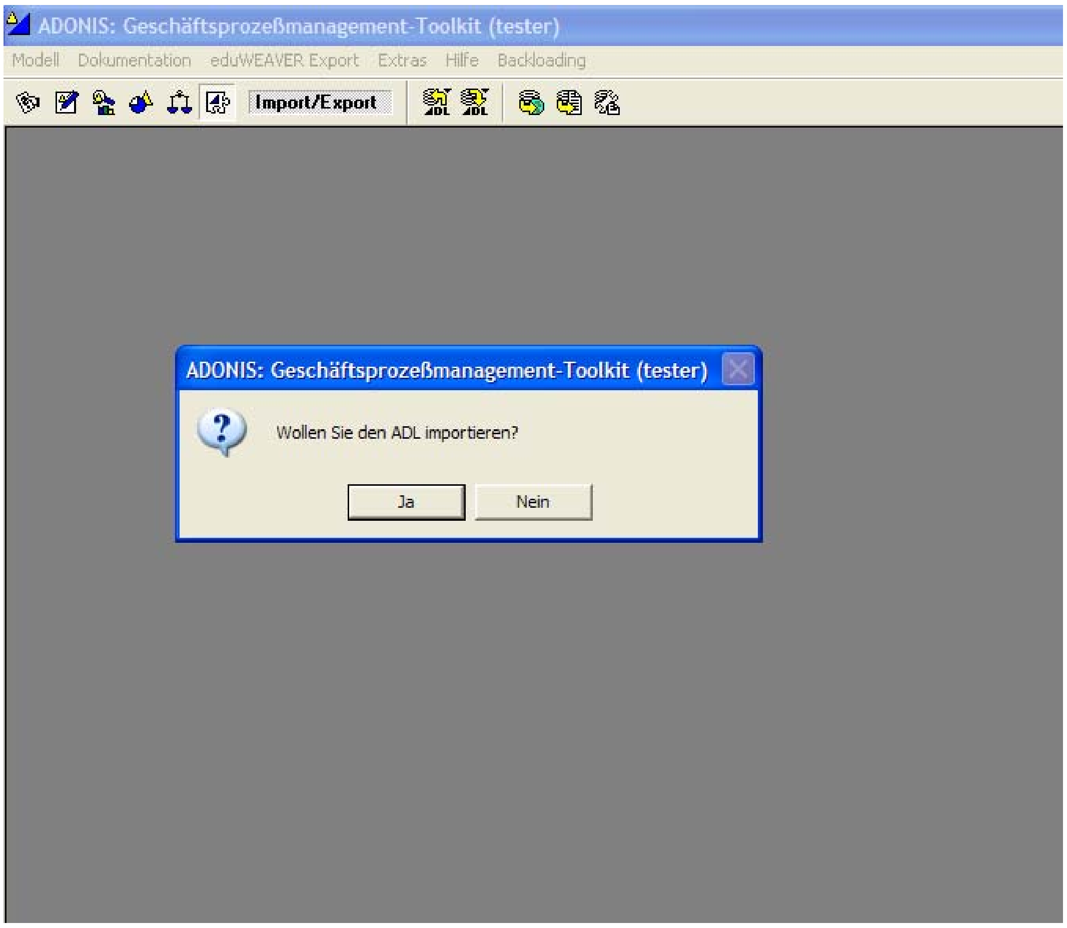
Output Directory: D:\PowerPoint_Presentations\Schulung-20021104.adl

Exporting Progress:

Current Action: Transform successfully finished.

Figure 4.10 The "Next" button turns into active state

12. The work flow of the application starts the "ADL Import" window and the user must show the path to the directory where the plug-in saved the .adl file as shown in the figures 4.11-12.


ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Modell
Dokumentation
eduWEAVER Export
Extras
Hilfe
Backloading
Import/Export
ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Ja
Nein

Wollen Sie den ADL importieren?

Figure 4.11 Pop-up window about the ADL import

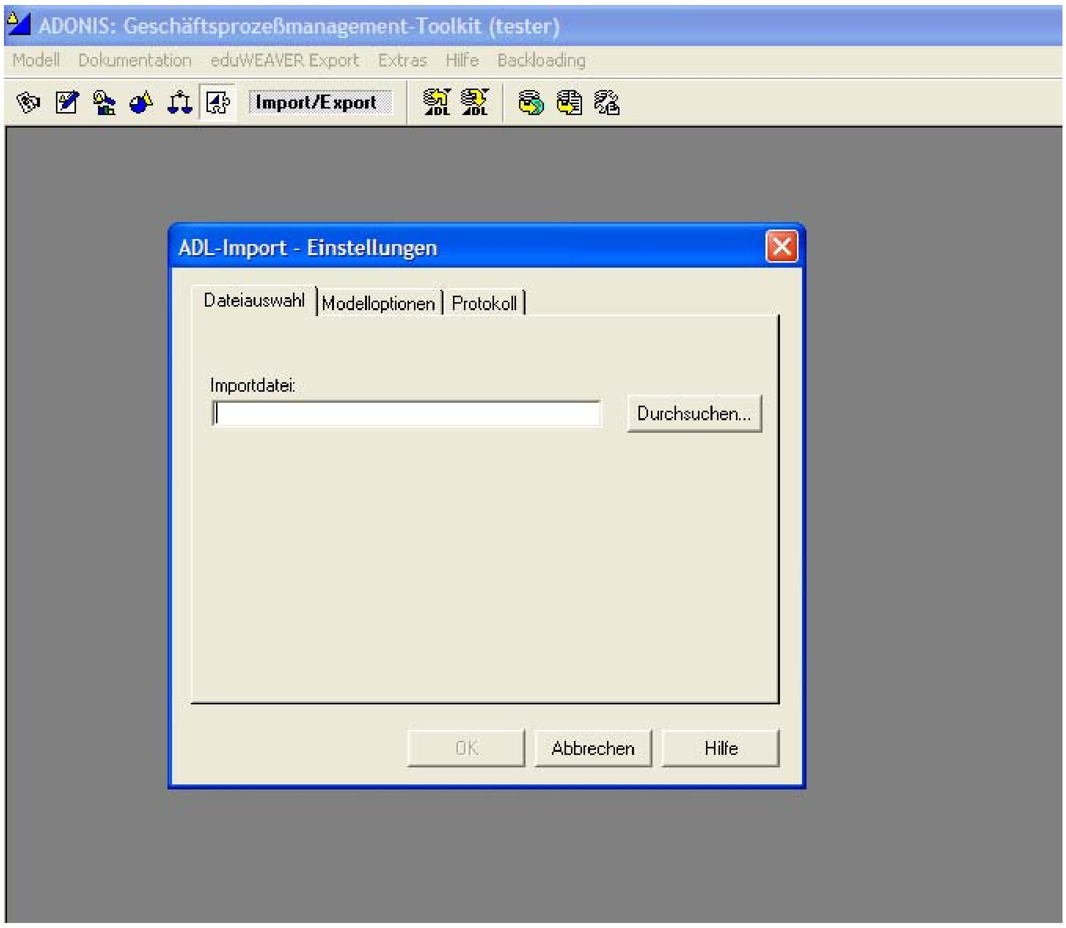
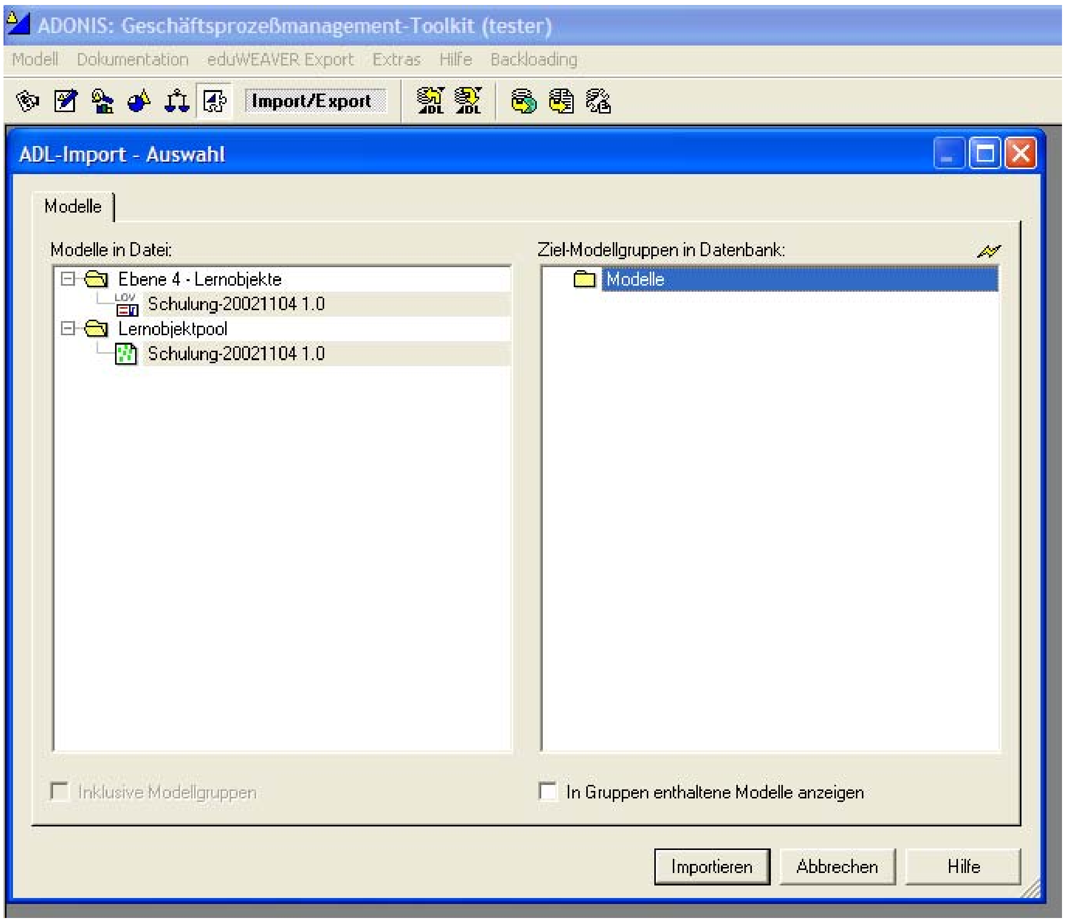

ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Modell Dokumentation eduWEAVER Export Extras Hilfe Backloading
Import/Export
ADL-Import - Einstellungen
Dateiauswahl
Modelloptionen
Protokoll
Importdatei:
Durchsuchen...
OK
Abbrechen
Hilfe

Figure 4.12 The ADL Import procedure in eduWEAVER (1)

13. The ADL Import follows the normal import procedure in eduWEAVER as shown in the figures 4.13-4.14.



Modelle in Datei:

- Ebene 4 - Lernobjekte
  - Schulung-20021104 1.0
- Lernobjektpool
  - Schulung-20021104 1.0

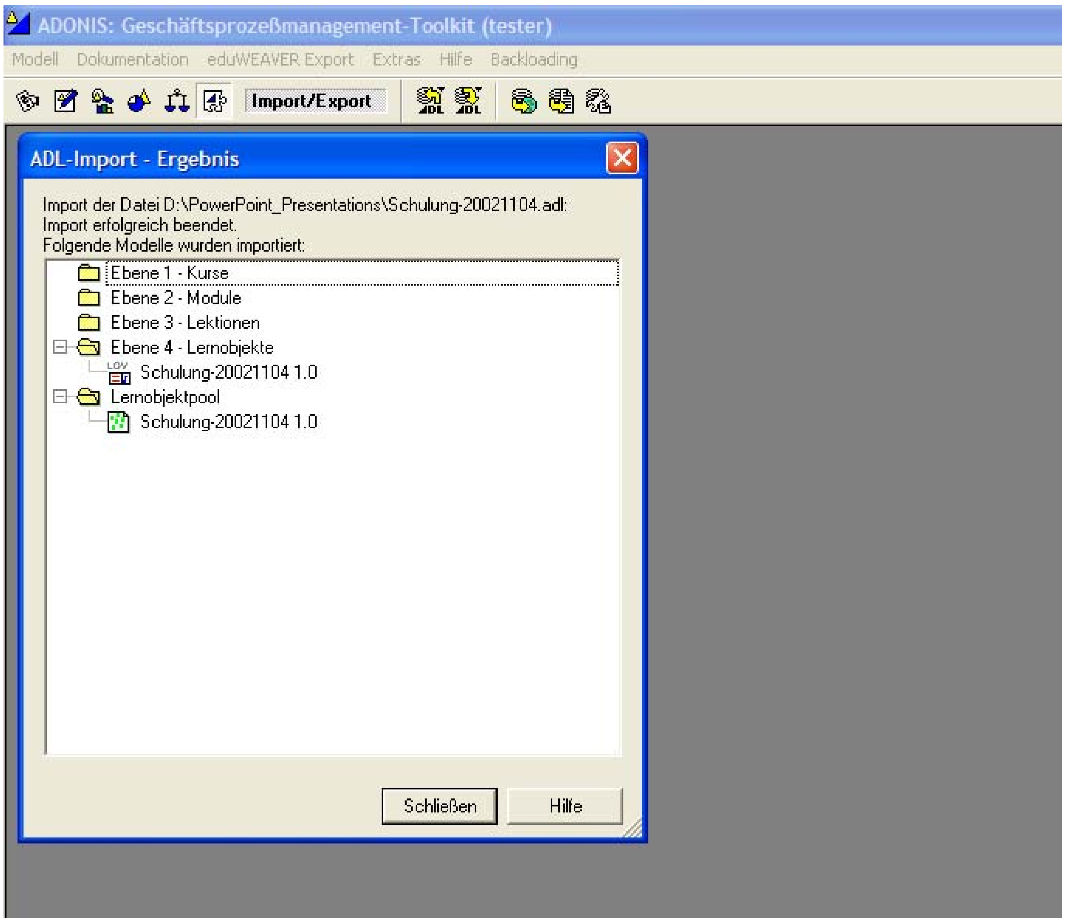
Ziel-Modellgruppen in Datenbank:

- Modelle

☐ Inklusive Modellgruppen

☐ In Gruppen enthaltene Modelle anzeigen

Figure 4.13 The ADL Import procedure in eduWEAVER (2)



Import der Datei D:\PowerPoint_Presentations\Schulung-20021104.adl:
Import erfolgreich beendet.
Folgende Modelle wurden importiert:

- Ebene 1 - Kurse
- Ebene 2 - Module
- Ebene 3 - Lektionen
- Ebene 4 - Lernobjekte
  - Schulung-20021104 1.0
- Lernobjektpool
  - Schulung-20021104 1.0

Figure 4.14 The ADL Import procedure in eduWEAVER (3)

14. After finishing the transformation process the user can quit the program or transform other PowerPoint® files.

The imported models will look like in the figures 4.15-4.16.

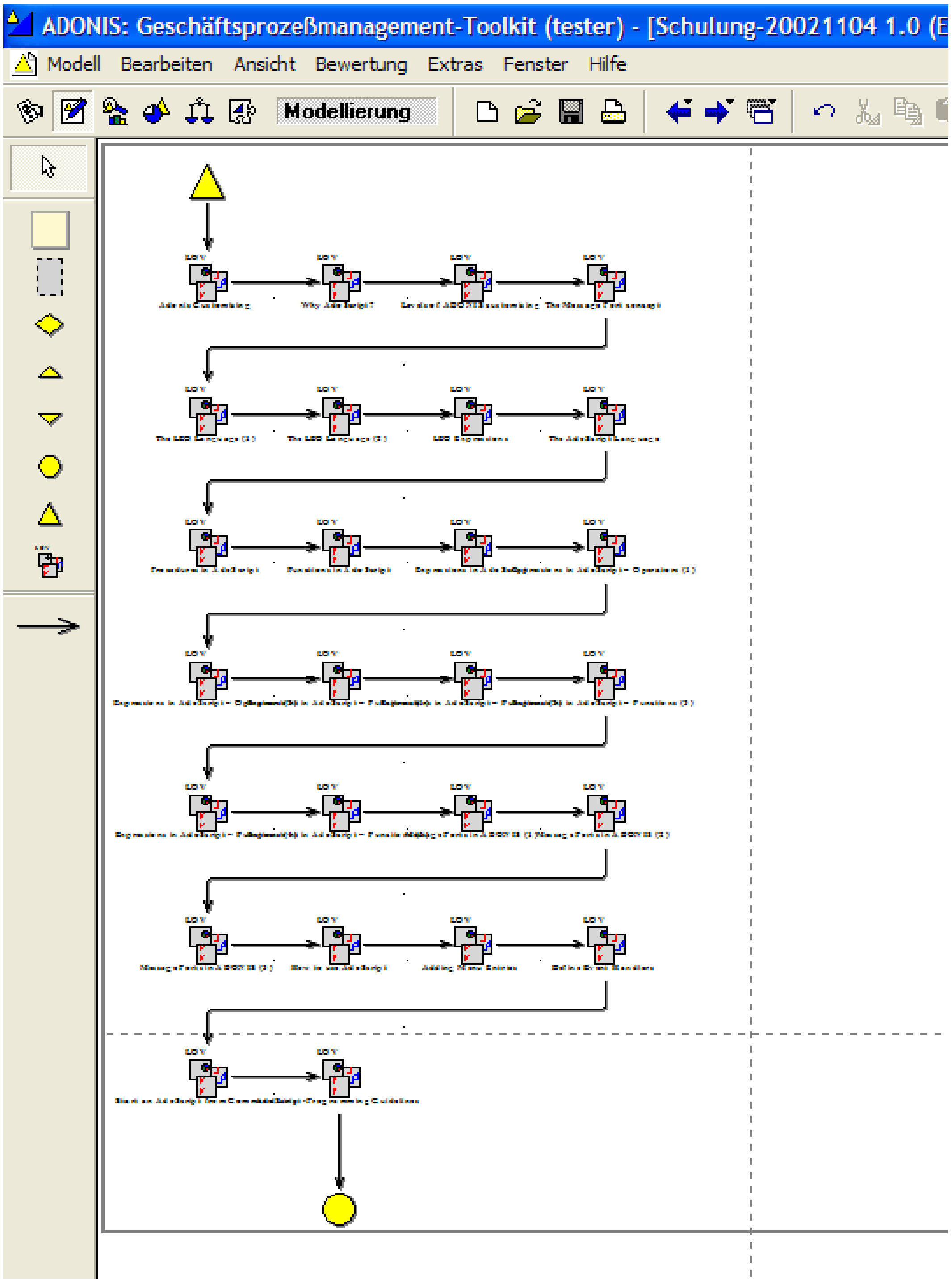

ADONIS: Geschäftsprozeßmanagement-Toolkit (tester) - [Schulung-20021104 1.0 (E
Modell
Bearbeiten
Ansicht
Bewertung
Extras
Fenster
Hilfe
Modellierung
LOV

Figure 4.15 The imported LOV Model in eduWEAVER

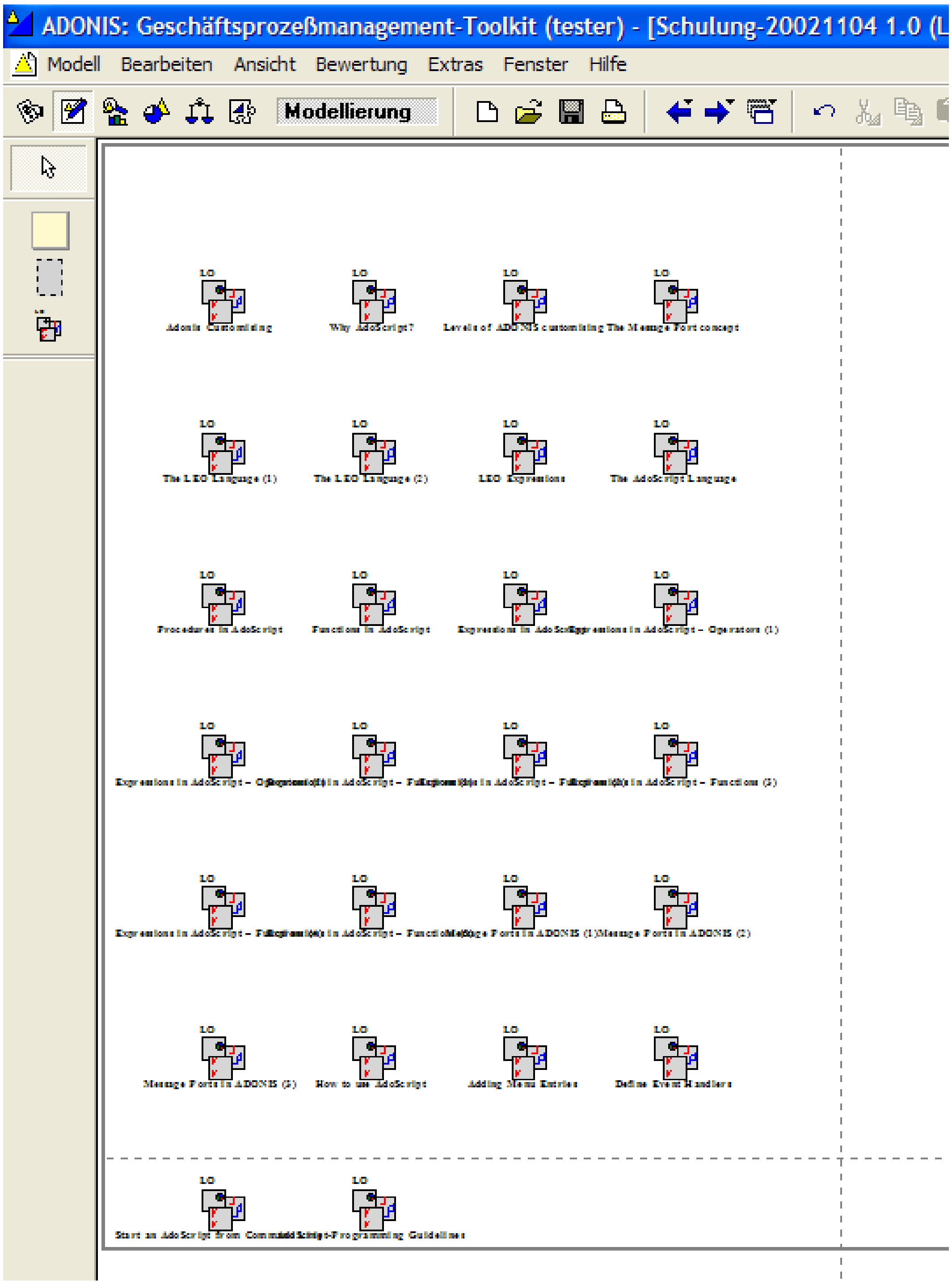
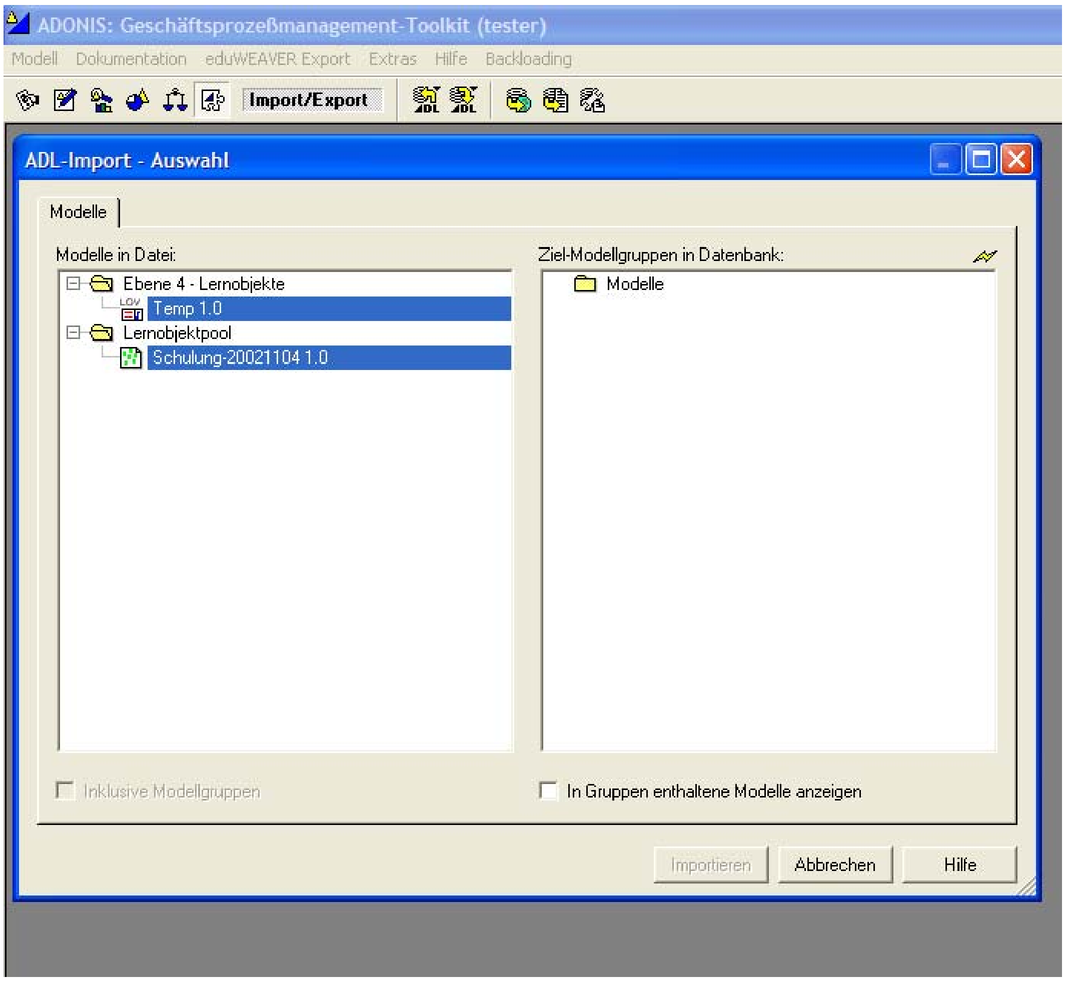


Figure 4.16 The imported LO Model in eduWEAVER

This possible workflow shows how easily the teaching staff can transform their PowerPoint® presentation into LOV/LO Models.

4.2.2 Scenario II

The second possible scenario is that all the PowerPoint® slides will be transformed into one LOV and corresponding LO. The steps of importing follow the same schema like steps 1-12 of the first scenario (see figure 4.1-4.12). The difference is the step 14 where another way of ADL import is happening. As shown in the figure 4.17 a dummy LOV is created (Temp 1.0 LOV). After successful import into another model eduWEAVER will delete that dummy LOV. If the user does not want to import the new LOV into another existing LOV she can reject the proposal of the pop up window shown in figure 4.18. After that she can rename the Temp 1.0 into a desired LOV name.


ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Modell Dokumentation eduWEAVER Export Extras Hilfe Backloading
Import/Export
ADL-Import - Auswahl
Modelle
Importieren
Abbrechen
Hilfe

Modelle in Datei:
- Ebene 4 - Lernobjekte
  - Temp 1.0
- Lernobjektpool
  - Schulung-20021104 1.0

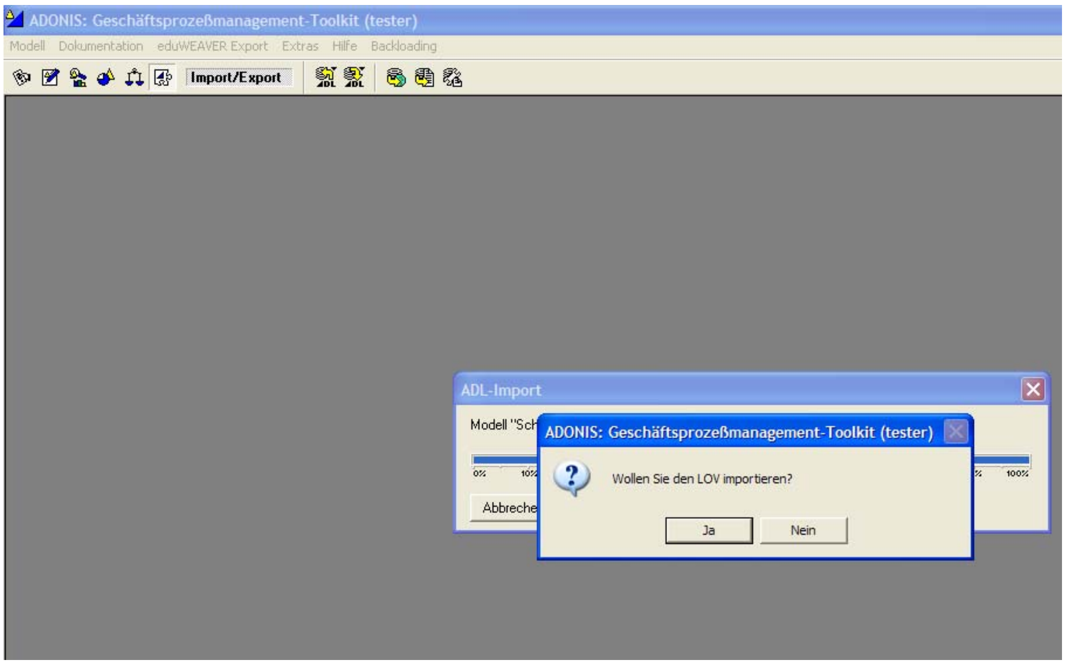
Ziel-Modellgruppen in Datenbank:
- Modelle

Inklusive Modellgruppen

In Gruppen enthaltene Modelle anzeigen

Figure 4.17 A dummy LOV (Temp 1.0) is created

The standard flow of the application will bring a pop-up window, asking the user if she want to import the LOV in an already existing one as shown in the figure 4.18.


ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Modell Dokumentation eduWEAVER Export Extras Hilfe Backloading
Import/Export
ADL-Import
Modell "Sch
Abbreche
0% 10% 100%
ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)
Ja
Nein

Wollen Sie den LOV importieren?

Figure 4.18 Ask to import the LOV

The application will show a collection of all possible (existing) LOV models where the user can import her model into (see figure 4.19).

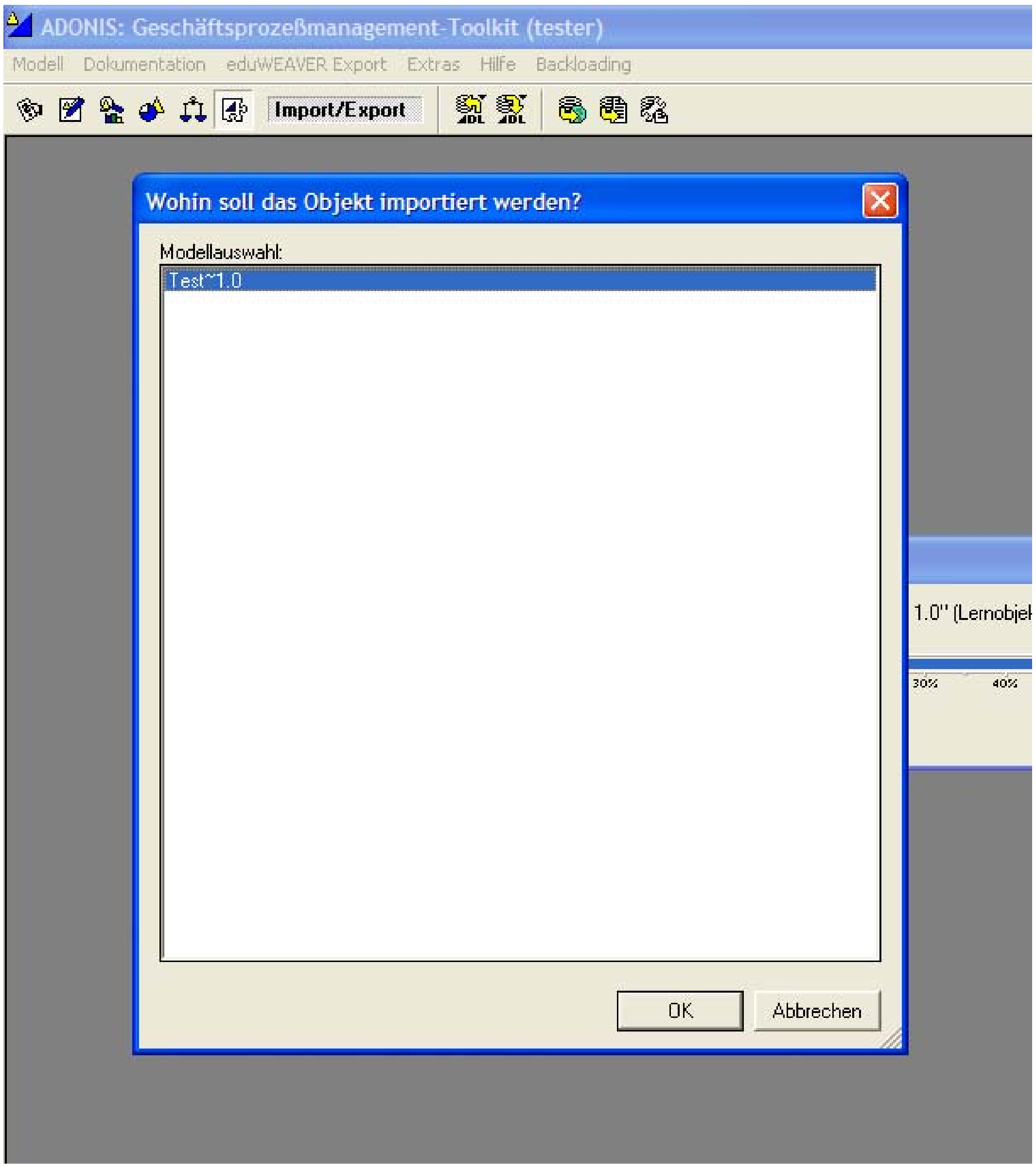
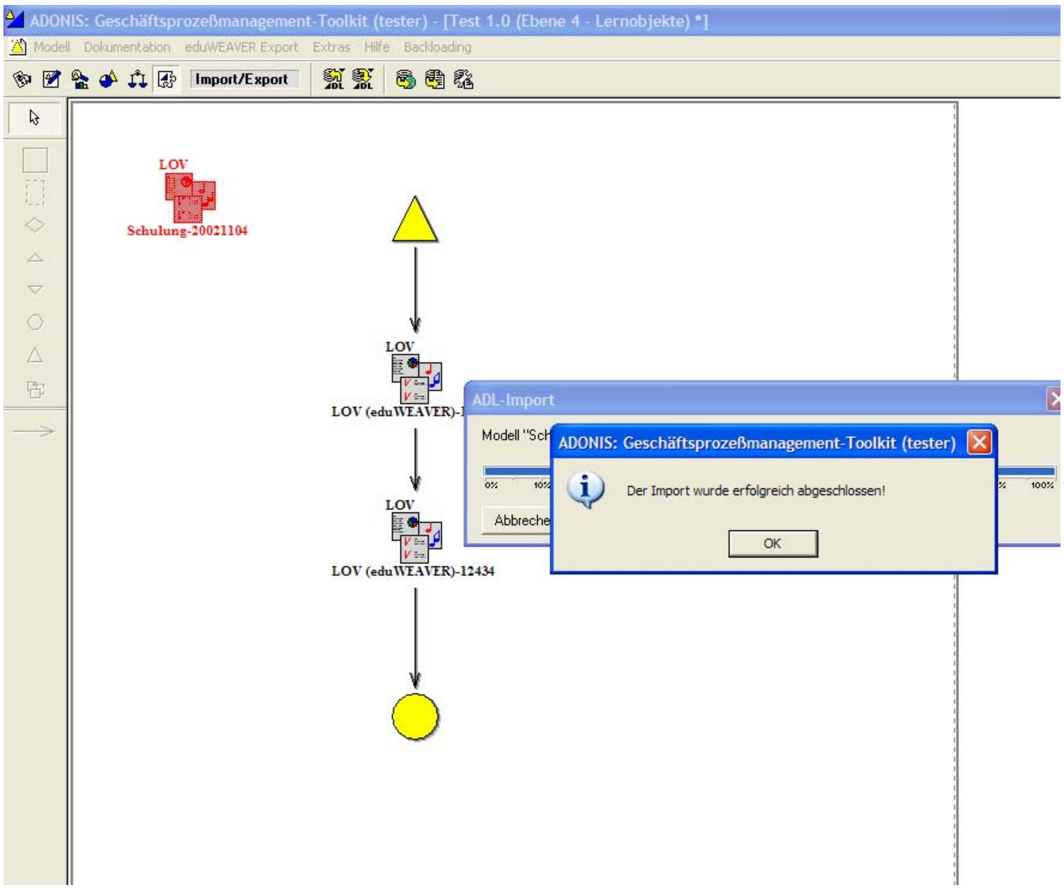


Figure 4.19 All existing LOV models

After the user choose the LOV model the application imports the new created LOV and put it at the upper left corner of the chosen LOV model as shown in the figure 4.20.


ADONIS: Geschäftsprozeßmanagement-Toolkit (tester) - [Test 1.0 (Ebene 4 - Lernobjekte) *]
Modell Dokumentation eduWEAVER Export Extras Hilfe Backloading
Import/Export
LOV
Schulung-20021104
LOV
LOV (eduWEAVER)-1
LOV
LOV (eduWEAVER)-12434
ADL-Import
Modell "Sch
Abbreche
100%

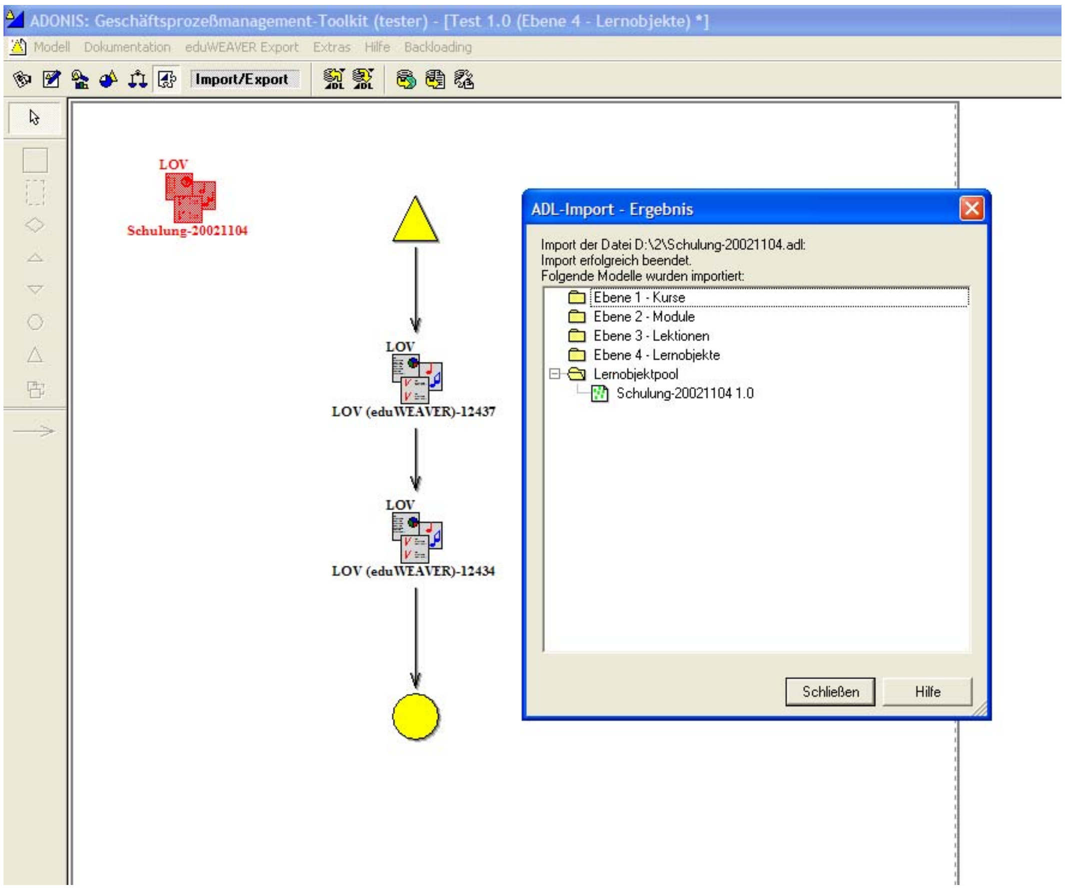
ADONIS: Geschäftsprozeßmanagement-Toolkit (tester)

Der Import wurde erfolgreich abgeschlossen!

OK

Figure 4.20 Successful import of the LOV

At the end the application prompts with the ADL import summary, which gives an overview of the imported models (see figure 4.21).


ADONIS: Geschäftsprozeßmanagement-Toolkit (tester) - [Test 1.0 (Ebene 4 - Lernobjekte) *]
Modell Dokumentation eduWEAVER Export Extras Hilfe Backloading
Import/Export
LOV
Schulung-20021104
LOV
LOV (eduWEAVER)-12437
LOV
LOV (eduWEAVER)-12434
ADL-Import - Ergebnis
Schließen
Hilfe

Import der Datei D:\2\Schulung-20021104.adl:
Import erfolgreich beendet.
Folgende Modelle wurden importiert:

- Ebene 1 - Kurse
- Ebene 2 - Module
- Ebene 3 - Lektionen
- Ebene 4 - Lernobjekte
- Lernobjektpool
  - Schulung-20021104 1.0

Figure 4.21 ADL Import summary

As shown above the second scenario and its application flow is also easy to understand and self explaining.

5 Conclusions

E-Learning course ware developing was a domain of the software developer some years ago. That situation changed in the recent years when the multimedia content developing application became more accessible and easily to cope with. The organizational situation of the teaching staff did not change a lot as only there are now expected to be up-to-date with the new technologies. Developing excellent e-Learning materials is now taken for granted but the extra time and strengths are not calculated. That is why the teaching staff community is always on search of comprehensive tools for easily designing of instructional materials. The idea of reusable high quality learning objects is one of the important prerequisites in the process of course ware development. The eduWEAVER modelling library embraced that idea from the very beginning by providing an open learning object pool. The teaching staff community is now enabled to exchange or reuse their multimedia materials or to use material created by professional authors. The export functionality of the tool helps the process of E-Learning broadcast through the creation of IMS packages consisting of the course structure and multimedia materials. The last step is the import of those packages in a learning management system (compliant with the IMS standard) allowing an easy delivering of web-based courses.

This diploma thesis shows a possible way of “backloading” procedure, meaning transforming learning materials (such as PowerPoint® presentations) and importing them into a learning process management system like eduWEAVER for further usage and transformation. The plug-in application also reveals the meta data saved in every Microsoft® Office document making the indexing and searching of content more easily. The program extracts also the content from the PowerPoint® presentation (putting it into .xml files) and enables the user to transform it into another presentational form.

The main idea behind every learning management system is to enable the user to cope with a large amount of learning materials, also organize and re-organize those materials into courses. The eduWEAVER plug-in serves exactly these main purposes. In addition, it enable a one step automatic transforming of PowerPoint® presentations into eduWEAVER *Learning Objects* saving time and efforts of the teaching staff who do not have to use many different applications until they get their work done.

The application build in this diploma thesis is the ground for further following plug-ins for eduWEAVER or other application which manage the exported XML and HTML files. On the next level the extracted meta information in the xml files can be indexed and made ready to search on the internet or on a university intranet. In the future when the web 3.0[97] (Semantic Web[98]) will become reality this meta information will become a standard for publishing content on the WWW. Another possible application can transform the content and the meta information into PDF files for printing of teaching materials or add another layout design to the online presentation of the Learning Objects. It is possible in the future application to have different presentation arts for the learning content for example one in Flash, one for mobile devices, one in plane HTML and so on. It is also possible to build a Learning Object Repository like a Web 2.0[99] environment (z.B. Wikipedia[100]) where the teaching staff can issue and edit Learning Objects in a team manner, which can raise the quality of the content and the satisfaction of the knowledge consumers.

[97] See Web 3.0.[Last visited 16.08.2007]
[98] See Semantic Web.[Last visited 16.08.2007]
[99] See Web 2.0.[Last visited 16.08.2007]
[100] See Wikipedia.[Last visited 16.08.2007]

V Bibliography

ADVISOR: Advanced instruction technology for services organisation.

(http://cordis.europa.eu/search/index.cfm?fuseaction=
proj.simpledocument&PJ_RCN=4014736&CFID=
2772438&CFTOKEN=53260157)
(http://www.boc.at/index.jsp?file=
WP_d5f091cc2d6dd919.dc1f04.107b31faa81.-7ff7&lg=en)
[Last visited 16.08.2007]

Akogrimo Consortium: Application Adaptation, Methods/Tools; Sixth Framework Programme; Editors: Woitsch R.; Schwab M., 2005

Bajnai, Judit / Karagiannis, Dimitris: ADVISOR® – Meta-Modeling Tool for Individual Instructional Design, 2004
(http://www.ro.feri.uni-mb.si/razno/icl2004/pdf/karagiannis.pdf)
[Last visited 04.05.2007]

Bajnai, Judit / Steinberger, Claudia: eduWEAVER – the Web-based Courseware Design Tool, 2003
(http://edubite.dke.univie.ac.at/IADISInternet2003.pdf)
[Last visited 04.05.2007]

Bates, Chris: XML in Theory and Practice, 1st edition, Wiley, Chichester, 2003

BOC Information Technologies Consulting AG
(http://www.boc-group.com/)
[Last visited 16.08.2007]

CodeGear
(http://www.codegear.com)
[Last visited 17.08.2007]

Cicso Systems Inc.: Reusable Learning Object Strategy, Definition, Creation Process and Guidelines for Building, Version 3.1, April 22, 2000
(http://www.reusablelearning.org/Docs/Cisco_rlo_roi_v3-1.pdf)
[Last visited 04.05.2007]

Dewath, Gizella: An Introduction to e-Learning: A Study of the Current State of e-Learning in the United Kingdom, 2004
(http://idp.bl.uk/downloads/e-Learning.pdf)
[Last visited 16.08.2007]

Doberenz, Walter / Kowalski, Thomas: Visual Basic 6, 1. Auflage, Carl Hanser Verlag, München, 1999

Eclipse - an open development platform
(http://www.eclipse.org/index.php)
[Last visited 17.08.2007]

eduBITE – Projektinformationen
(http://edubite.dke.univie.ac.at/Downloads.htm)
[Last visited 16.08.2007]

eduWEAVER
(www.formatex.org/micte2006/pdf/806-810.pdf)
[Last visited 06.05.2007]
(http://www.eduweaver.net)
[Last visited 25.11.2007]

Esprit Project: EU information technologies programme
(http://cordis.europa.eu/esprit/home.html)
[Last visited 16.08.2007]

Fill, Hans-Georg: UML Statechart Diagrams on the ADONIS Metamodeling Platform
(http://tfs.cs.tu-berlin.de/grabats/Final04/fill.pdf)
[Last visited 03.05.2007]

Freeze, Wayne: Visual Basic Developer's Guide to COM and Com+, 1st Edition, Sybex, USA, 2000

Geese, Elmar / Heiger, Markus / Lohrer, Matthias: XML, XSLT, VB und ASP, 1.Auflage, Galileo Computing, Bonn, 2001

Gunderloy, Mike: Parsing XML Data with MSXML
(http://msdn2.microsoft.com/en-us/library/aa163921(office.10).aspx)
[Last visited 04.05.2007]

Grillitsch, Silvia: Blended Learning
(http://elearningcenter.univie.ac.at/index.php?id=535)
[Last visited 16.08.2007]

Harold, Elliotte Rusty: XML Bible, 2nd edition, Hungry Minds, New York, 2001

Harold, Elliotte Rusty / Means, W. Scott: XML in a Nutshell, 2nd edition, O'Reilly, Sebastopol, 2002

Haupt, Horst: Visual Basic Referenz, 1.Auflage, Francis', Poing, 1999

HTML 4.01 Specification
(http://www.w3.org/TR/html401/)
[Last visited 16.08.2007]

Idris, Nazmul: Should I use SAX or DOM?
(http://developerlife.com/saxvsdom/default.htm)
[Last visited 04.05.2007]

IMS content packaging
(http://www.imsproject.org)
[Last visited 04.05.2007]

Karagiannis, Dimitris / Bajnai, Judit: eduXX – The Instructional Design Platform EISTA04 - International Conference on Education and Information Systems: Technologies and Applications, Orlando, 2004

Kühn, H., et al.: Re-Use in Business Process Management based on Metamodelling and Model Views. In: Bullinger, H.-J.; Vossen, P. H. (Eds.): Adjunct Conference Proceedings of the 8th International Conference on Human-Computer Interaction, HCI Int. '99, Munich, August 1999

Knobloch, Manfred / Kopp, Matthias: Web-Design mit XML, 1.Auflage, dpunkt.Verlag, Heidelberg, 2001

Lomax, Paul: VB & VBA in a Nutshell: The Language, 1st Edition, O'Reilly, Sebastopol, 1998

Martin, René: VBA mit Office 2000 lernen, 2.Auflage, Addison-Wesley, München, 1999

Mayhew, Deborah: The Usability Engineering Lifecycle, 1st Edition, Morgan Kaufmann, San Francisco, 1999

Meyer, Eric: Design Issues for Web-Based Teaching Materials

(http://meyerweb.com/eric/talks/www6/706/POSTER706.html)
[Last visited 12.11.2007]

Microsoft® Corporation: Microsoft® Office XP Developer's Guide, 1st Edition, Redmond, 2001

Microsoft® Office
(http://www.microsoft.com/austria/office/default.mspx)
[Last visited 17.08.2007]

Monadjemi, Peter: Jetzt lerne ich Visual Basic, 2.Auflage, Markt und Technik Verlag, Haar bei München, 1999

NetBeans IDE
(http://www.netbeans.org/index.html)
[Last visited 17.08.2007]

Nielsen, Jacob: Usability Engineering, Academic Press, 1st Edition, San Diego, 1993

Nielsen, Jacob: Ten Usability Heuristics, 2001
(http://www.useit.com/papers/heuristic/index.html)
[Last visited 03.05.2007]

Norman, D.: Things that makes us smart: Defending Human Attributes in the Age of the Mashine, Addison-Wesley Publishing, New York, 1994

PowerDesigner
(http://www.sybase.com/products/modelingmetadata/powerdesigner)
[Last visited 17.08.2007]

Preece, Jennifer. et al.:: Interaction design: beyond human – computer interaction, 1st Edition, John Wiley & Sons Ltd, New York, 2002

Puopolo, John P.: Writing OLE Controls, 1st edition, Prentice Hall, New Jersey, 1997

Raskin, Jef: Das intelligente Interface, 1.Auflage, Addison-Wesley Verlag, München, 2001

Ray, Erik T.: Learning XML, 1.Auflage, O’Reilly, Beijing [u.a.], 2001

Rogerson, Dale: Inside COM, 1st Edition, Microsoft Press, Redmond, 1997

Seelhofer, Martin: Microsoft® Office- Programmierung & Visual Basic for Applications, 1.Auflage, Smart Books, Kilchberg, 2003

Semantic Web
(http://www.semantic-web.at/36.175.35.catchword.kontext.semantic-web.htm)
[Last visited 16.08.2007]

Shneiderman, Ben / Plaisant, Catherine: Designing the User Interface, 4th Edition, Pearson Addison Wesley, Boston, 2005

Siler, Brian / Spotts, Jeff: Using Visual Basic 6, Special edition, Que, Indianapolis, 1998

Spolsky, Joel: User Interface Design for Programmers, 1st Edition, Apress, Berkeley, 2001

Steinberger, Claudia et al.: EduBITE: Educating Business and Information Technologies, Proceedings of e-Learn, Montreal, Canada, 2002

Visual Studio 2005
(http://www.microsoft.com/germany/msdn/vstudio/products/default.mspx)
[Last visited 17.08.2007]

Web 2.0
(http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1)
[Last visited 16.08.2007]

Web 3.0
(http://www.semantic-web.at/36.62.255.catchword.kontext.web-3-0.htm)
[Last visited 16.08.2007]

Wikipedia
(http://de.wikipedia.org/wiki/Hauptseite)
[Last visited 16.08.2007]

Wilson, Mark / Wilson, Tracey: XML Programmierung mit VB und ASP, 1.Auflage,
Addison-Wesley Verlag, München, 2001

World Wide Web Consortium
(http://www.w3.org/)
[Last visited 16.08.2007]

XML
(http://www.w3.org/XML/)
[Last visited 16.08.2007]

XML-DEV: XML-DEV Mail List
(http://www.xml.org/xml/xmldev.shtml)
[Last visited 03.05.2007]

VI Appendix

Abstract

At the end of the 90's E-Learning was a term with which large hopes were connected, regarding the improvement of learning. This improvement referred to several criteria. First one assumed the creation and management of the learning components will be simpler and above all more economical. Furthermore one believed that the entrance possibilities would be simplified, also in the sense that an individualized, based on the particular needs of the learning consumer's courseware would be issued. And finally the hope that more self organization in the learning process will become reality, which of the other side would cut the overall learning expenses.

All these intention did not find their full realization in the given period. The time line for this fundamental project was largely extended because of the lack of understanding the problems the participants were to face. One of those problems was the fact that quality courseware was not simple to create and when once created not simple to be managed. Because of the IT technological race it was difficult for the teaching staff to maintain their learning materials up-to-date. It was high time to build the tool not only for creating learning materials but also tools for their management. One such tool is eduWEAVER, an educational courseware design tool.

The aim of the diploma thesis will be to show a possible scenario in form of a plug-in application for the eduWEAVER. It should enable the easy transaction of existing class teaching materials (mostly Microsoft® Office files and particularly PowerPoint® presentations) into quality e-Learning materials with a minimum of manual work.

Keywords: eduWEAVER, E-Learning, ADONIS, Visual Basic, Visual Basic for Application, User Interfaces, XML, ADL, Power Point

Zusammenfassung

Am Ende der 90er Jahre war E-Learning ein Begriff, mit dem große Hoffnungen im Hinblick auf die Verbesserung des Lernens verbunden waren. Diese Verbesserung hatte mehrere Anhaltspunkte. Einer ging davon aus, dass die Entwicklung und das Management des Lehrmaterials einfacher und vor allem günstiger werden. Ein anderer war die Ansicht, dass eine breitere Akzeptanz geschaffen wird, in dem Sinne, dass eine individuelle, auf die besonderen Bedürfnisse des Verbrauchers angepasste Ausbildungslösung präsentiert wird. Und schließlich bestand die Hoffnung, dass mehr Selbstorganisation in dem Lernprozess zur Realität wird, und sich somit auch die gesamten allgemeinen Ausbildungskosten minimieren lassen. Alle diese Absichten wurden in der angegebenen Zeit bei weitem nicht verwirklicht. Die Zeit für die Realisierung dieses grundlegenden Projektes wurde stark erweitert, da der Mangel an Verständnis für die Probleme der Teilnehmer zu groß war. Ein Problem war die Tatsache, dass qualitativ hochwertige Kursmaterialen nicht einfach zu erstellen sind und, wenn sie einmal erstellt sind, nicht einfach zu verwalten sind.
Während der anhaltenden IT-technologischen Rennen ist es für die Lehrkräfte zunehmend schwierig geworden, ihre Lehrmaterialien up-to-date zu halten. Es war höchste Zeit ein Werkzeug zu präsentieren, das nicht nur für die Erstellung von Lernmaterialien, sondern auch für deren Verwaltung geeignet ist. Ein solches Instrument ist eduWEAVER, ein Bildungsmaterialen Design Tool.

Das Ziel der Diplomarbeit war es, eine mögliche Lösung der vorher besprochenen Probleme in Form einer Plug-In-Anwendung für eduWEAVER zu präsentieren. Diese sollte die Transformation der bestehenden Lehrmaterialien (meist Microsoft® Office Dateien und vor allem PowerPoint® Präsentationen) in hoch qualitative e-Learning Materialien mit einem Minimum an manueller Arbeit verwirklichen.

Schlüsselwörter: eduWEAVER, E-Learning, ADONIS, Visual Basic, Visual Basic for Application, User Interfaces, XML, ADL, Power Point

Curriculum Vitae

My name is Ivaylo Velikovski. I study International Economic at the University of Vienna. I began my study in the year 1996. Before that I studied for one year at the Economic University of Sofia, Bulgaria.

My priorities lay in the subjects of Economic Computer Science and Financial Analysis. During my study I have participate on additional courses like “Java Programming” at the Siemens Training Center in Vienna, “SAP Business Process Management Junior” at the SAP Business School Vienna and many other courses at the SAP Training Center in Vienna. Since Mai 2007 I work at IDS Scheer Austria at the department of SAP NetWeaver Enterprice Technologies.

import_into_model.asc

```
CC "AdoScript" QUERYBOX "Wollen Sie den LOV importieren?" yes-no
IF (endbutton = "no") { EXIT }
CC "Core" GET_MODEL_ID modelname:("Temp 1.0") modeltype:("Ebene 4 - Lernobjekte")

CC "Modeling" OPEN modelids:(STR modelid)

CC "Core" GET_ACCESS_MODE modelid:(modelid)

CC "Modeling" SELECT_ALL
CC "Modeling" COPY_SELECTED
CC "Core" DISCARD_MODEL modelid:(modelid)
CC "Core" DELETE_MODEL modelid:(modelid)

SET mod_list: ("")

CC "Core" GET_ALL_MODEL_VERSIONS modeltype:("Ebene 4 - Lernobjekte")

SET obj:(objid)

FOR modid in:(modelversionids)
{
  CC "Core" GET_MODEL_INFO modelid:(VAL modid)
  SET mod_list:(tokunion (mod_list, modelname + "~" + ver, "@"))
}


CC "Core" GET_CLASS_ID classname:("LOV (eduWEAVER)")

CC "AdoScript" LISTBOX entries:   (mod_list)
                         toksep:    ("@")
                         selection: ( token (mod_list, 0, "@"))
                         title:     ("Wohin soll das Objekt importiert werden?")
                         boxtext:   ("Modellauswahl")
                                                                  w:300
                         h:300

IF (endbutton = "cancel") { EXIT }
IF ((LEN selection - 1) > search (selection,"~",0)) { SET selmod:(replall(selection,"~" ," ")) }
ELSE { SET selmod:(copy (selection,0,search (selection, "~", 0))) }


CC "Core" GET_MODEL_ID modelname:(selmod) modeltype:("Ebene 4 - Lernobjekte")
CC "Modeling" OPEN modelids:(STR modelid)
CC "Core" GET_ACCESS_MODE modelid:(modelid)
CC "Modeling" PASTE modelid x:1.00cm y:1.00cm

#make the user happy
CC "AdoScript" INFOBOX "Der Import wurde erfolgreich abgeschlossen!"
```

External Coupling (Modeling Library Customising)

```
#----------------------------------------------
#--BACKLOADING--#
  ON_EVENT "EndADLImport"
{
IF (successful AND plgin_ver = "all")
{
CC "AdoScript" FREAD file:"import_into_model.asc"
EXECUTE (text)
}
}
##############
ITEM "PPT Import n:1" importexport: "Backloading" pos1:5
SETG plgin_ver:"all"
CC "Application" GET_VERSION version:version

  SET b:""
  FOR i from:8 to:(LEN version -1)
  {
  SET a:(version SUB i)
  SET b:(b + a)
   }

 CC "Core" GET_CURRENT_LIBS bplib:(bplib)

SET parameter:(b + "&" + bplib)
#CC "AdoScript" INFOBOX(parameter)
SYSTEM ("PPT2ADL_many2one.exe " + parameter)
CC "AdoScript" QUERYBOX "Wollen Sie den ADL importieren?" yes-no
IF (endbutton = "no") { EXIT }
CC "ImportExport" EXEC_ADL_IMPORT_DLG

###############
ITEM "PPT Import 1:1" importexport: "Backloading" pos1:6
SETG plgin_ver:"one"
CC "Application" GET_VERSION version:version

   SET b:""
  FOR i from:8 to:(LEN version -1)
  {
  SET a:(version SUB i)

   SET b:(b + a)
   }

 CC "Core" GET_CURRENT_LIBS bplib:(bplib)

SET parameter:(b + "&" + bplib)
#CC "AdoScript" INFOBOX(parameter)
SYSTEM ("PPT2ADL_one2one.exe " + parameter)
CC "AdoScript" QUERYBOX "Wollen Sie den ADL importieren?" yes-no
IF (endbutton = "no") { EXIT }
CC "ImportExport" EXEC_ADL_IMPORT_DLG
```

Program Code (PPT2ADL_one2one.exe)

Form_1to1.frm

```
Dim Path, pptFile, outputDir, stop_transform, item, n, version, lb_name, aprs As String

Private Sub Check1_Click()
'EventHandler for the slides selection table
If Check1.Value = vbChecked Then
    Call SendMessageLong(Slides.hwnd, LB_SETSEL, True, -1)
    Else:
    Call SendMessageLong(Slides.hwnd, LB_SETSEL, False, -1)
End If

End Sub

Private Sub Form_Load()

btnStop.Enabled = False
btnImport.Enabled = False
btnStart.Enabled = False
btnNext.Enabled = False

'reading the parameter coming from the EduWEAVER call
If Command = "" Then
    version = "3.7"
    Else:
    aprs = "&"
    b = Split(Command, aprs)
    version = b(0)
    lb_name = b(1)
End If

End Sub

Private Sub btnImport_Click()

'EventHandler for the "Import" Button

Dim oPPTPres_dummy, oPPTPres As PowerPoint.Presentation

Check1.Value = vbUnchecked

If (ppt_file.Text = "") Then
    MsgBox ("You should select a PowerPoint file first!")
    Else:
    Slides.Clear
    wait = "Please wait..."
    Slides.AddItem wait

'creat PowerPoint presentation object
    Set ppApp = CreateObject("PowerPoint.Application")

'check if PowerPoint installed on the user's computer
    If (ppApp = "") Then
        MsgBox ("You do not have PowerPoint installed on your computer. Please install PowerPoint and try again!")
        Action.Caption = ""
        Exit Sub
    End If

'open the PowerPoint presentation "behind the scene"
    Set oPPTPres = ppApp.Presentations.Open(pptFile, WithWindow:=msoFalse)
        slide_count = oPPTPres.Slides.Count
        Slides.Clear

'extract the slide titles for the selection screen
    For i = 1 To slide_count
        On Error Resume Next

        If oPPTPres.Slides(i).Shapes.HasTitle Then
```

```
            title_value = oPPTPres.Slides(i).Shapes.Title.TextFrame.TextRange.Text
            Else:
            title_value = i & " Slide"
        End If

        Slides.AddItem i & " : " & title_value
    Next i

End If
Check1.Value = vbChecked
ppApp.Quit
'Release variables
    Set oPPTPres = Nothing
    Set ppApp = Nothing

End Sub


Private Sub ppt_Click()

'EventHandler for the button "Open file" window
With CommonDlg1
    .CancelError = False
    .MaxFileSize = 2 * 1024
    .DialogTitle = "Please select a PowerPoint file!"
    .Filter = "Alle PowerPoint Präsentationen (*.ppt)|*.ppt"
    .Flags = cdlOfnOpenDefault + cdlOFNExplorer
    .ShowOpen
    FileName = .FileName
    ppt_file.Text = .FileName
    pptFile = ppt_file.Text
End With
'Import button will be activated if a PPT file was choosed
DoEvents: If (ppt_file.Text <> "") Then btnImport.Enabled = True Else btnImport.Enabled = False

End Sub

Private Sub adl_Click()

'EventHandler for the button "Save as" window
With CommonDlg

'check if PowerPoint presentation file has been selected
    If (CommonDlg1.FileName = "") Then
        MsgBox ("Please select a PowerPoint file first!")
        Exit Sub
    End If

    .CancelError = False
    .DialogTitle = "Save as"

'default name of the .adl file is the presentation name
    .FileName = Mid$(CommonDlg1.FileName, 1, Len(CommonDlg1.FileName) - 4) & ".adl"
    .Filter = "Alle ADONIS Definition Language Files(*.adl)|*.adl"
    .DefaultExt = "adl"
    .Flags = cdlOFNOverwritePrompt + cdlOFNPathMustExist
    .ShowSave
    Path = Left$(.FileName, Len(.FileName) - Len(.FileTitle))
    outPut.Text = .FileName
    outputDir = outPut.Text
End With

'Start button will be activated if a ADL file was choosed
DoEvents: If (outPut.Text <> "") Then btnStart.Enabled = True Else btnStart.Enabled = False

End Sub

Private Sub Slides_Click()

'EventHandler for the selection table
Dim d As Scripting.Dictionary
Set d = CreateObject("Scripting.Dictionary")
```

```
For i = 0 To Slides.ListCount - 1

    If (Slides.Selected(i) = True) Then
        res = res + 1
        d.Add i, "True"
    Else:
        res = res - 1
        d.Add i, "False"
    End If

Next i
If res = Slides.ListCount Then
    Check1.Value = vbChecked
    Else:
    Check1.Value = vbUnchecked

    For i = Slides.ListIndex To Slides.ListCount - 1
        Slides.Selected(i) = d(i)
    Next i

    For i = 0 To Slides.ListIndex
        Slides.Selected(i) = d(i)
    Next i

End If
If Slides.ListIndex = 0 Then
    Slides.Selected(0) = d(0)
End If

End Sub

Private Sub btnStart_Click()

'EventHandler for the "Start" button
    ProgressBar.Value = 0

If (ppt_file.Text = "" Or outPut.Text = "") Then
    MsgBox ("You should select a PowerPoint file and output directory first!")
    Else:
    transform
End If
End Sub

Private Sub btnStop_Click()

'EventHandler for the "Stop" button
    ProgressBar.Value = 0
    Action.Caption = ""
    stop_transform = True
    btnStart.Enabled = True
    Set OutStream = Nothing

End Sub

Private Sub btnNext_Click()

'EventHandler for "Next" button
End

End Sub

Private Sub btnQuit_Click()

'EventHandler for "Quit" button
End

End Sub

Private Sub transform()

Dim name_pres As String
Dim xmlDoc As DOMDocument
Dim propertiesNode As IXMLDOMElement
Dim propertyNode As IXMLDOMElement
Dim Index, Index1, Index2, i, n, slide_count, X, Y, X1, Y1 As Integer
Dim oPPTPres, oPPTPres_dummy As PowerPoint.Presentation
Dim FileFormat As PpSaveAsFileType
Dim propertyName, propertyValue, content, tect_effect, title_value, notes, _
content_textTable, content_notes, description, content_textEffect As String
Dim FSys As New FileSystemObject
Dim OutStream As TextStream

btnStop.Enabled = True

'choose LO Type
If (CI.Value = True) Then
    item = "CI"
    ElseIf (PI.Value = True) Then
    item = "PI"
    Else:
    item = "AI"
End If

    stop_transform = False
    Action.Caption = "Please wait..."
    Action.ForeColor = &HFF&

'creat PowerPoint presentation object
Set ppApp = CreateObject("PowerPoint.Application")

'open the PowerPoint presentation "behind the scene"
Set oPPTPres = ppApp.Presentations.Open(pptFile, WithWindow:=msoFalse)


    ProgressBar.Value = 0
    btnStart.Enabled = False
    title_1 = "<Start (eduWEAVER)> : <Start (eduWEAVER)>"

'if all slides are selected, then copy the open presentation 1 to 1 in the new dummy one
If (Slides.SelCount = oPPTPres.Slides.Count) Then
    Set oPPTPres_dummy = oPPTPres
    Else:

'if not all slides are selected, create a new dummy presentation
    Set oPPTPres_dummy = ppApp.Presentations.Add
    For f = Slides.ListCount - 1 To 0 Step -1

'copy the selected slides in the new dummy presentation
        If (Slides.Selected(f) = True) Then
            oPPTPres_dummy.Slides.InsertFromFile FileName:=pptFile, Index:=0, SlideStart:=f + 1, SlideEnd:=f + 1
        End If
    Next f

End If

slide_count_dummy = Slides.SelCount

For n = 1 To slide_count_dummy
    On Error Resume Next
    content = ""

'Creating an XML file
    Set xmlDoc = Nothing
    Set xmlDoc = CreateObject("Microsoft.XMLDOM")

'root element is <PowerPoint_document>
       xmlDoc.loadXML "<PowerPoint_document/>"
    Set titelNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("titel"))
        strFilename = Mid$(oPPTPres.Name, 1, Len(oPPTPres.Name) - 4)
        titelNode.Text = CStr(strFilename)
    Set propertiesNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("builtInProperties"))

'attaching the "BuiltInDocumentProperties" to the XML file
For Each G In oPPTPres.BuiltInDocumentProperties
      On Error Resume Next
```

```
       If Err.Number = 0 Then
            propertyName = G.Name
            propertyValue = _
            oPPTPres.BuiltInDocumentProperties.item(propertyName).Value
            propertyName = Replace(propertyName, " ", "_")
            Set propertyNode = _
            propertiesNode.appendChild(xmlDoc.createElement(propertyName))
            propertyNode.Text = CStr(propertyValue)
        End If
      Err.Clear
Next

'attaching the "customProperties" to the XML file
Set propertiesNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("customProperties"))
For Each q In oPPTPres.CustomDocumentProperties
      On Error Resume Next

      If Err.Number = 0 Then
            propertyName = q.Name
            propertyValue = _
            oPPTPres.CustomDocumentProperties.item(propertyName).Value
            propertyName = Replace(propertyName, " ", "_")
         Set propertyNode = _
         propertiesNode.appendChild(xmlDoc.createElement(propertyName))
            propertyNode.Text = CStr(propertyValue)
       End If
            Err.Clear
Next

'attaching the slide title to the XML
Set Slide_title = xmlDoc.documentElement.appendChild(xmlDoc.createElement("slide_title"))
        If oPPTPres_dummy.Slides(n).Shapes.HasTitle Then
            title_value = oPPTPres_dummy.Slides(n).Shapes.Title.TextFrame.TextRange.Text
            Slide_title.Text = CStr(title_value)
            Else:
            title_value = n & " Slide"
            Slide_title.Text = CStr(title_value)
        End If
Set Slide = _
xmlDoc.documentElement.appendChild(xmlDoc.createElement("slide_content"))


'progress bar made visable
    ProgressBar.Visible = True
    ProgressBar.Value = ProgressBar.Value + (Round(100 \ oPPTPres_dummy.Slides.Count))
    Action.Caption = "Transforming the " & n & " of " & oPPTPres_dummy.Slides.Count & " Slides"

'attaching the content of the slides notes to the XML file
For Index = 1 To oPPTPres_dummy.Slides(n).Shapes.Count
      On Error Resume Next
      text_content = ""
      text_content = oPPTPres_dummy.Slides(n).Shapes(Index).TextFrame.TextRange.Text

'extracting the content of the slides notes
      content_notes = oPPTPres_dummy.Slides(n).NotesPage.Shapes(Index).TextFrame.TextRange.Text
      content_textEffect = ""
      content_textEffect = oPPTPres_dummy.Slides(n).Shapes(Index).textEffect.Text

'save the text into the xml file
    If (text_content <> "") Then
        Set Text = xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
        Set text_content_CDATA = xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(text_content))
        Text.appendChild text_content_CDATA
        Slide.appendChild Text

       content = content & text_content & vbCr

'Text export when text as a text effect

        ElseIf (content_textEffect <> "") Then
        Set Text = _
        xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
        Set content_textEffect_CDATA =
xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_textEffect))
        Text.appendChild content_textEffect_CDATA
        Slide.appendChild Text
        textEffect = textEffect & content_textEffect & vbCr
    End If


'Text export when text in a table

    For Index2 = 1 To oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows.Count
        For Index3 = 1 To oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows(Index2).cells.Count
        On Error Resume Next

            content_textTable = ""
            content_textTable = content_textTable & vbCr &
oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows(Index2).cells(Index3).Shape.TextFrame.TextRange.Text

            If (content_textTable <> "") Then
               Set Text = _
               xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
               Set text_content_CDATA = _
               xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_textTable))
               Text.appendChild text_content_CDATA
               Slide.appendChild Text
               content_Table = content_Table & content_textTable & vbCr
            End If
        Next Index3
    Next Index2
    Next Index


'Notes export

    If (content_notes <> "") Then
        Set notes = xmlDoc.documentElement.appendChild(xmlDoc.createElement("notes"))
        Set notes_inhalt_CDATA = _
        xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_notes))
        notes.appendChild notes_inhalt_CDATA
        Slide.appendChild notes
    End If

'save properties to metaX.xml
    Set PropertiesToXML = xmlDoc
        PropertiesToXML.Save Path & "meta_" & n & ".xml"


    description = content & textEffect & content_textTable & content_notes
    description = Replace(description, """", "\""")

'cleaning up the new line notation
    For i = 1 To 31
        description = Replace(description, Chr(i), Chr(13))
    Next i


'max.3700 characters in description part of the LOV and LO
    If (Len(description) > 3700) Then

'if more than 3700 characters than the description should list only the slides titles
        description = description & n & " Slide" & vbNewLine
    End If

'Graphics coordination for the LO in the ADL file
    If (slide_count_dummy Mod 4 = 0) Then
        x_end = "15"
        y_end = (slide_count_dummy \ 4) * 4 + 4
        ElseIf (slide_count_dummy Mod 4 = 1) Then
        x_end = "3"
        y_end = (slide_count_dummy \ 4) * 4 + 8
        ElseIf (slide_count_dummy Mod 4 = 2) Then
        x_end = "7"
        y_end = (slide_count_dummy \ 4) * 4 + 8
        ElseIf (slide_count_dummy Mod 4 = 3) Then
        x_end = "11"
        y_end = (slide_count_dummy \ 4) * 4 + 8
    End If

'the .adl file name take the name of the presentation (if not extra changed by the user)
    AdlFile = CommonDlg.FileName
    Set OutStream = FSys.CreateTextFile(AdlFile, True, False)

    If (n Mod 4 = 1) Then
        X1 = 3
        Else:
        X1 = X1 + 4
    End If

    If (n / 4 <= 1) Then
        Y1 = 4
        ElseIf (n / 4 > 1 And (n Mod 4) > 0 And (n Mod 4) < 4) Then
        Y1 = 4 * (n \ 4 + 1)
        ElseIf (n / 4 > 1 And (n Mod 4) = 0) Then
        Y1 = 4 * (n \ 4)
    End If

'by default the name of the LO/LOV models will be the PPT presentation name (if the name of the LO /LOV is not exctra set by
the user)
    If (LO_LOV_name.Text <> "") Then
        LO_name_model = LO_LOV_name.Text
        LOV_name_model = LO_LOV_name.Text
        Else:
        LO_name_model = Left$(CommonDlg1.FileTitle, Len(CommonDlg1.FileTitle) - 4)
        LOV_name_model = Left$(CommonDlg1.FileTitle, Len(CommonDlg1.FileTitle) - 4)
    End If

'seting the elements of the adl file
        learn_object_use_definition = vbNewLine & "VERSION <" + version + ">" _
        & vbNewLine & "BUSINESS PROCESS MODEL <" + LOV_name_model + " 1.0> : <" + lb_name + ">" _
        & vbNewLine & "VERSION <1.0>" & vbNewLine & "TYPE <Ebene 4 - Lernobjekte>" & vbNewLine & "INSTANCE <Start
(eduWEAVER)> : <Start (eduWEAVER)>" _
        & vbNewLine & "ATTRIBUTE <Position>" & vbNewLine & "VALUE ""NODE x:3cm y:1.00cm""" & vbNewLine & "INSTANCE
<Ende (eduWEAVER)> : <Ende (eduWEAVER)>" _
        & vbNewLine & "ATTRIBUTE <Position>" & vbNewLine & "VALUE ""NODE x:" & x_end & "cm y:" & y_end & "cm"""

        learn_object_use = learn_object_use & vbNewLine & "INSTANCE <" & Slide_title.Text & "> : <LOV (eduWEAVER)>" &
vbNewLine & "ATTRIBUTE <Position>" _
        & vbNewLine & "VALUE ""NODE x:" & X1 & "cm y:" & Y1 & "cm""" & vbNewLine & "ATTRIBUTE <Beschreibung>" _
        & vbNewLine & "VALUE """ & description & """" & vbNewLine & "ATTRIBUTE <Referenzobjekt>" _
        & vbNewLine & "VALUE ""REF mt:\""Lernobjektpool\"" m:\""" & LO_name_model & " 1.0" & "\"" c:\""LO (eduWEAVER)\""
i:\""" & Slide_title.Text & "\""""" & vbNewLine


        relation = relation & vbNewLine & "RELATION <Nachfolger>" & vbNewLine & "FROM " & title_1 & vbNewLine & "TO <" &
Slide_title.Text & "> : <LOV (eduWEAVER)>" & vbNewLine


    If (n Mod 4 = 1 And n <> 1) Then
        relation = relation & "ATTRIBUTE <Positions>" & vbNewLine & _
        "VALUE ""EDGE 2 x1:15.00cm y1:" & Y1 - 2 & "cm x2:3.00cm y2:" & _
        Y1 - 2 & "cm""" & vbNewLine
    End If

    title_1 = "<" & Slide_title.Text & "> : <LOV (eduWEAVER)>"

    pool_definition = "BUSINESS PROCESS MODEL <" & LO_name_model & " 1.0" & "> : <" + lb_name + ">" _
    & vbNewLine & "VERSION <1.0>" & vbNewLine & "TYPE <Lernobjektpool>" & vbNewLine
    learnobjectpool = learnobjectpool & vbNewLine & "INSTANCE <" & Slide_title.Text & "> : <LO (eduWEAVER)>" _
    & vbNewLine & "ATTRIBUTE <Position>" & "VALUE ""NODE x:" & X1 & "cm y:" & Y1 & "cm""" _
    & vbNewLine & "ATTRIBUTE <Beschreibung>" & vbNewLine & "VALUE """ _
    & description & """" & vbNewLine & "ATTRIBUTE <Einstiegspunkt (" & item & " 1)>" _
    & vbNewLine & "VALUE ""ITEM \""<automatisch>\"" param:\""" _
    & Replace(Path, "\", "\\\\") & Mid$(CommonDlg1.FileTitle, 1, Len(CommonDlg1.FileTitle) - 4) _
    & "_" & n & ".html\""""" _
    & vbNewLine
```

```
'export of the dummy PPT presentation slides to HTML files
  With ppApp.DefaultWebOptions
    .FrameColors = ppFrameColorsWhiteTextOnBlack
    .IncludeNavigation = True
    .ResizeGraphics = True
  End With

  With oPPTPres_dummy
    .WebOptions.ResizeGraphics = True
    With .PublishObjects(1)
      .FileName = Path & Mid$(CommonDlg1.FileTitle, 1, Len(CommonDlg1.FileTitle) - 4) & _
      "_" & n & ".html"
      .SourceType = ppPublishSlideRange
      .SpeakerNotes = True
      .HTMLVersion = ppHTMLDual
      .RangeStart = n
      .RangeEnd = n
      .Publish
    End With
  End With

  DoEvents: If stop_transform Then Exit Sub
  DoEvents: If stop_transform Then ProgressBar.Value = 0

Next n

  btnStart.Enabled = True
  btnNext.Enabled = True

'closing the PPT presentations
  oPPTPres_dummy.Close
  oPPTPres.Close

'quit the PPT
  ppApp.Quit

 'write the adl file
  OutStream.WriteLine vbNewLine & learn_object_use_definition & learn_object_use
  OutStream.WriteLine vbNewLine & relation & vbNewLine & "RELATION <Nachfolger>" & vbNewLine & "FROM " & title_1 &
vbNewLine & "TO <Ende (eduWEAVER)> : <Ende (eduWEAVER)>"
  OutStream.WriteLine vbNewLine & pool_definition & learnobjectpool
  Set OutStream = Nothing
  ProgressBar.Value = 100
  Action.Caption = "Transform successfully finished."
  Action.ForeColor = &HC000&

End Sub
```

Program Code (PPT2ADL_many2one.exe)

Form_Nto1.frm

```
Dim Path, pptFile, outputDir, stop_transform, item, n, version, lb_name, aprs As String


Private Sub Check1_Click()
'EventHandler for the slides selection table
If Check1.Value = vbChecked Then
    Call SendMessageLong(Slides.hwnd, LB_SETSEL, True, -1)
    Else:
    Call SendMessageLong(Slides.hwnd, LB_SETSEL, False, -1)
End If

End Sub

Private Sub Form_Load()

btnStop.Enabled = False
btnImport.Enabled = False
btnStart.Enabled = False
btnNext.Enabled = False

'reading the parameter coming from the EduWEAVER call
If Command = "" Then
    version = "3.7"
    Else:
    aprs = "&"
    b = Split(Command, aprs)
    version = b(0)
    lb_name = b(1)
End If

End Sub

Private Sub btnImport_Click()

'EventHandler for the "Import" Button

Dim oPPTPres_dummy, oPPTPres As PowerPoint.Presentation

Check1.Value = vbUnchecked

If (ppt_file.Text = "") Then
    MsgBox ("You should select a PowerPoint file first!")
    Else:
    Slides.Clear
    wait = "Please wait..."
    Slides.AddItem wait

'creat PowerPoint presentation object
    Set ppApp = CreateObject("PowerPoint.Application")

'check if PowerPoint installed on the user's computer
If (ppApp = "") Then
    MsgBox ("You do not have PowerPoint installed on your computer. Please install PowerPoint and try again!")
    Action.Caption = ""
    Exit Sub
End If

'open the PowerPoint presentation "behind the scene"
 Set oPPTPres = ppApp.Presentations.Open(pptFile, WithWindow:=msoFalse)
    slide_count = oPPTPres.Slides.Count
    Slides.Clear

'extract the slide titles for the selection screen
For i = 1 To slide_count
    On Error Resume Next
    If oPPTPres.Slides(i).Shapes.HasTitle Then
        title_value = _
        oPPTPres.Slides(i).Shapes.Title.TextFrame.TextRange.Text
        Else: title_value = i & " Slide"
    End If

    Slides.AddItem i & " : " & title_value

Next i

End If
Check1.Value = vbChecked
ppApp.Quit

'Release variables
    Set oPPTPres = Nothing
    Set ppApp = Nothing


End Sub


Private Sub ppt_Click()

'EventHandler for the button "Open file" window
With CommonDlg1
    .CancelError = False
    .MaxFileSize = 2 * 1024
    .DialogTitle = "Please select a PowerPoint file!"
    .Filter = "Alle PowerPoint Präsentationen (*.ppt)|*.ppt"
    .Flags = cdlOfnOpenDefault + cdlOFNExplorer
    .ShowOpen
    FileName = .FileName
    ppt_file.Text = .FileName
    pptFile = ppt_file.Text
End With

'Import Button will be activated if a PPT file was choosed
DoEvents: If (ppt_file.Text <> "") Then btnImport.Enabled = True Else btnImport.Enabled = False

End Sub

Private Sub adl_Click()

'EventHandler for the button "Save as" window
With CommonDlg

'check if PowerPoint presentation file has been selected
    If (CommonDlg1.FileName = "") Then
        MsgBox ("Please select a PowerPoint file first!")
        Exit Sub
    End If

    .CancelError = False
    .DialogTitle = "Save as"

'default name of the .adl file is the presentation name
    .FileName = Mid$(CommonDlg1.FileName, 1, Len(CommonDlg1.FileName) - 4) & ".adl"
    .Filter = "Alle ADONIS Definition Language Files(*.adl)|*.adl"
    .DefaultExt = "adl"
    .Flags = cdlOFNOverwritePrompt + cdlOFNPathMustExist
    .ShowSave
    Path = Left$(.FileName, Len(.FileName) - Len(.FileTitle))
    outPut.Text = .FileName
    outputDir = outPut.Text
End With
DoEvents: If (outPut.Text <> "") Then btnStart.Enabled = True Else btnStart.Enabled = False

End Sub

Private Sub Slides_Click()
'EventHandler for the selection table
```

```
Dim d As Scripting.Dictionary
Set d = CreateObject("Scripting.Dictionary")

For i = 0 To Slides.ListCount - 1

   If (Slides.Selected(i) = True) Then
      res = res + 1
      d.Add i, "True"
   Else:
      res = res - 1
      d.Add i, "False"
   End If

Next i
If res = Slides.ListCount Then
   Check1.Value = vbChecked
   Else:
   Check1.Value = vbUnchecked

   For i = Slides.ListIndex To Slides.ListCount - 1
      Slides.Selected(i) = d(i)
   Next i

   For i = 0 To Slides.ListIndex
      Slides.Selected(i) = d(i)
   Next i

End If
If Slides.ListIndex = 0 Then
   Slides.Selected(0) = d(0)
End If

End Sub


Private Sub btnStart_Click()
'EventHandler for the "Start" button
   ProgressBar.Value = 0

If (ppt_file.Text = "" Or outPut.Text = "") Then
   MsgBox ("You should select a PowerPoint file and output directory first!")
   Else:
   transform
End If
End Sub

Private Sub btnStop_Click()
'EventHandler for the "Stop" button
   ProgressBar.Value = 0
   Action.Caption = ""
   stop_transform = True
   btnStart.Enabled = True
   Set OutStream = Nothing

End Sub

Private Sub btnNext_Click()
'EventHandler for the "Next" button
   End

End Sub

Private Sub btnQuit_Click()
'EventHandler for the "Quit" button
   End

End Sub

Private Sub transform()

Dim name_pres As String
Dim xmlDoc As DOMDocument
Dim propertiesNode As IXMLDOMElement
Dim propertyNode As IXMLDOMElement
Dim Index, Index1, Index2, i, n, slide_count, x, y, X1, Y1 As Integer
Dim oPPTPres, oPPTPres_dummy As PowerPoint.Presentation
Dim FileFormat As PpSaveAsFileType
Dim propertyName, propertyValue, content, tect_effect, text_content, title_value, notes, _
content_textTable, content_notes, description, content_textEffect As String
Dim FSys As New FileSystemObject
Dim OutStream As TextStream

btnStop.Enabled = True

'Choose LO Type
If (CI.Value = True) Then
   item = "CI"
   ElseIf (PI.Value = True) Then
   item = "PI"
   Else: item = "AI"
End If

   stop_transform = False
   Action.Caption = "Please wait..."
   Action.ForeColor = &HFF&

'create a PowerPoint presentation
Set ppApp = CreateObject("PowerPoint.Application")

'open the PowerPoint presentation "behind the scene"
Set oPPTPres = ppApp.Presentations.Open(pptFile, WithWindow:=msoFalse)

   ProgressBar.Value = 0
   btnStart.Enabled = False

'if all slides are selected, then copy the open presentation 1 to 1 in the new dummy one
If (Slides.SelCount = oPPTPres.Slides.Count) Then
   Set oPPTPres_dummy = oPPTPres
   Else:

'if not all slides are selected, create a new dummy presentation
   Set oPPTPres_dummy = ppApp.Presentations.Add
   For f = Slides.ListCount - 1 To 0 Step -1

      If (Slides.Selected(f) = True) Then

'copy the selected slides in the new dummy presentation
         oPPTPres_dummy.Slides.InsertFromFile FileName:=pptFile, Index:=0, SlideStart:=f + 1, SlideEnd:=f + 1
      End If
   Next f
End If

slide_count_dummy = Slides.SelCount

'Creating an XML file
Set xmlDoc = Nothing
Set xmlDoc = CreateObject("Microsoft.XMLDOM")

'root element is <PowerPoint_document>
   xmlDoc.loadXML "<PowerPoint_document/>"
Set titelNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("titel"))
   strFilename = Mid$(oPPTPres.Name, 1, Len(oPPTPres.Name) - 4)
   titelNode.Text = CStr(strFilename)
Set propertiesNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("builtInProperties"))

'attaching the "BuiltInDocumentProperties" to the XML file
For Each G In oPPTPres.BuiltInDocumentProperties
     On Error Resume Next

      If Err.Number = 0 Then
         propertyName = G.Name
         propertyValue = _
         oPPTPres.BuiltInDocumentProperties.item(propertyName).Value
         propertyName = Replace(propertyName, " ", "_")
         Set propertyNode = propertiesNode.appendChild(xmlDoc.createElement(propertyName))
         propertyNode.Text = CStr(propertyValue)
```

```
        End If

      Err.Clear
Next

'attaching the "customProperties" to the XML file
Set propertiesNode = xmlDoc.documentElement.appendChild(xmlDoc.createElement("customProperties"))
For Each q In oPPTPres.CustomDocumentProperties
      On Error Resume Next

      If Err.Number = 0 Then
            propertyName = q.Name
            propertyValue = _
            oPPTPres.CustomDocumentProperties.item(propertyName).Value
            propertyName = Replace(propertyName, " ", "_")
            Set propertyNode = propertiesNode.appendChild(xmlDoc.createElement(propertyName))
            propertyNode.Text = CStr(propertyValue)
       End If
            Err.Clear
Next

For n = 1 To slide_count_dummy
    On Error Resume Next

'progress bar made visable
    Action.Caption = "Transforming the " & n & " of " & slide_count_dummy & " Slides"
    ProgressBar.Value = ProgressBar.Value + (Round(100 \ slide_count_dummy))
    ProgressBar.Visible = True
    content = ""
    textEffect = ""

'attaching the slide title to the XML
    Set Slide_1 = xmlDoc.documentElement.appendChild(xmlDoc.createElement("slide"))

    Set Slide_title = xmlDoc.documentElement.appendChild(xmlDoc.createElement("slide_title"))
        If oPPTPres.Slides(n).Shapes.HasTitle Then
            title_value = _
            oPPTPres_dummy.Slides(n).Shapes.Title.TextFrame.TextRange.Text
            Slide_title.Text = CStr(title_value)
            Else: title_value = n & " Slide"
            Slide_title.Text = CStr(title_value)
        End If

    Set Slide = xmlDoc.documentElement.appendChild(xmlDoc.createElement("slide_content"))

    Slide_1.appendChild Slide_title
    Slide_1.appendChild Slide

'attaching the content of the slides to the XML file
    For Index = 1 To oPPTPres.Slides(n).Shapes.Count
        On Error Resume Next

'extracting the content of the slides text elements
        text_content = ""
        text_content = oPPTPres_dummy.Slides(n).Shapes(Index).TextFrame.TextRange.Text

'extracting the content of the slides notes
        content_notes = ""
        content_notes = oPPTPres_dummy.Slides(n).NotesPage.Shapes(Index).TextFrame.TextRange.Text

'extracting the content of the slides textEffect elemnts
        content_textEffect = ""
        content_textEffect = oPPTPres_dummy.Slides(n).Shapes(Index).textEffect.Text

'attaching the content of the slides text elements into the xml file
        If (text_content <> "") Then
            Set Text = xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
            Set text_content_CDATA = xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(text_content))
            Text.appendChild text_content_CDATA
            Slide.appendChild Text
            content = content & text_content & vbCr

'attaching the content of the textEffect elemnts into the xml file
```

```
      ElseIf (content_textEffect <> "") Then
      Set Text = xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
      Set content_textEffect_CDATA = xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_textEffect))
      Text.appendChild content_textEffect_CDATA
      Slide.appendChild Text
      textEffect = textEffect & content_textEffect & vbCr

    End If


'extracting the content of the slides text when situated in a table
    For Index2 = 1 To oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows.Count
      For Index3 = 1 To oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows(Index2).cells.Count
        On Error Resume Next

        content_textTable = ""
        content_textTable = content_textTable & vbCr &
oPPTPres_dummy.Slides(n).Shapes(Index).Table.rows(Index2).cells(Index3).Shape.TextFrame.TextRange.Text

        If (content_textTable <> "") Then

'attaching the content of the slides text when situated in a table into the xml file
          Set Text = xmlDoc.documentElement.appendChild(xmlDoc.createElement("text"))
          Set text_content_CDATA = xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_textTable))
          Text.appendChild text_content_CDATA
          Slide.appendChild Text
          content_Table = content_Table & content_textTable & vbCr
        End If
      Next Index3
    Next Index2
  Next Index

'attaching the content of the slides notes into the xml file
  If (content_notes <> "") Then
    Set notes = xmlDoc.documentElement.appendChild(xmlDoc.createElement("notes"))
    Set notes_inhalt_CDATA = _
    xmlDoc.documentElement.appendChild(xmlDoc.createCDATASection(content_notes))
    notes.appendChild notes_inhalt_CDATA
    Slide.appendChild notes
  End If


  description = description & n & " Slide" & vbNewLine & content & textEffect & content_textTable & content_notes & vbNewLine
  title_value_short = title_value_short & n & " Slide - " & title_value & vbNewLine

  DoEvents: If stop_transform Then Exit Sub
  DoEvents: If stop_transform Then ProgressBar.Value = 0

Next n

'cleaning up the new line notation
For i = 1 To 31
  description = Replace(description, Chr(i), Chr(13))
Next i

description = Replace(description, """", "\""")

'max.3700 characters in description part of the LOV and LO
If (Len(description) > 3700) Then

'if more than 3700 characters than the description should list only the slides titles
  description = title_value_short

End If

'the .adl file name take the name of the presentation (if not extra changed by the user)
AdlFile = CommonDlg.FileName
Set OutStream = FSys.CreateTextFile(AdlFile, True, False)

'by default the name of the LO/LOV models will be the PPT presentation name (if the name of the LO /LOV is not exctra set by the user)
If (LO_LOV_name.Text <> "") Then
  LO_name_model = LO_LOV_name.Text
  LOV_name_model = LO_LOV_name.Text
```

```
    Else:
    LO_name_model = Left$(CommonDlg1.FileTitle, Len(CommonDlg1.FileTitle) - 4)
    LOV_name_model = Left$(CommonDlg1.FileTitle, Len(CommonDlg1.FileTitle) - 4)
End If

'seting the elements of the adl file
    learn_object_use_definition = vbNewLine & "VERSION <" + version + ">" _
    & vbNewLine & "BUSINESS PROCESS MODEL <" & "Temp" & " 1.0> : <" + lb_name + ">" _
    & vbNewLine & "VERSION <1.0>" & vbNewLine & "TYPE <Ebene 4 - Lernobjekte>" & vbNewLine

    learn_object_use = vbNewLine & "INSTANCE <" & LOV_name_model & "> : <LOV (eduWEAVER)>" & vbNewLine &
"ATTRIBUTE <Position>" _
    & vbNewLine & "VALUE ""NODE x:3cm y:2cm""" & vbNewLine & "ATTRIBUTE <Beschreibung>" _
    & vbNewLine & "VALUE """ & description & """" & vbNewLine & "ATTRIBUTE <Referenzobjekt>" _
    & vbNewLine & "VALUE ""REF mt:\""Lernobjektpool\"" m:\""" & LO_name_model & " 1.0" & "\"" c:\""LO (eduWEAVER)\"" i:\""" &
LO_name_model & "\""""" & vbNewLine

    pool_definition = "BUSINESS PROCESS MODEL <" & LO_name_model & " 1.0" & "> : <" + lb_name + ">" _
    & vbNewLine & "VERSION <1.0>" & vbNewLine & "TYPE <Lernobjektpool>" & vbNewLine

    learnobjectpool = learnobjectpool & vbNewLine & "INSTANCE <" & LO_name_model & "> : <LO (eduWEAVER)>" _
    & vbNewLine & "ATTRIBUTE <Position>" & "VALUE ""NODE x:3cm y:2cm""" _
    & vbNewLine & "ATTRIBUTE <Beschreibung>" & vbNewLine & "VALUE """ _
    & description & """" & vbNewLine & "ATTRIBUTE <Einstiegspunkt (" & item & " 1)>" _
    & vbNewLine & "VALUE ""ITEM \""<automatisch>\"" param:\""" _
    & Replace(Path, "\", "\\\\") & Mid$(CommonDlg1.FileTitle, 1, Len(CommonDlg1.FileTitle) - 4) _
    & ".html\""""" _
    & vbNewLine

'export of the dummy PPT presentation slides to HTML files
With ppApp.DefaultWebOptions
    .FrameColors = ppFrameColorsWhiteTextOnBlack
    .IncludeNavigation = True
    .ResizeGraphics = True
End With

With oPPTPres_dummy
    .WebOptions.ResizeGraphics = True

        With .PublishObjects(1)
            .FileName = Path & Mid$(CommonDlg1.FileTitle, 1, Len(CommonDlg1.FileTitle) - 4) & ".html"
            .SourceType = ppPublishAll
            .SpeakerNotes = True
            .HTMLVersion = ppHTMLDual
            .Publish
        End With
End With

Set PropertiesToXML = xmlDoc
    PropertiesToXML.Save Path & "meta_" & Mid$(CommonDlg1.FileTitle, 1, Len(CommonDlg1.FileTitle) - 4) & ".xml"

    btnStart.Enabled = True
    btnNext.Enabled = True

'closing the PPT presentations
    oPPTPres.Close
    oPPTPres_dummy.Close

'quit the PPT
    ppApp.Quit

'write the adl file
    OutStream.WriteLine vbNewLine & learn_object_use_definition & learn_object_use
    OutStream.WriteLine vbNewLine & pool_definition & learnobjectpool
    Set OutStream = Nothing
    ProgressBar.Value = 100
    Action.Caption = "Transform successfully finished."
    Action.ForeColor = &HC000&

End Sub
```

Module1.bas (the same in the both VB Projects)

```
Public Declare Function SendMessageLong Lib "user32" Alias "SendMessageA" _
(ByVal hwnd As Long, ByVal wMsg As Long, _
ByVal wParam As Long, ByVal lParam As Long) As Long
Public Const LB_SETSEL = &H185&
```