

E-Learning Standards

Critical and Practical Perspectives

Matthias Kerstner

E-Learning Standards

Critical and Practical Perspectives

Master's Thesis

at

Graz University of Technology

submitted by

Matthias Kerstner

Institute for Information Systems and Computer Media (IICM),
Graz University of Technology
A-8010 Graz, Austria

4th April 2011

Advisor: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic



E-Learning Standards

Critical and Practical Perspectives

Masterarbeit
an der
Technischen Universität Graz

vorgelegt von

Matthias Kerstner

Institut für Informationssysteme und Computer Medien (IICM),
Technische Universität Graz
A-8010 Graz

4. April 2011

Diese Arbeit ist in englischer Sprache verfasst.

Begutachter: Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helic



Abstract

Due to rapid technological advances in the past decade, e-learning has experienced substantial growth. Especially bigger, training intensive companies have recognized the potential of reusable electronic learning material.

As a consequence, a lot of effort has been spent on creating e-learning standards, that address every possible learning scenario imaginable. Unfortunately, it is exactly this objective that has caused them to evolve into complex structures, that are very costly to implement. In addition, after years of design, current prominent e-learning standards, such as SCORM, are still in a developmental stage. This reflects the endeavor to merge existing diverse business interests of the companies involved in the standardization process.

Moreover, the majority of current e-learning standards have been designed by technicians rather than educators and, according to critics, lack the pedagogical aspect of learning, being a collection of computer standards, rather than of learning standards per se. The implied conformity of current e-learning standards even created the “EduPunk” movement, which denounces the constrictive inherent generality of existing electronic learning standard approaches.

This thesis focuses on presenting e-learning in general, together with a selection of current prominent e-learning standards, such as SCORM and QTI, from a critical and practical perspective. Based on this theoretical background, this thesis presents a practical implementation of an e-learning platform called “Wörterwelt”, which has been developed with a focus on common web standards in contrast to strict adherence to existing de facto e-learning standards.

Kurzfassung

E-Learning hat auf Grund der rasant fortschreitenden technologischen Entwicklung seit dem letzten Jahrzehnt erheblich an Bedeutung gewonnen. Besonders größere, trainingsintensive Unternehmungen haben das Potential von wiederverwendbaren elektronischen Lernmaterialien erkannt.

Folglich wurde viel Aufwand zur Entwicklung von E-Learning Standards aufgebracht, um jedes erdenkliche Lernszenario abbilden zu können. Leider hat genau dieser Ansatz dazu geführt, dass die Standards zu hoch komplexen Strukturen gewachsen sind, die nur mit großem monetären Einsatz implementiert werden können. Zusätzlich befinden sich die derzeit prominenten Standards wie etwa SCORM trotz langjährigem Design noch immer in der Entwicklungsphase. Dies veranschaulicht nur zu deutlich das Unterfangen bestehende divergente Geschäftsinteressen, der in dem Standardisierungsprozess involvierten Firmen, zu vereinheitlichen.

Weiters prangern Kritiker den fehlenden pädagogischen Aspekt derzeitiger E-Learning Standards an, da diese ihrer Meinung nach überwiegend von Technikern statt Pädagogen entwickelt wurden. Folglich wären diese nur eine Sammlung von Computerstandards, im Gegensatz zu Lernstandards im eigentlichen Sinne. Basierend auf der in den Standards geforderten Konformität entstand auch die "EduPunk" Bewegung, die es sich zum Ziel gemacht hat, diese erzwungene Allgemeingültigkeit anzuprangern.

Diese Masterarbeit legt das Hauptaugenmerk auf eine kritische und praktische Betrachtung von E-Learning im Allgemeinen, sowie einer Auswahl an derzeit prominenten E-Learning Standards, wie etwa SCORM und QTI. Basierend auf dem theoretischen Hintergrund präsentiert diese Arbeit die praktische Umsetzung der E-Learning Plattform "Wörterwelt", bei dessen Entwicklung darauf geachtet, den Fokus speziell auf gängigen Webstandards und nicht ausschließlich auf de facto E-Learning Standards zu richten.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place

Date

Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort

Datum

Unterschrift

Contents

Contents	iii
List of Figures	vi
Acknowledgements	ix
Credits	xi
1 Introduction	1
2 E-Learning	3
2.1 History	3
2.2 Overview	4
2.2.1 Definition	5
2.2.2 Digital Learning Objects	6
2.2.2.1 Metadata	8
2.2.2.2 Learning Object Content Structures	8
2.3 Learning Management Systems	9
2.3.1 Moodle	10
2.3.1.1 Learning Centered Approach	11
2.3.1.2 Course Concept	12
2.3.1.3 Course Representation	12
2.3.1.4 Extensibility	13
3 E-Learning Standards	15
3.1 Learning Technology Standardization	15
3.1.1 Objectives	16
3.1.2 The Process	16
3.1.3 Related Approaches	17
3.2 Standards in detail	18
3.2.1 SCORM	18
3.2.1.1 Content Packages	19
3.2.1.2 Manifest	20
3.2.1.3 Package Interchange File	20
3.2.2 QTI	20
3.2.2.1 Assessment Test, Section, and Item Information Model	21
3.2.2.2 QTI Lite	22
3.3 Criticism	22
3.3.1 EduPunk	23
3.3.2 Diminishing Pedagogical Aspect of Learning	23
3.3.3 Conformity through Generality	24

4	Practical Implementation	25
4.1	Motivation	25
4.2	Dictionary Module	27
4.2.1	Goals and Tasks	27
4.2.2	Technology and Tools	28
4.2.2.1	ZK	30
4.2.2.2	Apache Tomcat	33
4.2.2.3	Java	33
4.2.2.4	MySQL	33
4.2.2.5	log4j	34
4.2.2.6	Apache Subversion	34
4.2.2.7	Alexik HTML TranslationExtractor	34
4.2.2.8	TextTools	37
4.2.2.9	phpMyAdmin	38
4.2.3	Implementation	38
4.2.3.1	Development Environment Setup	39
4.2.3.2	Requirements	40
4.2.3.3	User Interface	41
4.2.3.4	Data Schema	42
4.2.3.5	Database Schema	45
4.2.3.6	Architecture	46
4.2.3.7	Standards Used	50
4.3	Exercises Module	50
4.3.1	Goals and Tasks	51
4.3.2	Technology and Tools	52
4.3.2.1	Dojo Toolkit	52
4.3.2.2	PHP	56
4.3.2.3	JSON	56
4.3.3	Implementation	57
4.3.3.1	Development Environment Setup	57
4.3.3.2	Requirements	57
4.3.3.3	Data Schema	59
4.3.3.4	User Interface	60
4.3.3.5	Architecture	61
4.3.3.6	Standards Used	62
5	Feedback	65
5.1	Formal Experiment	65
5.1.1	Test Procedure	66
5.1.2	Test Users	66
5.1.3	Test Environment	67
5.1.4	Training	67
5.1.5	Tasks	67
5.1.6	Feedback Questionnaire	68
5.1.7	Final Interview	68
5.2	Lessons Learned	69
5.2.1	Interactive Help	69
5.2.2	Progress Information	70

6 Outlook	73
6.1 General Trends	73
6.2 Related Work	74
6.3 Ideas for Future Work	75
7 Concluding Remarks	77
A Feedback Questionnaire	79
B TextTools	81
B.1 Statistics	82
B.2 Extraction	82
B.3 Rotation	82
B.4 Replacing	82
B.4.1 Replacement Rules	82
B.5 Sorting	83
B.6 Removal of Duplicates	83
B.7 Output	83
B.8 Default Filenames	83
C Exercise Template	85
Bibliography	93
Glossary	95

List of Figures

2.1	Overview of Fields of Thought and Practice involved in E-Learning	5
2.2	The E-Learning Process Lifecycle	6
2.3	Content Object Hierarchy for Learning Objects	7
2.4	Example of a SCORM Manifest File	8
2.5	Example of Curricular Taxonomies	9
2.6	Functionality Overview of a Learning Management System	10
2.7	Moodle Representation of SCORM Course Format	11
2.8	Moodle Representation of SCORM Course Format for Quizzes	13
3.1	Collaborative Development Model for Formal Learning Standards	17
3.2	SCORM Content Hierarchy	18
3.3	SCORM Content Package	20
3.4	QTI Assessment Model	21
3.5	Moodle Representation of QTI Test from Listing 3.1	22
4.1	E-Learning Platform Wörterwelt Architecture Overview	27
4.2	ZK Application as a Collection of ZUL Pages	31
4.3	ZK Architecture	32
4.4	ZK Flow of events	33
4.5	Connection between server side components and client side widgets	34
4.6	Translation Data Processing Pipeline	34
4.7	Alexik HTML Translation Extractor Processing Pipeline	35
4.8	Translation Data Transformation Process	35
4.9	Folder Structure used by ALEXIK HTML Translation Extractor	37
4.10	Excerpt from a German-Czech Microsoft Word Translation Document	38
4.11	TextTools Processing Pipeline	38
4.12	Unified Data Administration Backend Tool <i>phpMyAdmin</i>	39
4.13	Dictionary Module User Interface No Results Message	41
4.14	Dictionary Module User Interface Functionality Overview	42
4.15	Dictionary Module Database Schema	45
4.16	Dictionary Module Initial Page Load	47
4.17	Dictionary Module Database Abstraction Layer	47
4.18	Dictionary Module Business Logic Layer Architecture	48
4.19	Dictionary Module Event Handling	49

4.20	Dojo Toolkit Architecture	53
4.21	Exercise Templates Folder Structure	58
4.22	Exercise Module Course Structure	58
4.23	Exercise Module Course Selection User Interface	61
4.24	Exercise Module Exercise User Interface	62
4.25	Exercises Module Architecture	63
5.1	Interactive Help System of the Exercises Module	67
5.2	Interactive Help System of the Exercises Module highlighting answers	68
5.3	Interactive Help System Showing Incomplete Results	69
5.4	Interactive Help System Showing Complete Results	70
5.5	Interactive Help System for the Exercises Module	70
6.1	SprichWort-Plattform Welcome Page	74
6.2	Example of SprichWort-Plattform's cloze text exercises	75
6.3	SprichWort-Plattform Feedback Messages	75
A.1	Feedback Survey Page 1	79
A.2	Feedback Survey Page 2	80

Listings

3.1	QTI Sample Test Question	22
4.1	Naming Schema of Microsoft Word Translation Documents	35
4.2	Outcome of Translation Data Transformation Process	36
4.3	Dictionary Data Syntax Rule Schema	42
4.4	Example using the → symbol	43
4.5	Example using the symbol	43
4.6	Example using the symbol where plural matches singular form	43
4.7	Example using the symbol where plural form partially differs from singular form	43
4.8	Example using the symbol where plural differs singular form	43
4.9	Example using the symbol where no plural form exists	43
4.10	Example using the () symbol	44
4.11	Example using the : symbol	44
4.12	Example using the :=⇐: symbol	44
4.13	Example using the bold Formatting Rule	44
4.14	Example using the <u>underline</u> Formatting Rule	44
4.15	Example using the <i>italics</i> Formatting Rule	44
4.16	Example of a Formatted Dictionary Entry	45
4.17	Keyword Word Relation	46
4.18	ZK Default Document Type Definition	50
4.19	Example for including Dijits using the <i>dojoType</i> Directive	54
4.20	Example of JSON Notation [Crockford [2006]]	56
4.21	Exercise JSON Data Schema	59
4.22	Exercise Template Filename Convention	60
B.1	Example TextTools Call	81
B.2	TextTools Help	81
B.3	TextTools Rules Format	82
B.4	TextTools Rules Environment Format	82
C.1	Wörterwelt Drag&Drop Exercise Template Example	85

Acknowledgements

This thesis has been written with the help and support of people, to whom I would like to express my gratitude and appreciation.

First and foremost, I would like to express my gratitude to Prof. Denis Helic. He has proven to be an excellent mentor throughout my entire master degree study. As a master thesis advisor and assessor, he has always been very helpful, patient and supportive.

Furthermore, I am deeply indebted to Prof. Rudolf Muhr for offering me the chance to be a part of the ALEXIK project team. It is always a pleasure to work with him.

Also, I would also like to thank Prof. Keith Andrews for his wonderful \LaTeX skeleton that provided a very solid foundation for writing this thesis.

Finally, I would like to thank my dear friends and especially my girlfriend, who always helped me with whatever problems I ran into. In particular, I would like to express my gratitude to Florian Klien for providing me with a GIT repository for this thesis. It is a pleasure to have such great people around one.

I will always look back grateful at my study time, thanks to the previously mentioned people.

Matthias Kerstner
Graz, Austria, April 2011

Credits

First and foremost, credits go to my mentor Prof. Denis Helic, who provided me with support and ideas for this master thesis.

Moreover, this thesis has been written using Prof. Keith Andrews' wonderful skeleton thesis [Andrews \[2010\]](#), which proved to be a more than pleasant choice in the endeavour to control L^AT_EX.

Finally, many scientific articles used for this thesis have been provided by digital libraries, such as the “Association for Computer Machinery” (ACM) and the “Institute of Electrical and Electronics Engineers” (IEEE). Without these libraries the search for appropriate literature would have been a very difficult and time consuming task. Thus, I would also like to thank the Graz Technical University for providing full access to the previously mentioned libraries.

Chapter 1

Introduction

“To learn while being entertained is always an effective means in education.”

[Hui et al. [2007]]

This thesis describes a critical and practical approach to current prominent e-learning standards and technologies.

In the past decade, empowered by the rapid advances in the communication technology sector, electronic learning (*e-learning*) has become a viable alternative to traditional, face-to-face teaching methodologies. Large, training-intensive companies in particular have realized the inherent potential of reusable learning material in a “write-once-use-often” approach.

Nevertheless, despite initial enthusiasm for e-learning standards, which focus on combining the diverse business interests of companies involved, to ultimately achieve globally valid specifications, critics now denunciate the missing pedagogical aspects, as well as the conformity forced on developers. Moreover, in an attempt to address every possible learning scenario, e-learning standards have evolved into complex structures that are very costly and time consuming to implement.

This thesis is structured as follows: Chapter 2 introduces the reader to the topic of e-learning. Section 2.1 begins with an historical background and an overview of the primary organizations involved. Section 2.2 defines e-learning in a broader sense and provides a list of prominent definitions. In this, Section 2.2.2 presents the core principle of e-learning, reusable digital learning objects. Based on this background information, metadata and learning object content structures will be explained in section 2.2.2.1 and 2.2.2.2 respectively. Chapter 2 concludes with an overview of the functionality provided by so-called Learning Management Systems in Section 2.3 and presents Moodle as an example in section 2.3.1.

Chapter 3 introduces the reader into current e-learning standards based on the overview of e-learning core concepts. In order to illustrate the complexity involved in defining globally valid specifications, Section 3.1 discusses the learning technology standardization process. Afterwards, Section 3.2 then presents current prominent e-learning standards. Section 3.2.1 presents SCORM, Section 3.2.2 covers the QTI standard, which was specifically designed for assessment environments. This chapter concludes with criticism concerning current e-learning standards in section 3.3, by illustrating inherent problems.

Chapter 4 presents the practical implementation of an e-learning platform called *Wörterwelt*, which consists of two main components: a dictionary and an exercise module. Section 4.1 provides the reader first with the motivational background for this implementation. Section 4.2 covers the dictionary module, and Section 4.3 presents the exercise module in greater detail, using the same structure for both. In these sections the the goals and tasks are first identified, and then the required tools and technologies are described. Following this technological overview, the second part of these sections covers a more detailed presentation of the implementation, starting with a description of the development environment

selected, followed by the requirement specifications, the user interface, the data schema and architecture used and finally, the standards implemented.

Chapter 5 presents the feedback gathered in the formal experiment, conducted between the two development phases, in order to detect usability issues with the e-learning platform. First of all, Section 5.1 presents an overview of the test design including the test procedure, test users, test environment, training, tasks, the feedback questionnaire and the final interview. Concluding this chapter, Section 5.2 presents the results and the lessons learned to improve the e-learning platform.

The final Chapter 6 discusses general trends in the e-learning and standardization sector, presents related work done in the form of *SprichWort-Plattform* in section 6.2. Section 6.3 outlines some ideas for future work and research.

Chapter 2

E-Learning

“Knowledge acquisition is no longer mainly restricted to classical institutions and formal learning (as in schools and universities) but is also connected to informal learning settings at home in leisure time or at the workplace.”

[Hesse [2009]]

Due to the technological advances in the field of information-, network- and multimedia technology in the last decade, electronic learning (*e-learning*) has become a buzzword for a new trend in teaching methodology, that “breaks the limitation of traditional teaching model in space-time” Yu and Fan [2009]. What started as an electronic alternative to traditional face-to-face teaching methods in the aviation industry in the early 1980s Fallon et al. [2002], has become a flourishing market the the last couple of years with a wide range of vendors and platforms, and is predicted to grow significantly in the next years Eklund et al. [2003].

This chapter serves as an introduction to the topic of e-learning. Whereas section 2.1 presents an historical background and evolution of e-learning, Section 2.2 focuses on the specificities of e-learning, such as *digital learning objects* (Section 2.2.2) and *learning object content structures* (Section 2.2.2.2). Section 2.3 concludes this chapter by presenting the concept of *learning management systems* and discussing an exemplary open source learning platform called *Moodle* (Section 2.3.1).

2.1 History

The concept of e-learning has its origins in the early 1980s when the aviation industry realized that it is vital to train personnel with the most current information, in order to maintain the highest safety levels Fallon et al. [2002]. They determined computer-based training (CBT) as the best option to deliver flexible, accurate and media-rich content in a up-to-date fashion.

This idea caught on quickly and the aviation industry spent millions of dollars to create CBT materials over the next years. With the increasing volume of CBT content and variety of vendors, compatibility issues emerged. At that time, CBT were not only software specific, but were locked to certain hardware configurations Fallon et al. [2002], forcing manufactors to supply specific hard- and software configurations eventually leading to enormous investment and maintenance costs.

In answer to this, the *Aviation Industry CBT Committee* (AICC) was founded and in 1993 produced the *AICC CMI specification* (AGR¹-006, AICC CMI Subcommittee [2004]) that defined standards for sharing data across computer-managed instruction (CMI) systems from different vendors Fallon et al.

¹AICC Guidelines and Recommendations

[2002]. As a result, CMI systems basically represent the predecessor of today's learning management systems, as described in section 2.3.

The technological advances, and more importantly the standardization of HTML by the *World Wide Web Consortium* (W3C, W3C [1999]), started a revolution in the e-learning sector Jones [2002]. Accordingly, the AICC CMI specification was later updated to include support for web-based CBT (WBT), resulting in the web-based CMI guidelines (AGR-010) AICC CMI Subcommittee [1998]. In this, AGRs represent the official documentation overview papers for particular areas of interest. They are linked to the actual technical specifications in the form of technical reports and whitepapers, each of which are identified by distinct prefixes, such as CMI, to mark the corresponding subcommittee AICC [2010]. Due to its widespread acceptance, the AICC CMI specification became the first industry standard for e-learning Fallon et al. [2002].

Despite AICC's efforts to standardize learning content, based on the White House Office of Science and Technology Policy Sonwalkar [2002a], the U.S. Department of Defense (DoD) in 1997 founded the *Advanced Distributed Learning Initiative (ADL)* Jones [2002]. Its main objective was to modernize the delivery of training materials to their forces Fallon et al. [2002]. The resulting e-learning *specification*, called *Sharable Content Object Reference Model (SCORM)*, was published in 2000 and has undergone multiple versions since Rustici [2009a]. The SCORM specification will be presented in greater detail in Chapter 3.

At about the same time, three additional key organizations became involved in the process of standardizing e-learning: the *IMS Global Learning Consortium (IMS²)*, the *Alliance of Remote Instructional Authoring Distribution Networks for Europe (ARIADNE)* and the *IEEE's Learning Technology Standards Committee (IEEE LTSC³)*. The IMS produces open specifications for several aspects of e-learning Kanendran et al. [2005], primarily in the sector of metadata Sonwalkar [2002a], such as the IMS Learning Resources Meta-data Specification (LRMDS) IMS [2011]. ARIADNE is Europe's counterpart of IMS and also focuses on producing specifications for metadata and reusability Kanendran et al. [2005].

The IEEE LTSC on the other hand represents the designated body to create *accredited standards* through independent evaluation of draft specifications submitted by organizations like the IMS or AICC Kanendran et al. [2005], which will be described in greater detail in section 3.1. Thus, the IEEE LTSC paved the way to convert specifications developed by a multitude of organizations to e-learning standards Fallon et al. [2002]. Ultimately, it is expected that the majority of standards developed by the IEEE LTSC will be submitted to the International Organization of Standardization (ISO) for formal internationalization to serve as international standards (Sonwalkar [2002a], Kanendran et al. [2005]). Common e-learning standards and the standardization process itself will be discussed in greater detail in Chapter 3.

2.2 Overview

Based on the historical background of e-learning presented in the previous section, this section focuses on providing the reader with a general overview of e-learning's underlying concepts and specificities.

With regard to the immense technological advances, as well as standardization of web technologies such as HTML in recent years (Bowles [2004], Jones [2002]), traditional teaching methodologies have been extended by their electronic counterpart: E-Learning. The ongoing growth in the e-learning sector is directly related to the increasing access to information and communication technology, as well as its simultaneously decreasing costs Naidu [2006]. Furthermore, it is driven by the expectations of users growing up with these technologies and using them productively and with pleasure Eklund et al. [2003]. Schroeder [2009] even argues, that the influence of Web 2.0, as well as social software together with the

²formerly EDUCAUSE

³<http://ltsc.ieee.org>

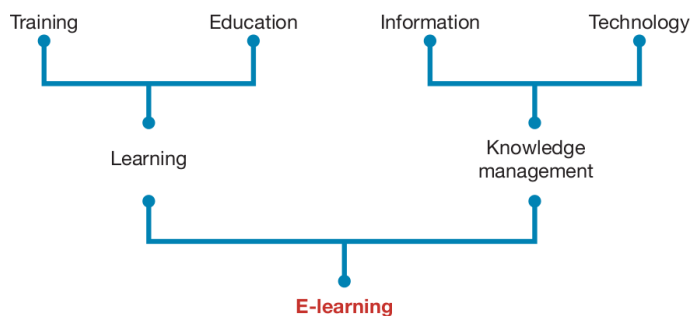


Figure 2.1: Overview of Fields of Thought and Practice involved in E-Learning [Bowles [2004]]

rapidly evolving mobile technology sector represents the driving force behind e-learning.

E-learning not only breaks the limitations of traditional teaching models Yu and Fan [2009], but also provides further enhancements by incorporating a multitude of *assets*, such as images and videos, to produce media-rich, flexible, interactive courses. According to Naidu [2006], these *virtual learning environments* motivate learners with clever use of multimedia components to capture and visualize real-world scenarios. Moreover, they enable the concept of *distant learning* by freeing learners from the constraints of residential educational settings through flexible, 24/7 available learning material Bowles [2004].

One of the main concepts behind e-learning is the idea of *reusing* existing learning content. Once created, it can be reused and *restructured* arbitrarily often, ideally even in different contexts, which naturally concurs with the objective to optimize low costs Naidu [2006]. In contrast, according to Bowles [2004], the effort of organising traditional face-to-face classroom training can account for as much as 40 per cent of corporate training budgets. Jones [2002] on the other hand states, that the initial costs for developing e-learning courses can be amortized over several years. Ultimately, learning content also conforms to common standards to enable the exchange across different e-learning systems. Due to these numerous advantages and flexibility, e-learning has been adopted worldwide by training and educational organizations worldwide Fallon et al. [2002].

As shown in Figure 2.1, e-learning can be described as a combination of multiple fields of thought and practice. According to Bowles [2004], e-learning is comprised of *learning* and *knowledge management*, which in return can be further distinguished by *training* and *education* on the one hand and *information* and *technology* on the other. Whereas the knowledge management component focuses on the technical aspects, learning covers the pedagogical background. Numerous papers exist that deal with the pedagogical aspects of e-learning (Yu and Fan [2009], Leacock et al. [2010], Naidu [2006]). This thesis focuses on the technical aspects of e-learning.

2.2.1 Definition

Based on the previous overview, this section is dedicated to provide a more refined definition of e-learning. In the literature, there are many different approaches to define e-learning, especially with regard to the aforementioned fields of thought and practice involved. Whereas some definitions are more precise, particularly concerning the technologies used, others merely scratch the surface by providing a general idea of its meaning. The author has picked a distinct collection of definitions which in his opinion best reflect the concepts behind e-learning.

The most generic definition of e-learning found by the author is by Bowles [2004], who states that e-learning “encompasses any type of learning content that is delivered electronically”. This concurs with the definition of the American Society for Trainers and Development (ASTD) that defines e-learning as “instructional content or learning experiences delivered or enabled by electronic technology” IsoDy-

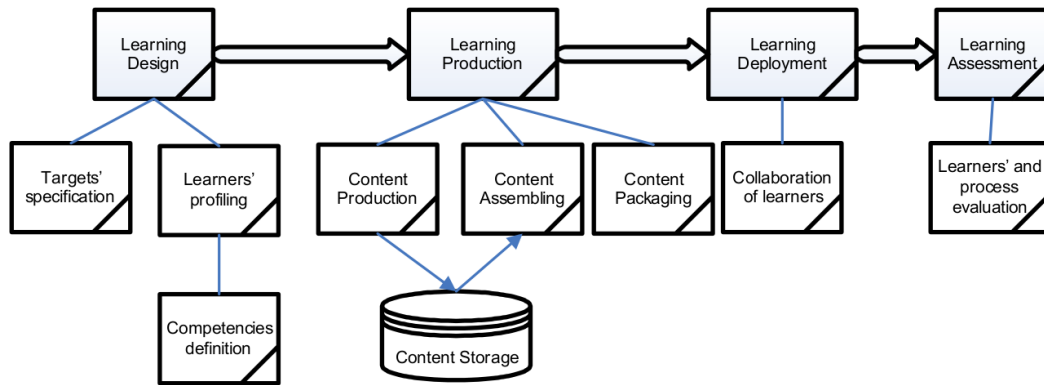


Figure 2.2: The E-Learning Process Lifecycle [Varlamis et al. [2006]]

dynamic [2001]. Yu and Fan [2009] refine these definitions by stating that e-learning is the set of "whole activities of teaching and learning based on computer management environments constructed from network information techniques with interactive communications". From a more technical perspective, Stojanovic et al. [2001] defines it as a "distributed, student-oriented, personalized and non-linear/dynamic learning process that aims to provide on-demand, task relevant educational material". Probably the best definition is given by Fallon et al. [2002], stating that e-learning is "any learning, training or education that is facilitated by the use of well-known and proven computer technologies, specifically networks based on Internet technology". This definition includes the key ingredients for the growing success of e-learning in the last couple of years: the advances in network and communication technologies.

According to Varlamis et al. [2006], there are four main phases in the e-learning process ranging from defining the targets and requirements (*design*), through generating and packaging learning materials (*production*) and distributing it (*deployment*), to the final *assessment* of learners and the process itself, as depicted in Figure 2.2. The feedback gathered in the assessment phase can then be fed back into the process for the next iteration.

The e-learning process addresses the issues of interoperability and standardization of tasks involved Varlamis et al. [2006]. First and foremost, it shows the strong, mutual influence between the tasks. For instance, the definition of competencies to be covered by the resulting learning materials heavily depends on the features of a learner's profile. Secondly, the *semantic interoperability* of tasks involved also plays an integral part, which promotes mutual understanding of learning goals to be achieved between teachers and learners through standardized concepts Varlamis et al. [2006]. Finally, in order to prevent the fragmentation of incompatible technologies and to promote the ability to distribute interoperable learning material, system vendors must conform to common standards.

Thus, the core objective of e-learning is to create and maintain interoperable learning material, represented by so-called *digital learning objects*. Section 2.2.2 focuses on digital learning objects, which are the building blocks of e-learning.

2.2.2 Digital Learning Objects

In e-learning itself, there exists a wide range of definitions for *digital learning objects* (LO). For instance, Naidu [2006] generically refers to LOs as electronic entities that "have the potential to promote learning", and due to their discrete nature, can be managed independently. Varlamis et al. [2006] define LOs as "digital parts of courses, that range in size and complexity ranging from single graphics to entire courses themselves". Finally, Fallon et al. [2002] refers to LOs as "the smallest chunk of content that can stand by itself as a meaningful unit of learning".

Although these definitions might suggest that LOs must be quite compact in size and functionality,

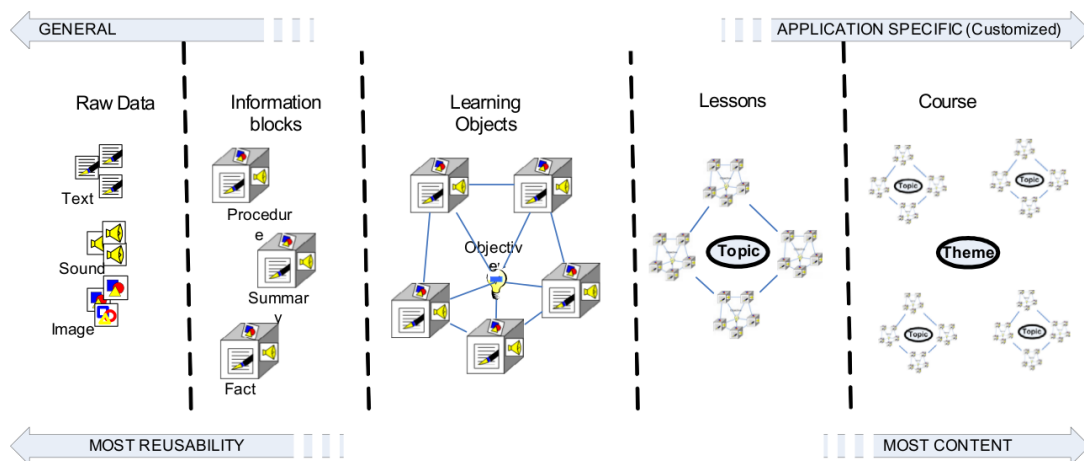


Figure 2.3: Content Object Hierarchy for Learning Objects [Varlamis et al. [2006]]

their authors can determine their actual complexity. Nevertheless, independent of their size, LOs are the smallest learning units available in e-learning that can be separately addressed, authored and delivered (Fallon et al. [2002], Varlamis et al. [2006]). In the author's opinion, LOs can be best compared to *bytes* in common computer architectures. Although bytes are made up of even smaller units, the *bits*, they are the smallest directly addressable memory units. The granularity of LOs will be discussed shortly.

Their name is derived from the object oriented programming paradigm Naidu [2006], where content and functionality is encapsulated into single *object* entities. Hence, they can be regarded as the *building blocks* of e-learning, since without them learning material could not be electronically represented. This makes LOs the key aspect of e-learning systems, which promote reusability, interoperability and adaptability Varlamis et al. [2006].

In order to be reusable, LOs must be *self-contained* and *independent of context* Fallon et al. [2002]. This leads to certain limitations when designing courses and LOs, especially when it comes to defining *course sequences*. To retain their discrete nature, it is considered best practice for single LOs to only map onto only specific learning objectives or concepts Fallon et al. [2002]. This enables systems to assemble courses automatically or "on-the-fly", based on specific contexts Varlamis et al. [2006].

As shown in Figure 2.3, LOs are based on hierarchical representations of granular content Varlamis et al. [2006], which can be separated into five main levels. Although LOs are the smallest meaningful learning units in e-learning environments, they are comprised of even smaller entities. *Raw data*, such as plaintext or images represent the most granular, reusable and flexible items in this hierarchy. Since the average information content of raw data is rather limited, they are combined to form *assets*, or *information blocks*, which in return serve as foundation for assembling learning objects. Consequently, the higher up in the content object hierarchy, the richer and more specific the information content becomes, while simultaneously losing flexibility and universality. As a result, LOs are the link between loose collections of raw assets and desired learning courses.

While adding *context* to the information blocks, LOs still provide reusability and flexibility, as assets can be easily exchanged, for instance by using *Learning Object Authoring* (LOA) tools. But most important, they enable interoperability when designed according to common standards Varlamis et al. [2006]. Consequently, their discrete nature eliminates the discrepancy between reusability and context Varlamis et al. [2006], by combining the benefits of both sides of the content object hierarchy shown in Figure 2.3.

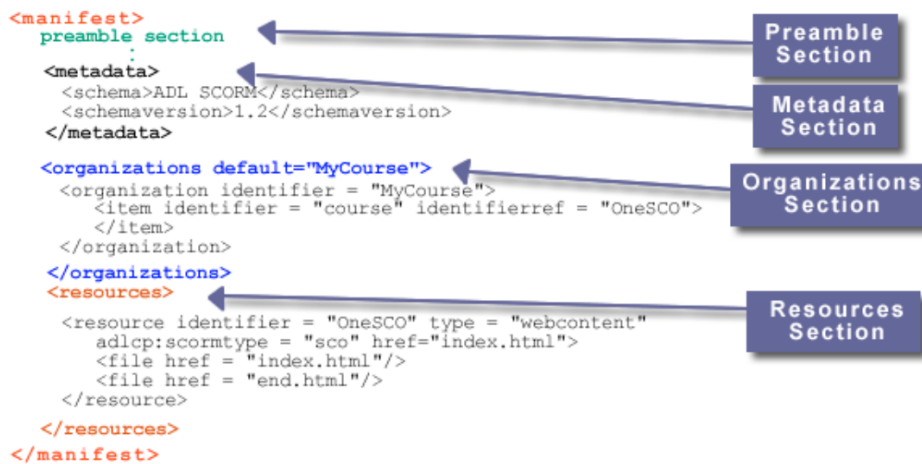


Figure 2.4: Example of a SCORM Manifest File [Jones [2002]]

2.2.2.1 Metadata

Based on the structure of learning objects presented in the previous section, the next step is to define how LOs can be identified, located and organized. In order to be able to easily retrieve LOs, they have to be uniformly and systematically described with learning object *metadata* Naidu [2006]. Thus, with interoperability and reusability in mind, metadata must conform to certain standards. In the past decade, several organizations and initiatives have been founded to establish common metadata standards.

One of the most prominent standards is the *Learning Object Metadata* (LOM) standard created by a cooperation between the IMS and the IEEE LTSC Sonwalkar [2002a]. It is comprised of nine hierarchical categories: *General*, *Lifecycle*, *Meta-Metadata*, *Technical*, *Educational*, *Rights*, *Relation*, *Annotation* and *Classification* and has been approved by the IEEE as an accredited standard IEEE [2002].

Metadata defines several key aspects of LOs, such as its contents, objectives, authors and targeted audiences Fallon et al. [2002]. From a practical perspective they resemble ordinary library catalogue cards containing information about their resources in a consistent format Naidu [2006]. Figure 2.4 illustrates an exemplary metadata section contained in a SCORM manifest file, which will be presented in greater detail in Chapter 3. Returning to the library analogy, so-called *Learning Object Repositories* (LOR) represent the electronic libraries for storing LOs. Based on the metadata provided, LOs can then be easily located, shared and reused Naidu [2006].

It is important to note that metadata definitions are not directly contained in the LOs, but rather are attached in the form of separate descriptive files Fallon et al. [2002]. Using this approach, metadata information can be examined without being forced to open the entire LO. Since single learning objects do not often contain sufficient material to cover an entire learning area, it is of great interest to form *meta structures* of LO. These *learning object content structures* will be discussed in section 2.2.2.2.

2.2.2.2 Learning Object Content Structures

As depicted in Figure 2.3, learning objects can be combined to form larger hierarchical *content structures*, such as *lessons* and *courses*. In order to be able to represent a broad range of possible content structures, simple, yet flexible mechanisms and must be defined.

Figure 2.5 shows an exemplary curricular taxonomy, representing a defined set of named hierarchical learning levels Fallon et al. [2002]. Although all taxonomies result in courses, they differ in their granularity. For instance, whereas the “Army” and “Air Force” taxonomies only distinguish between learning objectives and lessons, the “Marine Corps” taxonomy uses the additional “Task” level.

Army	Air Force	Marine Corps	Canadian
Course	Course	Course	Course
Module	Block	Phase	Performance objective
Lesson	Module	SubCourse (annex)	Enabling objective
Learning objective	Lesson	Lesson	Teaching point
Learning step	Learning objective	Task	
		Learning objective	
		Learning step	

Figure 2.5: Example of Curricular Taxonomies [reproduced from Fallon et al. [2002]]

To achieve reusable and interoperable content structures, the work of various standards groups on expandable content hierarchy models has resulted in two prominent e-learning standards:

- *SCORM Content Hierarchy*
- *AICC Content Hierarchy*

Both standards incorporate three main components. Whereas the SCORM Content Hierarchy includes *content aggregations*, *shareable content objects* (SCO) and *assets*, the AICC Content Hierarchy consists of *courses*, *instructional blocks* and *assignable units* (AU) Fallon et al. [2002]. Hereby, SCOs and AUs represent the digital learning objects in the respective models.

Thus, when speaking of standard conformant e-learning content, two key definitions are available: *AICC-* and *SCORM-conformant* learning content Fallon et al. [2002]. The SCORM standard will be discussed in greater detail in section 3.2.1. AICC specifications are released as so-called *guidelines* Fallon et al. [2002], such as the most widely known *CMI001 AICC/CMI Guidelines for Interoperability* AICC CMI Subcommittee [2004]. The interested reader is encouraged to consult the official documentation for further information on the AICC Content Hierarchy Standard available on <http://www.aicc.org>.

In order to manage learning materials and to track users' progress efficiently, special tools are needed which will be presented in section 2.3.

2.3 Learning Management Systems

As of today there exist more than a hundred different *Learning Management Systems* (LMS) on the e-learning market Cantoni et al. [2004]. Some of the most prominent ones are WebCT, Moodle, Blackboard, Lotus Learning Space and FirstClass Naidu [2006]. Prior to presenting Moodle as an exemplary LMS in section 2.3.1, an overview of the functionality provided by LMS will be given.

According to Naidu [2006], LMS are tool suites that generally share the following set of features:

- course content delivery capabilities
- management of online class transactions
- tracking and reporting of learner progress
- assessment of learning outcomes
- reporting of achievement and completion of learning tasks
- management of student records

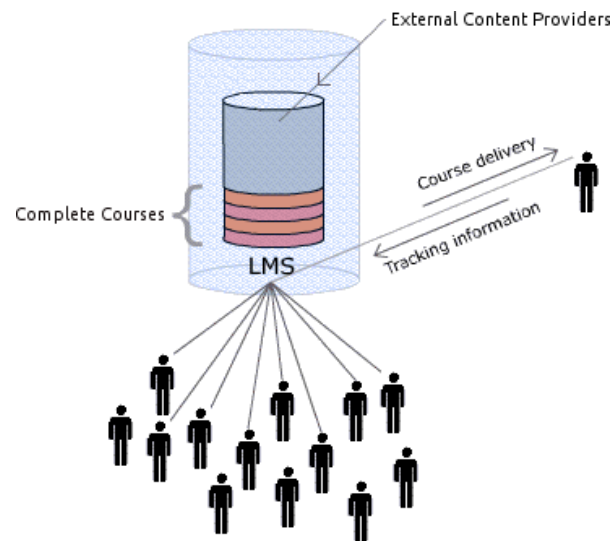


Figure 2.6: Functionality Overview of a Learning Management System [reproduced from Nichani [2001]]

Their main objective is to provide standardized and flexible access to resources and services needed to engage users in learning courses. In contrast to *Learning Content Management Systems* (LCMS), such as Macromedia's Dreamweaver⁴ Nichani [2001], LMSs generally do not provide means to create learning material and are more concerned with capturing learning activities to measure and manage learning progress.

Thus, according to Bowles [2004], LMSs can be best described as administrative tools that aim to simplify the management of learners' enrolment and registration, tracking the learner's overall progress and recording it for assessments. In addition, LMSs also *sequence* learning material in the course of content delivery Fallon et al. [2002]. Thus, LOs contained in courses might be delivered based on certain sequencing rules. Unfortunately, the support and implementation of the sequencing functionality heavily depends on the LMS deployed Fallon et al. [2002].

Figure 2.6 depicts a typical LMS configuration. It shows that LMSs provide APIs and services for delivering courses to learners while tracking their progress. As can be seen, content created by external tools or content providers can be added to the LMS. Hereby, courses are the smallest self-contained entities in LMSs Nichani [2001], as previously described in section 2.2.2.

Consequently, the most important concept behind LMSs is that they provide means to *reuse* these courses among an arbitrary number of users. In order to give the reader a better view of the functionality provided by LMSs, the following section presents one of the best known open source LMSs available today: Moodle.

2.3.1 Moodle

"Social constructionism is based on the idea that people learn best when they are engaged in a social process of constructing knowledge through the act of constructing an artifact for others"

[Cole and Foster [2008]]

The *Modular Object-Oriented Dynamic Learning Environment* (Moodle) is an example of a so-called Learning Content Management System, a hybrid between LMSs and Content Management Systems

⁴<http://www.dreamweaver.com>

Figure 2.7: Moodle Representation of SCORM Course Format

Nichani [2001]. Cole and Foster [2008] calls it a *Course Management System (CMS)* for short. Whereas LMS in their pure form generally do not provide means for creating content, LCMS combine the benefits from both worlds.

According to Cole and Foster [2008], LCMSs' core features can be summed up in five main groups. First, learners and teachers alike are able to upload and *share material*. Moreover, LCMSs provide users with *forums and chats*, which serve as the primary place for social interaction, informal announcements and open discussions on knowledge acquired. Learners are also able to *run quizzes* and teachers can *gather and review assignments*. Furthermore, LCMSs incorporate flexible grading schemas for *grading and keeping records*.

2.3.1.1 Learning Centered Approach

What distinguishes Moodle from other CMSs is its *learning-centered* approach, as compared to tool-centered CMSs Cole and Foster [2008]. The key concept behind Moodle is the idea of *social constructivism* to provide means for representing the social process of *constructing knowledge* by incorporating freshly acquired knowledge into existing know-how Moodle [2010a]. This idea concurs with Bowles [2004], who states that like any learning process, e-learning heavily depends on effective communication of knowledge.

During the knowledge construction process, learners negotiate the meaning of shared artifacts and symbols, to find common understandings Cole and Foster [2008]. Consequently, Moodle's main objective is to deliver tools for discussing and sharing these artifacts, in order to engage and support learners in this process.

This approach is based on Moodle's *five key principles* Moodle [2010b]. Firstly, every participant is simultaneously a potential learner and teacher. Secondly, when creating and expressing content for others, learning occurs. Thirdly, we also learn by observing others, that is, others are able to transform our behavior. Finally, learning occurs best in a flexible and adaptable learning environment.

2.3.1.2 Course Concept

Moodle's primary data concept is based on *courses*. As of this writing, Moodle supports the following course formats:

- *SCORM Format*
- *Learning Activity Management System (LAMS⁵) Format*
- *Social Format*
- *Topics Format*
- *Weekly Format*

Whereas the SCORM Format displays SCORM/AICC-conformant packages described in Section 2.2.2.2) on the courses' start page, the LAMS Format provides means to visualize learning material generated using the Learning Activity Management System. The Social Format presents users with the main discussion forum for the respective courses, the Topics Format displays the courses' *topics*, and in the Weekly Format users are shown the courses' *sections*. Thus, whereas the Topics Format is suitable for concept-oriented courses Cole and Foster [2008], the Weekly Format is more likely to be used when learners are required to meet certain deadlines for specific learning sections, as it provides a compact visualization of the course's underlying schedule.

Moreover, it is important to note that SCORM packages can either be imported to represent entire courses, as shown in Figure 2.7, or created as so-called *activities*, which will be described in a moment.

2.3.1.3 Course Representation

Figure 2.7 depicts an exemplary SCORM/AICC course representation using the sample SCORM Golf-packages available via ADL's homepage⁶.

Hereby, the left hand side of the course page displays the learning sections: *Playing the Game*, *Etiquette*, *Handicapping* and *Having Fun*. Each section, or *lesson*, ends with a *quiz* to check acquired knowledge. On the right hand side the corresponding course contents are displayed. Users either can browse through the course using the lessons displayed on the left or by using the navigation bar shown at the bottom.

Furthermore, the exemplary SCORM course ends each learning section with a quiz, as shown in Figure 2.8. The functionality provided in the quiz view matches the learning section pages. In addition, users in this example are required to answer simple questions in the form of true/false, multiple choice and free text answers. For convenience, the course navigation bar can be dragged and dropped across the entire course page. The interested reader is encouraged to download available sample AICC content packages via AICC's homepage⁷ to test Moodle's AICC conformance and representation.

Depending on the course format chosen, additional content types can be added. For instance, for the Weekly and Topics course formats *resources* and *activities* can be added, whereas *discussion forums* are available for the Social Format.

Resources represent tools for creating and attaching content to courses. Currently, files, folders, IMS content packages, labels, pages and URLs can be added. On the other hand, activities represent interactive tools Cole and Foster [2008]. As of this writing, assignments, chats, choices, databases, forums, glossaries, lessons, quizzes, SCORM/AICC packages, surveys, wikis and workshops are available to

⁵<http://www.lamsinternational.com/>

⁶<http://scorm.com/scorm-explained/technical-scorm/golf-examples/>

⁷<http://www.aicc.org/SampleLesson/>

The screenshot displays a Moodle SCORM course interface. At the top, it shows the user is logged in as 'Admin User'. The breadcrumb trail is 'Home > Courses > scormformat > Section 0 > SCORM Test Package'. The left sidebar contains a 'Navigation' menu with options like 'My home', 'Site pages', 'My profile', and 'Courses'. Below this is a 'Settings' section for 'SCORM/AICC administration'. The main content area is titled 'Golf Explained - CP One File Per S...' and shows a tree view of course sections: 'Playing the Game', 'Etiquette', 'Handicapping', and 'Having Fun'. The 'Playing the Game' section is expanded, showing 'How to Play', 'Par', 'Keeping Score', 'Other Scoring Systems', 'The Rules of Golf', and 'Playing Golf Quiz'. The 'Playing Golf Quiz' is selected, leading to a 'Knowledge Check' quiz. The quiz asks for the formula to calculate 'course handicap' and provides three options: 'Course Handicap = Handicap index + number of holes + number of lost balls in last round', 'Course Handicap = Number of years experience / annual equipment spending', and 'Course Handicap = Handicap Index * Slope Rating / 113'. The correct answer is the third option. Below the question, there are two example problems with input fields and a 'Submit Answers' button.

Figure 2.8: Moodle Representation of SCORM Course Format for Quizzes

choose from. Note that for each course added, a respective forum is automatically created which serves as the primary place to for instance start discussions or to announce new content.

2.3.1.4 Extensibility

In terms of feature sets and extensibility, Moodle can compete with the big commercial systems, such as Blackboard and WebCT Cole and Foster [2008]. Currently, Moodle provides almost 800 modules and plugins Moodle [2011], which greatly enhance existing core features. Hereby, the core features are easily accessible through structured menus, as shown in Figure 2.7. Context-based navigation options are displayed on the right hand side through a wide range of different topics, such as forum searches, upcoming events, social- and recent activities, as well as course overviews. Again, the interested reader is encouraged to consult the official documentation for a complete list of features available at the project's homepage <http://www.moodle.org/>.

In conclusion, this chapter has provided an introduction into the topic of e-learning. The reader was given an overview of the key concepts and the historical background. Furthermore, it has been shown that certain standards are required to achieve self-contained, reusable and flexible learning materials, so-called *digital learning objects*, that can be packaged into meta-structures, such as *lessons* and *courses* to be used across different e-learning systems.

Based on this background, Chapter 3 presents current e-learning standards.

Chapter 3

E-Learning Standards

“E-Learning is now shifting from a chaotic “no standards” stage, to a phase of rules’ and standards’ definition in an attempt to avoid the Babel syndrome”

[Varlamis et al. [2006]]

Standards play an integral role in the development of software applications. Their advent indicates a certain level of maturity and commercial success. Based on their normative character, they ensure interoperability and integration of systems through a consensus among stakeholders regarding the accepted norms, as well as the criteria for certification Sonwalkar [2002a]. Furthermore, in order to be successful, companies have to choose a distinct set of standards and enforce them strictly Kanendran et al. [2005].

A clear distinction has to be made between technological areas in the process of establishing standards. For instance, hardware standards are generally based upon measurable parameters of physical systems, whereas information technology standards often emerge as normative and informative specifications Sonwalkar [2002a]. As a result, normative standards compete with each other to become the formal industry standard. There is of course, commercial opportunity for companies whose proprietary standards become industry standards by distributing them in the public domain Sonwalkar [2002a]. In the end the winning standard is defined by its ease of implementation and widespread adoption.

The ultimate goal for companies developing LOs (see section 2.2.2) is to make them interoperable across different LMSs (see section 2.3). Consequently in the last decade, the e-learning industry has started several initiatives for the development of industry-wide standards and specifications to promote reuse of learning material, interoperability and integration (Naidu [2006], Cantoni et al. [2004]).

Based on the general introduction into the topic of e-learning in chapter 2, this chapter focuses on presenting two of the most prominent e-learning standards to-date: *SCORM* and *QTI*. Prior to discussing these particular standards, section 3.1 presents an overview of the learning technology standardization process, whereas section 3.2 illustrates the complexity of common standards available today. In conclusion, Section 3.3 discusses evident drawbacks of current e-learning standards.

3.1 Learning Technology Standardization

Until the emergence of standards, organizations were often forced to acquire entire e-learning systems from single vendors, as learning material was tightly coupled to the LMS provided and could therefore not be reused on other systems Fallon et al. [2002].

As a consequence, it was also not possible to mix learning materials (LOs) from different sources, or move complete courses between LMSs, which posed a serious problem for the early adopters of the e-learning technology and decelerated the growth of e-learning communities (Fallon et al. [2002],

Varlamis et al. [2006]). Moreover, collaboration between authors of electronic learning courses was hindered, which eventually contributed to increased development costs Jones [2002]. Thus, in contrast to these proprietary formats and approaches, standardized interfaces and APIs should be defined to ensure interoperability and integration of electronic learning material, independent of the LMS deployed, by using the means of *learning technology standardization processes*.

3.1.1 Objectives

According to Varlamis et al. [2006], workable e-learning standards would satisfy *four main objectives*. First and foremost, standardization would provide users with flexible means to switch between programs and platforms adhering to these standards. Content producer on the other hand could focus on generating standard compliant learning materials instead of being forced to customize content for a variety of different areas of application. Thirdly, LMS vendors could direct their effort at developing standard conformant software instead of filling compatibility gaps between systems. Finally, a vast range of reusable content, as well as standards compliant systems would become available to application and platform designers, enabling a viable e-learning market.

3.1.2 The Process

In general, standardization processes are long-lasting endeavors, which often incorporate laborious and tedious tasks to eventually achieve *accredited standards* Sonwalkar [2002a]. As the pioneer learning technology organizations in the past decade joined forces and developed specification documentation to serve as templates for succeeding *industry standards* Fallon et al. [2002], learning technology standardization process faces the same problem.

To promote conformity and interoperability, critical documentation being collected include information regarding *metadata, course structure hierarchies* (section 2.2.2.2), *data models, LOs* (section 2.2.2), *content aggregations* and *system architectures* Sonwalkar [2002a]. The key member organizations involved in aggregating this documentation and even more importantly proposing it to the designated standards body IEEE LTSC are:

- IMS
- ADL
- AICC

Behind the scenes, these key member organizations collaborate with other projects and companies. For instance, the IMS works together with the *Alliance of Remote Instructional Authoring and Distribution Networks for Europe* (ARIADNE¹) on the topic of metadata definitions. Thus, despite their varying expertise, these organizations are cooperating to promote standards for electronic learning technologies. They individually draw attention on issues in their field of expertise that need to be addressed for the future of learning standards Sonwalkar [2002a]. Based on their proposals, the IEEE LTSC then develops *specifications* and industry standards, which are then submitted to ISO for formal internationalization to eventually become accredited standards Sonwalkar [2002a].

Figure 3.1 depicts the complexity involved in the collaborative integration process of developing formal learning standards. Based on concepts originating from research and development, *technical specifications* are produced by a consortium of AICC, IMS and ARIADNE. These specifications are then used by ADL as a foundation to create test beds for *conformance testing*, which eventually result

¹<http://www.ariadne-eu.org/>

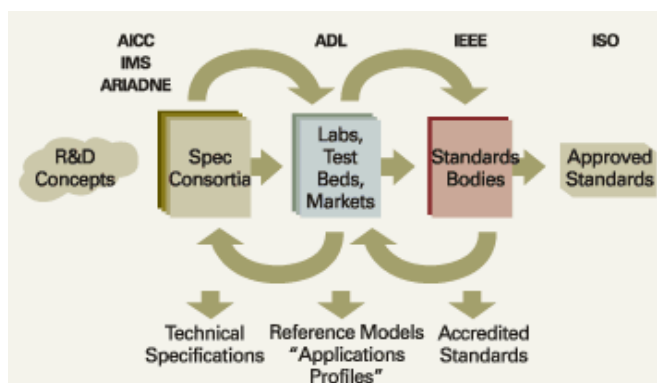


Figure 3.1: Collaborative Development Model for Formal Learning Standards [reproduced from Naidu [2006]]

in *reference models* for selected technologies. These *application profiles* are then submitted to the standards bodies, such as the IEEE to become accredited standards. Ultimately, accredited standards will be submitted to international standards bodies, such as the ISO for formal internationalization to become *approved standards* (Sonwalkar [2002a], Kanendran et al. [2005]).

An essential aspect of the standardization process is that specifications are also reviewed to ensure that they are broadly applicable and do not favour any specifics of given industries and originators. Once a standard becomes accredited, it generally leads to widespread acceptance and implementation Fallon et al. [2002]. Furthermore, with their increasing usage inherent problems simultaneously become evident, which can then be fed back into the standardization process to produce refined specifications in the next iteration.

3.1.3 Related Approaches

Despite the primary learning standard organizations previously mentioned, there exist a handful of related approaches to promote standardization of electronic learning systems and content. For instance, according to Sonwalkar [2002a], the W3C has produced several specifications and standards concerning certain technological aspects, such as additions to XML and Web accessibility standards.

On the other hand, the *Open Knowledge Initiative* (OKI²) pursues the idea of establishing a service-based component architecture and APIs to promote interoperability, as well as activities related to the topic of online learning (Blackboard Inc. [2004], Sonwalkar [2002a]).

Finally, another very interesting approach was made by Vossen and Westerkamp [2008], who introduced a service-oriented e-learning architecture as an alternative to common standards-based e-learning systems. By identifying the major activities involved in learning environments as processes that can be broken down into basic e-learning components, independent *services* can be designed.

The resulting service-based architecture consists of clients and a web-services. Contrary to the approach of using LOs in common e-learning standards, Vossen and Westerkamp [2008] propose that e-learning content also be implemented as services that can be plugged into an integration platform. Thus, instead of physical learning content packages, the service-based architecture is based entirely upon services. According to Vossen and Westerkamp [2008], this approach offers significant benefits over common e-learning standards, as it simplifies them or even makes them obsolete.

Based on the overview of the learning technology and the primary organizations involved in the standardization process provided in this section, the following Section 3.2 presents a distinct selection of two of the most prominent e-learning standards to-date: SCORM and QTI.

²<http://www.okiproject.org>

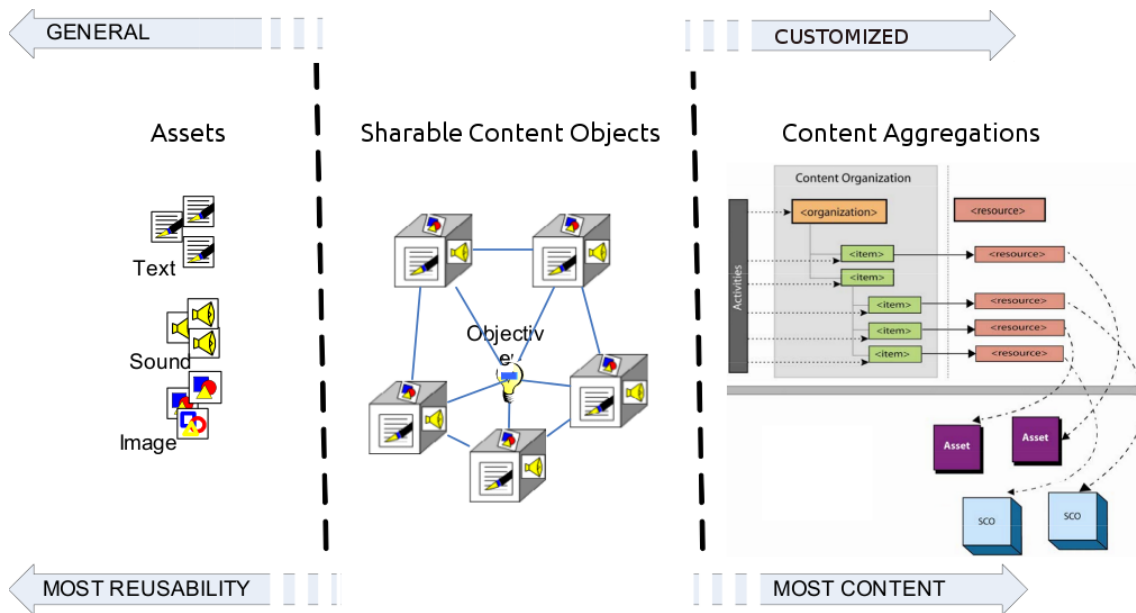


Figure 3.2: SCORM Content Hierarchy [reproduced from Varlamis et al. [2006] and Jesukiewicz [2009a]]

3.2 Standards in detail

As previously mentioned, a broad range of international organizations are involved in the learning technology standardization process. Bush [2002] and GuideTools Ltd. [2009], for instance, list almost twenty organizations focusing on the development and implementation of advanced learning practices.

The process of developing common standards is affected by diverging business interests. Due to this a clear distinction has to be made between industry, or *de facto* standards, and globally accepted accredited, or *de jure* standards, released by designated standards bodies, such as the ISO. Thus, before standards become accredited they must conform to globally valid specifications, independent of the involved companies' interests Fallon et al. [2002].

Two of the most prominent *de facto* e-learning standards for digital learning resources are *SCORM* and *QTI*. Whereas Section 3.2.1 presents *SCORM* in greater depth, Section 3.2.2 focuses on *QTI*.

3.2.1 SCORM

SCORM can be generically described as a set of technical standards for e-learning software Rustici [2009a]. It serves as a *reference model* for a suite of standards developed by a variety of standards bodies, such as the AICC, ARIADNE, IEEE LTSC and IMS (Kanendran et al. [2005], Sonwalkar [2002a]). Instead of reinventing the wheel, *SCORM* merges existing specifications into *SCORM* releases Sonwalkar [2002b].

SCORM adheres to four core functional requirements, called the *RAID* principles (Jones [2002], Deibler [2008]):

- *Accessibility*
- *Reusability*
- *Interoperability*
- *Durability*

The basic concepts behind these requirements have been elaborated in the previous sections. Furthermore, SCORM specifications are based on four main books:

- *Overview* (ADL [2001])
- *Content Aggregation Model* (CAM, Jesukiewicz [2009a])
- *Run-Time Environment* (RTE, Jesukiewicz [2009b])
- *Sequencing and Navigation* (SN, Jesukiewicz [2009c]) Montandon [2004]

Whereas the main concepts and objectives behind SCORM are presented in the Overview, CAM covers specifications for metadata (i.e. LOM), content structures, content packaging, as well as sequencing. On the other hand, RTE focuses on the interoperability between LOs and LMSs by defining a set of APIs and formats. Finally, SN defines the run-time model provided by SCORM which can be used by LMSs to iterate through courses. This thesis focuses on the content packaging aspect of SCORM.

3.2.1.1 Content Packages

Ultimately, SCORM learning material should be packaged into *content packages* (i.e. ZIP files) based on the *SCORM Content Packaging specification*, which in return is based on *IMS Content Packaging Information Model* (IMS [2003], Jesukiewicz [2009a]). Content packages adhering to these standards fulfill three major aspects Rustici [2009b]:

- described with (XML) metadata
- able to communicate via JavaScript with compliant LMSs
- incorporate sequencing rules used for run-time environments

The resulting *self-contained* packages can then be used across multiple LMSs. With regard to the learning object content structures described in section 2.2.2.2, Figure 3.2 depicts SCORM's content hierarchy. As previously mentioned, assets are the most granular units of information. Due to their limited information content, they are grouped in meta-structures, so-called *Sharable Content Objects* (SCO), which represent SCORM's counterpart for digital learning objects. SCOs can then again be embedded into bigger structures, forming *content aggregations*. Finally, in order to exchange these encapsulated content aggregations, they are embedded into content packages.

As shown in Figure 3.2, SCORM's content aggregations are composed of several components. Hereby, *activities* represent what Jesukiewicz [2009a] defines as “meaningful units of instruction” that are contained inside content *organizations*. Content organizations in return are meta structures that represent the intended use of contained activities. They provide the structural foundation to (optionally) define run-time *sequences* for the activities contained. SCORM's sequencing and navigation aspect is not covered in greater detail in this thesis. Nevertheless, the interested reader is encouraged to consult Jesukiewicz [2009c] for more information.

As depicted in Figure 3.2, activities can refer to SCOs or assets directly (i.e. *resources*), or include subsequent activities forming learning taxonomies, such as *lessons*, *modules* and *courses*. In order to remain context-free, content that is suitable for incorporating it into other courses should be designed as a SCO Jones [2002].

Figure 3.3 depicts the general structure of SCORM content packages which represent the basic “unit of learning” Jesukiewicz [2009a] and can be separated into two key components: *manifest* and *actual content*. Whereas manifest content defines a structured inventory, the contents of packages, including *metadata* declarations, *organizations*, *resources* and additional *sub-manifests*, the actual content resides

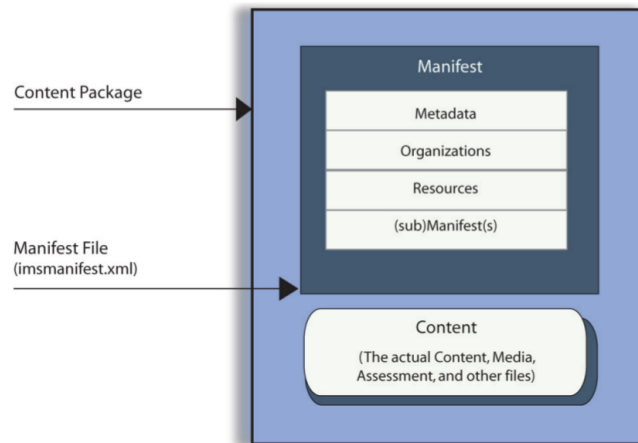


Figure 3.3: SCORM Content Package [Jesukiewicz [2009a]]

in separate structures. It follows that metadata included in the manifest does not only describe the content aggregation as such, but also enables it to be identified and discovered during searching Jesukiewicz [2009a].

3.2.1.2 Manifest

A exemplary manifest file is shown in Figure 2.4. Once again, it depicts the main sections of SCORM manifest files and also illustrates the aforementioned concept of linking arbitrary resources to organizations. For instance, this SCORM content package contains one content organization (*MyCourse*) that in return is comprised of a single SCO (*OneSCO*). Finally, the SCO itself contains two resources (*assets*), *index.html* and *end.html*. Moreover, the metadata section defines this package to be SCORM version 1.2 conformant. Note that this example does not include sequencing specific information, as it contains only a single activity. SCORM manifests must always be named *imsmanifest.xml* Jesukiewicz [2009a].

3.2.1.3 Package Interchange File

SCORM content packages should be created according to the *Package Interchange File* (PIF) format Fallon et al. [2002]. As the name suggests, it is a standardized exchange format for content packages. It comprises entire content packages in single archive-formatted files, such as a ZIP files. Ultimately, if content packages are created using PIF, they must conform to RFC 1951 and be archived using PKZip (.zip), which in return conforms to RFC 1951 Jesukiewicz [2009a].

3.2.2 QTI

In contrast to SCORM, the *IMS Question and Test Interoperability Specification* (QTI) is a more specialized specification targeted at defining methods for sharing test questions and corresponding results amongst arbitrary learning management systems Fallon et al. [2002]. Its key concept is to eliminate the need for separate physical content files by *describing* the test's and question's contents, presentation styles and behavior Fallon et al. [2002]. As a consequence, it is the responsibility of the learning system to interpret these specifications and present them in a standardized fashion to end users.

QTI specifically targets content providers, such as question and test authors IMS [2006]. It is primarily based on two main data models:

- *Assessment Test, Section, and Item Information Model* (ASI)

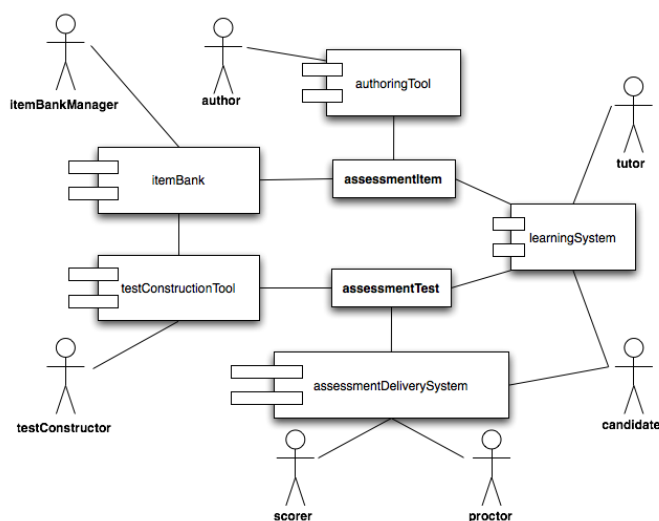


Figure 3.4: QTI Assessment Model [IMS [2006]]

- *Results Reporting* model

Whereas the ASI provides learners with actual test content, response processing, sequencing functionality and test scoring, the *Results Reporting* model enables standardized storage of test results and usage in different contexts Fallon et al. [2002]. In the following, ASI will be presented in greater detail.

3.2.2.1 Assessment Test, Section, and Item Information Model

ASI content follows a strict hierarchy. *Items* represent individual questions, *sections* describe structured groups of questions, *assessments* represent entire tests and *object banks* describe unstructured “packages” of other QTI components Fallon et al. [2002]. The Results Reporting model on the other hand defines four types of result, depending on the required level of granularity Fallon et al. [2002]: *summary-*, *assessment-*, *section-* and *item results*.

In order to illustrate this concept, Listing 3.1 shows a minimal example of a QTI-conformant test. The most important XML entities are marked in bold letters. Line 1 defines the assessment (i.e. the test), which serves as container for questions and respective answers. Line 2 starts the definition of the section *SingleSection*, which in this example contains a single item, i.e. the question. Attached to the question are two answers, *True* and *False*. The *presentation* container serves as meta structure for elements contained in an assessment. Due to QTI’s underlying principle to not define styling properties via assessment definitions, only structural information is specified in the presentation section. *Material* entities (e.g. Line 5) define the actual contents of test entities, such as the question’s text.

Although this exemplary assessment reflects only a simple true/false scenario, QTI contains specifications for a multitude of different question types, such as multiple choice, drag&drop and select text (Fallon et al. [2002], IMS [2006]). Figure 3.5 shows the visual representation of the QTI test from Listing 3.1 using Moodle, as described in section 2.3.1. Again, it is important to note that no styling properties are attached to the assessment, as it is the learning management system’s duty to interpret and visualize tests.

The basic concept of QTI and the main actors involved is depicted in Figure 3.4. In order to assemble tests (*assessmentTests*), questions (*assessmentItems*) are selected from so-called *itemBanks*, which serve as storage for QTI assessments. During the tests, result reports generated which are scored by the *assessmentDeliverySystem*, or redirected to a special *scorer*, which can either be a person or an external system IMS [2006]. The remaining actors and components shown in Figure 3.4 are self-explanatory.

1 Is this a sample QTI question?

Marks: -1.00

Answer: True False

Submit

Figure 3.5: Moodle Representation of QTI Test from Listing 3.1

3.2.2.2 QTI Lite

Due to its aim to address every possible scenario, QTI has become a very complex specification Fallon et al. [2002]. As a result, *QTI Lite* has been developed, which defines the requirements to develop the simplest form of QTI-conformant content IMS [2002]. Despite the vast range of question-types available in QTI, only a fine selection of base types are allowed in QTI Lite, such as multiple-choice and Likert scale questions IMS [2002]. It also limits or even cancels support for various other aspects, such as additional metadata, extensions or time-based functionality.

```

1 <assessment title="QTISample" ident="QTI_Test_1">
2   <section title="SingleSection" ident="Section_1">
3     <item title="SampleItem" ident="Question_1">
4       <presentation>
5         <material>
6           <mattext>Is this a sample QTI question?</mattext>
7         </material>
8         <response_lid ident="Question_Q_1" rcardinality="Single" rtiming="
          "Yes">
9           <render_choice>
10            <response_label ident="Answer_1">
11              <material>
12                <mattext>True</mattext>
13              </material>
14            </response_label>
15            <response_label ident="Answer_2">
16              <material>
17                <mattext>False</mattext>
18              </material>
19            </response_label>
20          </render_choice>
21        </response_lid>
22      </presentation>
23    </item>
24  </section>
25 </assessment>

```

Listing 3.1: QTI Sample Test Question

3.3 Criticism

Existing e-learning standards, such as ADL's SCORM and AICC's counterpart are complex structures. They are comprised of several hundred pages of specifications, specifically targeted at including every

possible scenario covering the broadest range of applications. Furthermore, numerous organizations are involved in the long-enduring standardization process, further contributing to its complexity.

For less intricate applications the cost-benefit ratio to conform to these standards might be especially not feasible. This additional layer of complexity poses a steep barrier for small commercial and individual developers Godwin-Jones [2004]. Moreover, although SCORM for instance has been widely adopted in the e-learning field, it is still under development and leaves room for interpretations which impact LMS learning content developers (Vossen and Westerkamp [2008], Rustici [2009a]).

Another important aspect that should not be ignored is the differentiation between learning content producers and projects that also need to implement corresponding LMSs and LCMSs. Although there exist open source LCMSs such as Moodle, additional effort might be required to customize existing functionality, for instance. Furthermore, developing standards-conformant LMS or LCMS from scratch requires tremendous effort and thorough understanding of the complex standards involved.

3.3.1 EduPunk

As a consequence, there are also exist critics of the implied conformity and complexity of e-learning standards, such as Jim Groom, who introduced the concept of *EduPunk* Brooks [2008]. This term is adopted from the Punk ideology of the 1970's and projected on the e-learning sector, as a reaction against the conformity of e-learning standards and tools Young [2008].

Although no exact definition can be found, Brooks [2008] refers to it as a “scrappy, do-it-yourself spirit in some sectors of educational technology”. According to Downes [2008], another supporter of EduPunk, the fact that no exact definition can be found shows “that true edupunks deride definitions as tools of oppression used by defenders of order and conformity”.

3.3.2 Diminishing Pedagogical Aspect of Learning

Moreover, Marshall [2004] argues that the majority of the standards involved in the e-learning sector are not truly learning standards per se, but rather technical, *computer* standards, required for instance in making entire e-learning systems and their learning materials interoperable.

According to Marshall [2004] they do not contribute to the actual educational outcomes, as they do not provide for the *pedagogical* aspects of learning, but rather focus on the technical aspects. He concludes that the insufficient attention to pedagogical aspects of e-learning in favor of extensive technical specifications is based on the fact that educators appear to have “left this area firmly in the hands of technologists” Marshall [2004].

The core of this problem according to Marshall [2004] lies in the fact, that “learning is not a tidy, mechanical process that responds well to rigid frameworks and defined, quality assured, processes and checklists”. On the contrary, it seems that learners are much more motivated when they are able to freely communicate and engage with each other, rather than being forced into a fixed learning schema defined through standards. Thus, although he acknowledges that the “integration of pedagogical concerns into standards is challenging”, he concludes that if e-learning standards recognize the “learning context more explicitly” learners and teachers alike will profit greatly from it.

Another interesting point made by Marshall [2004] is that in fact the core “benefits” of e-learning standards are its *economic aspects*. He points out that these benefits “seem to be dominated by issues and outcomes that seem only distantly related to the human processes of learning and teaching”. Thus, he believes that the true objectives of e-learning heavily interfere with inherent economic aspects.

3.3.3 Conformity through Generality

Ultimately, e-learning standards should represent “a defined model of e-learning” that enables the “deterministic development of successful e-learning environments” Marshall [2004]. Unfortunately, according to Marshall [2004] past experience has shown that the effort spent in creating standards is rendered useless if it is “applied in a deterministic, mechanistic way”.

Aside from the technical aspects of e-learning standards, Cressman and Friesen [2005] also discuss the inherent local, heterogenous and contextual nature of education and learning. They argue that limitations due to the conformity of e-learning standards, localized know-how of how to teach for instance universal scientific formulas is not likely to be replaced even by the best designed electronic learning material or standardized practice. They conclude that existing resistance towards e-learning standards is based on the understanding of the localized nature of education.

Moreover, although Marshall [2004] acknowledges the efforts spent on developing e-learning standards and specifications, he also warns that they might not be universally applicable. Furthermore, he also argues that standards can be applied to obscure but important details in order to simplify complex issues. He concludes that one is easily seduced by technology and hence implies that the greater use of technology automatically results in more effective learning delivery. Thus, e-learning standards should also tap into the average learner’s individual creativity rather than strictly demonstrating compliance to existing technology Marshall [2004].

In conclusion, it should be stated that based on the complexity of current e-learning standards such as SCORM, smaller sized projects might opt for proprietary standards and tools, aside from the fact that they most likely refrain from developing entire standard-conformant LMSs and LCMSs.

In the case of the ALEXIK project, described in chapter 4, we have decided to use proprietary standards and common web standards to implement an e-learning platform called Wörterwelt. This approach concurs with Godwin-Jones [2004], who states that although for instance SCORM integration may make sense in specific environments, its controlled environment may not be ideal for all aspects of language learning.

Chapter 4

Practical Implementation

“Technologies are not inherently good or bad, but are only good or bad depending on how they are used.”

[Oaho et al. [2009]]

The previous chapters provided a thorough introduction to the topic of *e-learning* and its current *standards*. It has been shown, that e-learning systems are still gaining popularity in a variety of applications (Fallon et al. [2002], Eklund et al. [2003], Naidu [2006]). Although standards play an important role, especially when designing software systems, it has been shown that current popular e-learning standards often are too complex when implementing less intricate e-learning applications.

Based on the previous overview, this chapter presents a practical implementation of an *asynchronous e-learning platform* called *Wörterwelt*¹, which has been implemented in the course of the ALEXIK project in cooperation with the Austrian German Research Center² and the Institute for Information Systems and Computer Media³ at the Graz Technical University.

First of all, the reader will be given a description of the ALEXIK project, including its aims and requirements, the process of generating and transforming linguistic data developed by linguist experts, the target audience, as well as the people involved. Following this introduction, the body of this chapter provides a thorough description of the practical e-learning platform implementation called *Wörterwelt*, including its two fundamental parts, the *dictionary module*, discussed in section 4.2, as well as the *exercises module*, presented in section 4.3. Furthermore, the tools and technologies used throughout the entire planning and development process will be discussed. Afterwards, chapter 5 presents the results and lessons learned from the formal experiment conducted between the two development phases of the practical implementation.

4.1 Motivation

At the beginning of the Austrian school year 2005/06, a corpus-based school dictionary, called *Wörterwelt*, was published for the first time by Muhr and Kadric [2005]. Developed in the course of a project sponsored by the Austrian Federal Ministry for Education, Arts and Culture⁴, it has also been added to the official Austrian schoolbook list Austrian Federal Ministry for Education, Arts and Culture [2010]. Whereas the first edition originally comprised the languages German, Bosnian, Croatian and Serbian

¹<http://www.woerterwelt.at>

²<http://www-oedt.kfunigraz.ac.at/>

³<http://www.iicm.tugraz.at/>

⁴<http://www.bmukk.gv.at>

with a total of 12,905 entries, Turkish, Albanian and Czech were added in the second edition, resulting in an elaborate language corpus of almost 70,000 entries.

Based on this work, the *ALEXIK* project was established with the goal to create a standards-based, open-source *e-learning platform* called *Wörterwelt*⁵ to promote the integration of languages for non-German speaking children. Its main aims can be separated into five distinct categories. First of all, a dictionary application was developed and integrated into the platform. Secondly, existing contents of the *Wörterwelt* print dictionary were transformed and imported into the dictionary application. Furthermore, in order to enrich the existing language corpus, additional translation languages including Albanian, Czech, Slovakian and Turkish were to be generated. Moreover, interactive, uni- and multilingual exercises were developed that pinpointed typical language differences. Finally, the e-learning platform itself was provided in two versions, *online*, as well as *offline*, e.g. using a CD-ROM.

Based on these aims, the *ALEXIK* project basically had two main objectives. On the one hand, linguist experts were to generate additional translations for the existing language corpus, in order to reach a broader spectrum of potential users. On the other hand, from the technical perspective the main goal was to develop an e-learning platform, specifically targeted at non-German speaking users (children) that consisted of a *dictionary-* and an *exercise module*, as shown in Figure 4.1. To this purpose the middleware layer, consisting of additional libraries and tools required by the modules, had to abstract access to the underlying database and web server through a standardized interface.

Apart from the conventional search functionality, such as wildcard searching, case-insensitivity and support for an arbitrary amount of translation languages, the dictionary module also provided an auto-completing search query box, to facilitate the selection of search terms. Search results were presented as a dynamic grid, allowing users to browse the dictionary, much like leafing through a printed dictionary. On the other hand, the *exercises module* served as the primary interface for interactive and adaptive language exercises. They were organized into *learning packages* which were based upon a previously defined didactic-relevant *exercise topology* and comprise an arbitrary set of *base exercise types*, including multiple choice, simple text, drag&drop and text selection. *Learning courses* were then represented as temporal sequences of arbitrary base exercise type combinations from a variety of learning packages. Furthermore, through a simple *feedback* mechanism, users were able to determine their overall course progress.

Thus, expertise from two different fields of research was required for this project: Linguistics and computer science. On the one hand, experienced linguists were required to generate and manage translations for a variety of languages, as well as design exercises specifically targeted at a young audience. On the other hand, software engineers were required to design and implement the technical aspects of the e-learning platform. The fact that both aspects are equally important becomes evident by imagining a perfectly designed e-learning software platform without accurate underlying linguistic data. From the other point of view, imagine optimally structured language data that cannot be accessed efficiently, due to a poorly performing software platform. Therefore, these activities were split, producing two different work groups. Whereas the linguist experts, lead by Prof. Rudolf Muhr of the Austrian German Research Center, were responsible for managing the linguistic data, the software engineers, lead by Prof. Denis Helic of the Graz University of Technology, have been put in charge of the technical planning and implementation of the e-learning platform *Wörterwelt*.

The target audience represented a further demanding aspect concerning the design and implementation of the e-learning platform. The main user group were children at around the age of around ten, currently attending the 4th grade elementary school. Consequently, several aspects needed to be taken into consideration, especially when designing the user interface (UI). With regard to the young target audience, we decided to conduct a simple usability study in form of a formal experiment between the two main development phases. More detailed information on this study, including the feedback information gathered and the lessons learned is presented in chapter 5.

⁵in accordance with the naming of the school dictionary *Wörterwelt*

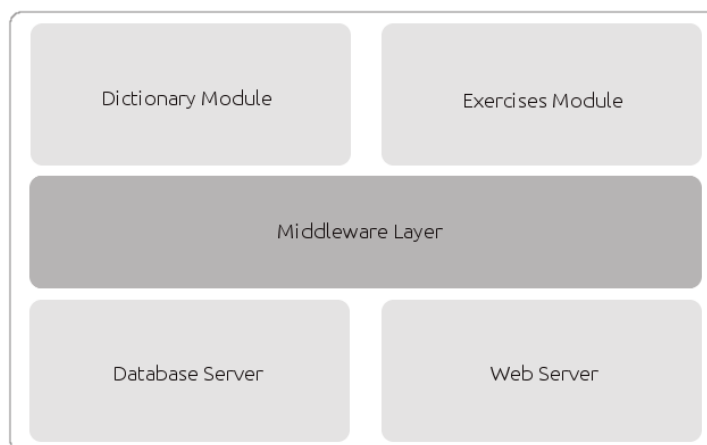


Figure 4.1: E-Learning Platform Wörterwelt Architecture Overview

Based on the objectives discussed in this section, the remainder of this chapter is dedicated to presenting the practical implementation of the e-learning platform *Wörterwelt*, which is divided into two main parts. Whereas section 4.2 discusses the implementation of the *dictionary module*, section 4.3 presents the work done for the *exercises module*.

4.2 Dictionary Module

The dictionary module represents the first of the two modules incorporated in the e-learning platform *Wörterwelt*. It is an interesting and challenging task to design and implement a web-based dictionary, requiring deep insight into Web technologies in general, as well as a thorough understanding of the additional tools and libraries used throughout the implementation. Moreover, due to the need for special purpose tools to convert existing dictionary data to the internal format used by this module, additional tools have been developed alongside the main implementation. Before presenting the actual implementation in section 4.2.3, the following section presents the module's goals and tasks, which impacted on the final decision for the tool and technologies described in section 4.2.2.

4.2.1 Goals and Tasks

Prior to the beginning of the implementation, the dictionary module's main goals and tasks had to be identified. The main goals could primarily be derived from the project proposal and extended by user requirements collected during the requirement specification phase. Furthermore, additional goals could be identified based on personal experience in the field of designing and implementing web applications. Consequently, the following list represents the main goals identified for the dictionary module, ordered by priority, starting with the most important:

- Web standards- and browser-based implementation
- language independent data representation
- lightweight and standards-based client-server data exchange protocol
- *Ajax*⁶-based presentation layer

⁶Asynchronous JavaScript and XML Garrett [2005]

- dynamic results grid that supports paging
- server-centric architecture minimizing additional client side code
- widget-based framework architecture enabling rapid development
- platform independent implementation
- support for online and offline execution
- web server support for framework(s) used
- administration interface for unified data management (backend)
- unified method to extract and import linguistic data

Based on the aforementioned goals, the next step was to extract the corresponding tasks, which had influenced the choice of the tools and technologies used for the final implementation. The main tasks could be separated into two categories, the dictionary module web application itself, as well as additional tools that provide unified means for extracting and importing linguistic data into the application's database. The following list presents the dictionary module's main tasks:

- asynchronous processing of GET and POST requests
- unified handling of client-server XML data
- handling of multi-byte character encoded data (i.e. UTF-8) to maximize language compatibility
- serving template- and Ajax-based presentation layer, to enable desktop-like application experience, including paging of results
- provide SQL-based search and retrieval functionality for linguistic data

Furthermore, the main task assigned to the additional tools was to provide a *unified* process to

- extract linguistic data from existing translations in the form of Microsoft Word documents
- convert existing syntax and formatting rules to customized HTML code
- import resulting translation data into the e-learning platform, while checking for existing entries

Upon identification of the main tasks, the next step was to determine the appropriate tools and technologies for the final implementation, which will be described in section 4.2.2.

4.2.2 Technology and Tools

Choosing the right set of tools and technologies for specific applications generally is a difficult task. Poorly chosen tool-technology combinations will most certainly have a negative influence on the success of the software project. Consequently, determining the trade-offs between possible software package combinations is a definitive task in the software engineering process. Furthermore, the fact that the decision for a specific software programming language is closely related to the range of supported tools and frameworks makes this decision even more critical. Moreover, especially when it comes to web-based applications, standards-compliance and browser-independence play a vital role to achieve high user-acceptance rates on the one hand, as well as low maintenance-effort on the other.

At the time of planning the dictionary module, various Ajax-based frameworks had already been available. Some were more advanced than others compared to their feature set, some were still in beta stage or even worse, limited in their stability and support. In order to decide for the best suitable framework/programming language combination, a set of simple prototypes have been implemented and evaluated. In the following, a quick summary of the prototypical language/framework configurations tested during the prototyping phase will be discussed. Finally, the configuration chosen for the actual implementation will be presented in greater detail.

Configuration 1: Zend Framework, Dojo Toolkit, HTML

Using a combination of the Zend Framework⁷ and the underlying PHP⁸ interpreter programming language for the server- and the Dojo Toolkit⁹ on the client side, a very simple prototype has been implemented to test its practical usage. The shortcomings of this configuration were the need for two rather complex frameworks and a considerable amount of additional programming on the client side to achieve a dynamic grid to be used for the dictionary frontend, as described in section 4.1.

To be fair, it has to be stated that at the time of the prototyping phase Dojo's Grid widget¹⁰ that could have been used as the primary template for the client side dictionary implementation was still in an early developmental stage. Even as of this writing, it is still not part of Dojo's *Core*¹¹ distribution, or its corresponding official widget library (*Dijit*). Nevertheless, in the meantime overall support for the combination of the Zend Framework and Dojo has improved dramatically, due to the *Zend_Dojo* module Zend Technologies Ltd. [2010].

Configuration 2: PHP, script.aculo.us, HTML

Another possibility would have been to use plain PHP for the server side backend and a slim client side JavaScript framework, such as script.aculo.us¹² to implement the dictionary's grid. Using this approach requires a lot of client side programming effort, as at the time of this prototype script.aculo.us did not provide a ready-to-use dynamic grid widget. On the other hand, there would have been a rather lightweight backend, that merely processed search requests by responding with the corresponding result data using a unified exchange format, such as JSON¹³ or XML¹⁴. In the end, the additional client side programming effort required outran this factor.

Configuration 3: Groovy, script.aculo.us, HTML

The main idea behind this configuration was to use a server side Java-based framework to handle backend operations and a client side JavaScript framework that provides a thin, browser-independent and Ajax-based user interface. Although Groovy¹⁵ offered a lot of advantages compared to the configurations using plain PHP, such as type safety, still code for two separate frameworks had to be maintained, which in the end overpowered the positive aspects.

Configuration 4: ZK, ZUML, HTML

This combination represents the winning configuration, due to two essential factors: *simplicity* and *robustness*. Using solely the JavaServer Pages based¹⁶ ZK Ajax framework¹⁷ on the server side, no additional client side frameworks were needed, as it provides ready-to-use *components* in the form of

⁷<http://www.zend.com>

⁸<http://www.php.net>

⁹<http://www.dojotoolkit.com>, also see Section 4.3.2.1 for a more detailed description

¹⁰<http://www.dojotoolkit.org/reference-guide/dojox/grid/DataGrid.html>

¹¹see Section 4.3.2.1

¹²<http://script.aculo.us/>

¹³JavaScript Object Notation, see Section 4.3.2.3

¹⁴<http://www.w3.org/TR/REC-xml/>

¹⁵<http://groovy.codehaus.org/>

¹⁶<http://java.sun.com/products/jsp/>

¹⁷<http://www.zkoss.org>, also see Section 4.2.2.1

so-called *widgets*, such as the Grid¹⁸. Hereby, components are entirely based upon pure Java classes (JavaBeans). Thus adding or changing functionality can be achieved by implementing the appropriate interfaces, which enables rapid development. Furthermore, ZK fully supports scripting-codes inside presentation layer files using the Expression Language (EL). Thus, apart from components being managed from the server side, they can also be directly accessed and manipulated from within the presentation layer.

Moreover, ZK fully automatic takes care of the tedious task to generate browser-independent JavaScript and HTML code, without requiring any special further programming on the client side, making additional client side frameworks obsolete. By default, ZK also uses XML as standardized message exchange format. Another positive aspect of ZK is the support for supplementary development branches, such as the one for Java- and Android-based mobile phones Potix Corp. [2009b], which provides essential benefits regarding future applications, as mobile devices are more and more becoming one of the major targets of web applications Qiu et al. [2004].

Combining these factors, ZK would provide significant advantages over the alternative prototype configurations. Consequently, in the end, the ZK Ajax Framework as server side framework and Java as corresponding programming language were chosen. The final configuration was determined by its minimal dependency on external libraries and frameworks, as well as its flexibility and stability. The following section provides a more detailed description of the ZK framework, before diving into the discussion of the module's implementation in section 4.2.3.6.

4.2.2.1 ZK

“ZK is an open source Ajax web framework that enables a rich user interface for web applications with no JavaScript and little programming.”

[Potix Corp. [2010g]]

ZK represents an event-driven, component-based framework that enables rich user interfaces for web applications (RIA) Chen and Cheng [2007]. Basically, it consists of a powerful Ajax-based, event-driven engine at its heart, a rich set of XML User Interface Language (XUL) and XHTML components, as well as its own markup language, called ZK User Interface Markup Language (ZUML¹⁹). Hereby, ZK applications represent collections of ZK User Interface Language (ZUL) pages, that in return may comprise an arbitrary amount of XUL and XHTML components, written in the proprietary ZUML format, as depicted by Figure 4.2.

In contrast to many other Ajax-based frameworks, ZK's incorporated Ajax-technology plays a “behind-the-scenes role” Chen and Cheng [2007]. By using a server-centric approach, the entire client side event-handling mechanism is delegated to ZK, thus making it possible to directly interact with the client side from within the server side, hence the name *Direct Ajax framework*. Furthermore, ZK's standards conformance successfully hides browser incompatibility hassles Seirand Institute [2008]. This provides developers with the possibility to concentrate on the application's essential business logic.

According to Chen and Cheng [2007], ZK's main characteristics can be broken down into three categories. First, its character as a *presentation layer tool*, secondly, its incorporated *server-centric model* and finally its light-weight, *component-based GUI*. Based on its server-centric model, “everything” is done at the server side, ranging from assembling UI-components, over processing events triggered on the client side, to reacting to them by sending appropriate (update) responses Potix Corp. [2010c]. Due to the fact, that ZK was designed to be independent of additional backend technologies, its main focus lies on a thin *presentation tier*.

¹⁸<http://www.zkoss.org/javadoc/5.0/zk/org/zkoss/zul/Grid.html>

¹⁹http://docs.zkoss.org/wiki/ZUML_Overview

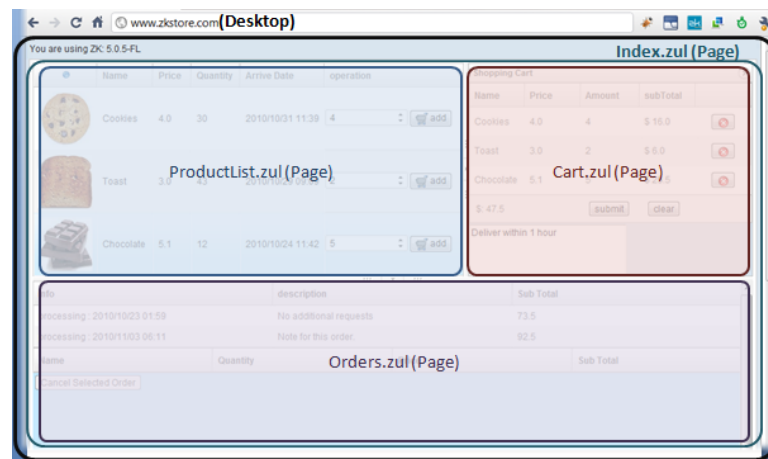


Figure 4.2: ZK Application as a Collection of ZUL Pages [Potix Corp. [2010e]]

Figure 4.3 summarizes ZK’s core architecture. It is comprised of three main parts, the client side *ZK Client Engine* and *ZK AU Engine*²⁰, as well as the *ZK Loader* residing on the server side Chen and Cheng [2007]. Whereas the ZK Client Engine is entirely written in JavaScript and takes care of handling browser-triggered events, the ZK Loader and the ZK AU Engine are composed of Java servlets. In order to synchronize events between the server- and the client side, the ZK Client Engine and the ZK AU Engine²¹ exchange messages in the form of XML-based Ajax requests.

This client-server information exchange can be divided into two categories, one being the *initial page loads* and the other spontaneous, event-based Ajax *update-requests*. This differentiation also illustrates the purpose of the ZK Loader, compared to the ZK AU Engine. Whereas the ZK Loader handles incoming initial page-load requests by generating the corresponding HTML page, the ZK AU Engine takes care of processing Ajax requests received from the ZK Client Engine by sending the appropriate Ajax response. Consequently, ZK Loader’s job is to basically “bootstrap” requested pages, while the ZK AU Engine takes care of maintaining a synchronized state between the client- and the server side through Ajax requests.

As shown in Figure 4.4, there are a couple of basic steps involved in processing events triggered on the client side. For instance imagine the case when a DOM²² *onClick*-event is triggered. The corresponding widget then notifies the ZK Client Engine, which in return sends an Ajax request to the ZK AU Engine residing on the server side. Since widgets and components share a 1:1 connection, the corresponding component gets notified of the event and takes the appropriate measures. Finally, the ZK Update Engine sends an Ajax response to the ZK Client Engine, which in return forwards the update event to the original widget. In ZK, this synchronization mechanism is completely transparent to the application, thus giving developers total control of handling events Potix Corp. [2010f].

ZK’s underlying component structure begins by differentiating between a *Desktop* and *Page*. As shown in Figure 4.2, it is possible to nest ZUL pages that are all serving the same URI. Due to this the concept, the *Desktop* component was introduced. It serves as a container for *Pages* belonging to a specific application, enabling developers to interactively add/delete *Pages* (or to that effect any other nested component) from a *Desktop*.

Since ZK components are entirely written in Java, they can be changed rather easily. Hereby, components follow strict interface declarations, thus substituting them can easily be achieved. As previously mentioned, components represent the server side Java implementation of the corresponding client side

²⁰ZK Asynchronous Update Engine

²¹now called ZK Update Engine, see Figure 4.4

²²Document Object Model W3C [2005]

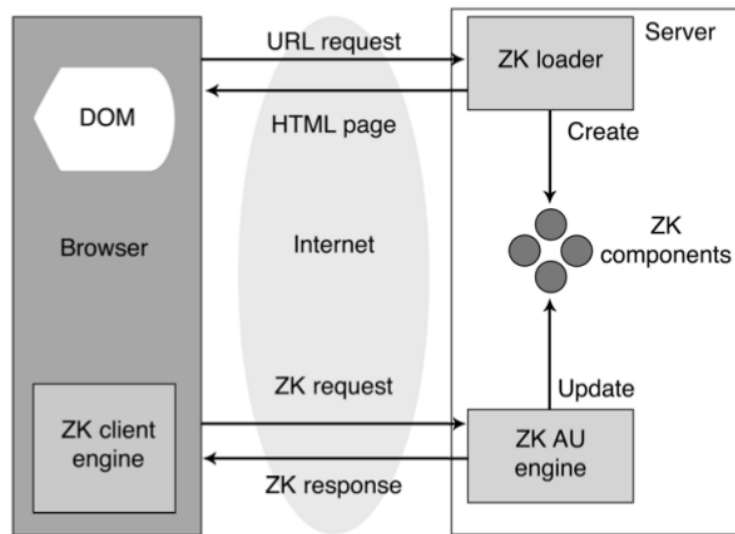


Figure 4.3: ZK Architecture [Chen and Cheng [2007]]

widgets, written in ZUML, XUL or XHTML and contained in ZUL pages. The resulting widgets are automatically generated by ZK, using JavaScript, as shown in Figure 4.5. Applying what Potix Corp. [2011] calls *Direct RIA*, these widgets can then be directly managed from the server side using the appropriate Java components. Consequently, styling widgets can be done as usual by defining the appropriate tags in the ZUL pages, whereas the actual functionality is defined on the server side.

Through a vast collection of built-in components ZK provides developers with tools for a rich user experience. Currently, there are over one hundred state-of-art, Web Accessibility compliant Ajax components with versatile RIA features Potix Corp. [2010b]. Although they can be used directly out-of-the-box, further customizations can be achieved by modifying the respective component classes.

In order to keep the server side synchronized with the client side the ZK framework uses what it calls “*Direct Push*” technology, which enables applications to send spontaneous updates to clients using minimal effort and costs Potix Corp. [2010b].

Although writing Java classes is the more rigid route when building more elaborate Ajax applications, ZK also supports the use of scripting-code and EL- expressions inside ZUL pages. Currently supported scripting-code languages range from Java to Groovy and there are more are still to come. But unlike JavaScript for example, ZK executes these codes on the server side, therefore relieving the client’s browser.

One of the most common problems when using (emerging) open source frameworks is the lack of documentation. Unlike many other well known frameworks, ZK provides comprehensive, up-to-date documentation and great user support through various forums. Furthermore, based on personal experience, feature requests are processed rather quickly. Another point to be made is ZK’s separate development branch for mobile devices, called *ZK Mobile*. This compact framework was specifically developed to be used on resource-scarce mobile devices and currently offers a handful of so-called Mobile Interactive Language (MIL) components Potix Corp. [2009a].

Apart from the actual framework, Potix Corp.²³, the company behind ZK, also provides *ZK Studio*²⁴, an advanced Eclipse²⁵ plugin that serves as an integrated development environment (IDE) for ZK-based projects Potix Corp. [2009c]. Apart from standard features, such as context-aware code completion it further offers WYSIWYG editors for ZUL pages, called *ZUL Visual Editor*, as well as the *ZK Style*

²³<http://www.zkoss.org/support/about.dsp>

²⁴<http://www.zkoss.org/download/zkstudio.dsp>

²⁵<http://www.eclipse.org>

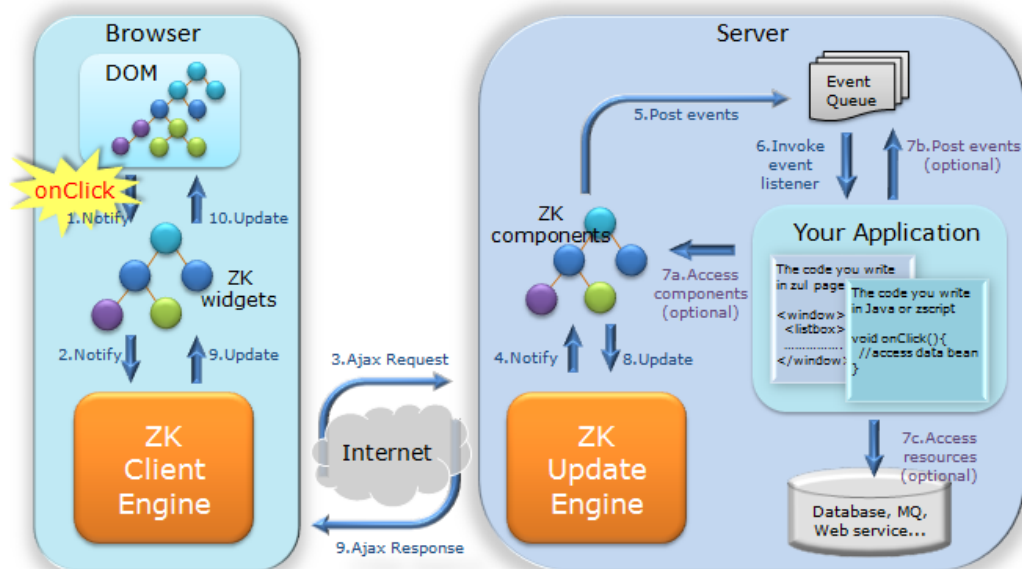


Figure 4.4: ZK Flow of events [Potix Corp. [2010c]]

Designer that represents a GUI to edit CSS styles for ZUML components.

ZK provides a very rigid way to write Ajax-based web applications. It is based on a clean architecture, with simplicity in mind. As a summary, Potix Corp. [2010a] lists the top reasons why one should decide in favor of ZK, including support for a rich user experience, its characteristics of an open source, standards-based Direct RIA framework, its support for various scripting languages and Direct Push technology, its extensibility, customizability, security, as well as scalability through support of clustering and failover mechanisms, mobile access through additional development branches and finally its optional enterprise support.

4.2.2.2 Apache Tomcat

In order to be able to deploy ZK-based applications, a Java Servlet container is required. As of this writing, ZK officially supports ten different servers ranging from Apache Tomcat, through Oracle to the Google App Engine Potix Corp. [2010h]. The final decision has been made in favor of Apache Tomcat, due to its support for ZK, cost factors, as well as previous positive experiences.

4.2.2.3 Java

Wörterwelt's dictionary module has been developed in Java, based on the ZK framework presented in section 4.2.2.1, using Sun's Standard Development Kit (SDK²⁶) 1.4, but has also been successfully tested with the more current versions 1.5 and 1.6.

4.2.2.4 MySQL

According to the e-learning platform's overall objectives described in section 4.1, linguistic data should be kept in a central database. The MySQL²⁷ database server was selected for this purpose. In order to bridge the gap between Apache's Tomcat and the MySQL server the MySQL Connector/J JDBC

²⁶<http://java.sun.com>

²⁷<http://www.mysql.com>

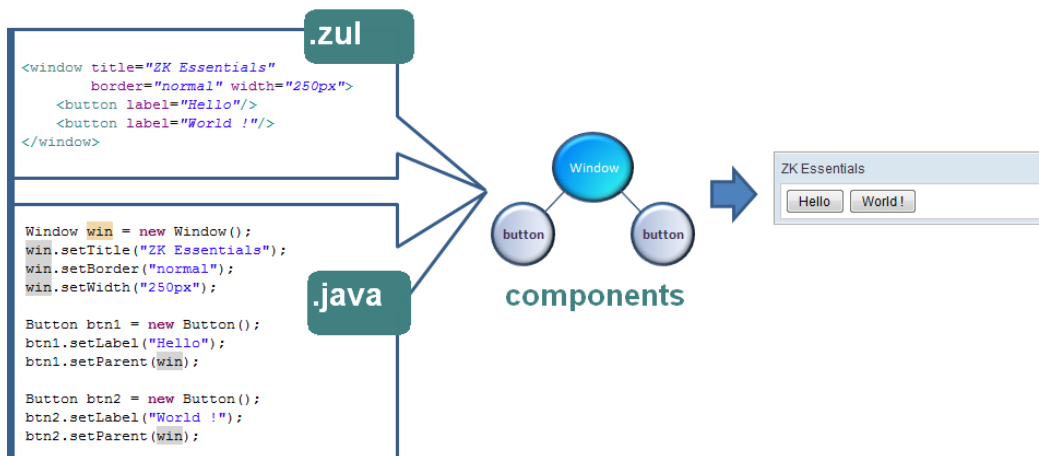


Figure 4.5: Connection between server side components and client side widgets [Potix Corp. [2010e]]

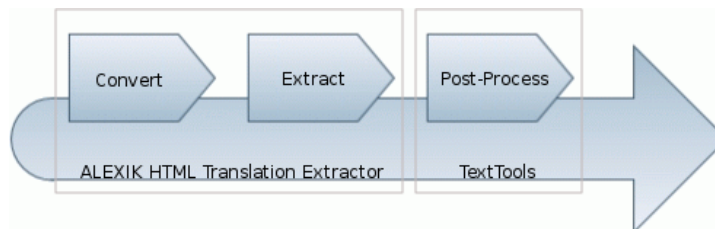


Figure 4.6: Translation Data Processing Pipeline

driver class²⁸ has been used. Section 4.2.3.5 discusses the corresponding database schema used for the dictionary module.

4.2.2.5 log4j

For the purpose of providing a unified way of logging events, as well as for debugging purposes log4j²⁹ has been used. See section 4.2.3.1 for a more detailed description of the setup used.

4.2.2.6 Apache Subversion

In order to be able to synchronize code between developers, Apache Subversion³⁰ has been used, in conjunction with the Tortoise SVN plugin³¹ under Windows and RapidSVN³², as well as the corresponding command-line tools using Linux.

4.2.2.7 Alexik HTML TranslationExtractor

Apart from the tools and technologies used for the main implementation of the e-learning platform Wörterwelt, several additional tools have been developed alongside, in order to provide easy-to-use

²⁸<http://dev.mysql.com/doc/refman/5.0/en/connector-j-reference.html>

²⁹<http://logging.apache.org/log4j/>

³⁰<http://subversion.apache.org/>

³¹<http://tortoisesvn.tigris.org/>

³²<http://www.rapidsvn.org/>

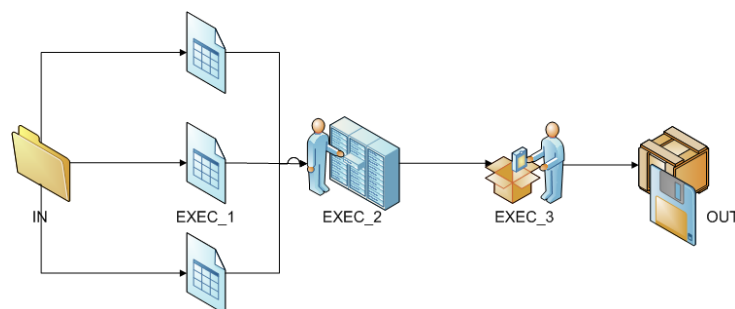


Figure 4.7: Alexik HTML Translation Extractor Processing Pipeline

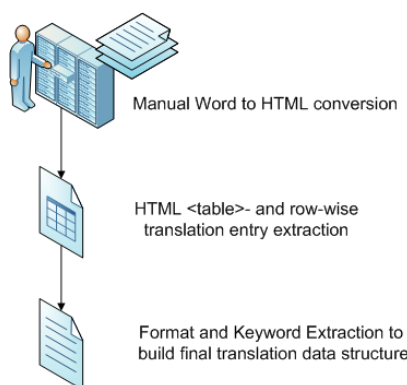


Figure 4.8: Translation Data Transformation Process

means for handling linguistic data, as described in section 4.2.3.2. Translations for the dictionary were originally in the form of Microsoft Word documents, as shown in Figure 4.10. This approach was chosen to provide linguists responsible for aggregating the corresponding data with a well-known tool, for which they did not require additional training. Translation documents were intentionally kept as simple as possible to ensure maximum compatibility with the different versions of Microsoft Word used by the linguists throughout the project. It basically consisted of two columns. Whereas the left-hand side contained the German source term, the right-hand side represented the corresponding translation entry. In order to provide sufficient space for more complex examples and explanatory text, entries may comprise multiple lines. Furthermore, special syntax and formatting rules have been applied to reflect different grammatical meanings and language-specific rules, which will be described in greater detail in section 4.2.3.4. Additionally, for the purpose of providing better means of collaboration between members of the linguistic work groups the Microsoft Word translation documents were split according to the schema shown in Listing 4.1, producing filenames such as TU-BKS.doc.

```
| <alphabet character(s)> - <ISO 3166-1 ALPHA-2 code33>.doc
```

Listing 4.1: Naming Schema of Microsoft Word Translation Documents

To be able to import translation data into the e-learning platform, several processing steps were required. Overall, we required a transformation process that converted Microsoft Word data to structured HTML code, while retaining the formatting and syntax rules described in section 4.2.3.4. These processing activities can basically be divided into three categories, resulting in the *processing pipeline* depicted in Figure 4.6. First of all, translations contained in the Microsoft Word documents must be *converted* to HTML code. Afterwards, significant linguistic data should be *extracted* and separated into *formatted*

³³see ISO [2010]

and *plaintext* versions, while retaining the syntax and formatting rules. Finally, the plaintext should be optionally *post-processed* for “cleanup” purposes.

The main objective behind the tools *ALEXIK HTML Table Extractor* and *TextTools* was to optimize and most importantly to standardize the steps involved in this processing pipeline. Thus, in case of changes to the Microsoft Word translation documents, a unified set of tools would be available. Whereas the *Alexik HTML Translation Extractor* covers the first two steps of the processing pipeline, the *TextTools* presented in section 4.2.2.8, provide the required functionality for the third and final post-processing step. Figure 4.6 depicts this share of functionality to satisfy the processing activities.

Unfortunately, from experience Microsoft Word’s HTML export functionality does not produce source code that can always be used efficiently without any further transformation and optimization effort. As a result, the *Alexik HTML Translation Extractor*³⁴ has been developed to extract translation data from an arbitrary amount of HTML files and import them into the respective database. As shown in Figure 4.8, the overall translation transformation process can be broken down into three basic steps. First of all, the translation documents must be manually exported as HTML using the built-in export function. Secondly, the HTML table contained must be extracted and any unnecessary tags, such as empty or illegally nested tags removed. Using row-wise extraction, the resulting translation data can then be parsed for existing formatting and syntax rules. Finally, based on these rules, the essential content can be identified and extracted. The final outcome of this process is represented by a multidimensional array containing three fields for each translation entry, as shown in Listing 4.2, which can then be imported into the database.

```

1 array (
2   row1 ⇒ array(1 ⇒ HTML, 2 ⇒ plain-text, 3 ⇒ keyword(s)),
3   row2 ⇒ ...,
4   ...
5   rown ⇒ ...
6 )

```

Listing 4.2: Outcome of Translation Data Transformation Process

The processing activities contained in the ALEXIK HTML Translation Extractor have been designed as a pipeline, as depicted in Figure 4.7, which includes the following steps:

1. *IN*: consecutively load source files contained in the current language folder.
2. *EXEC_1*: assemble internal data structure (Listing 4.2) by extracting significant translation data, while automatically removing unnecessary markup code.
3. *EXEC_2*: identify and extract formatting rules and keywords, thus refining existing data.
4. *EXEC_3*: import data to database while automatically checking for existing entries.
5. *OUT*: display statistical output and continue with next file in queue determined in step *IN* for current language, or quit if there are no files left to process.

In order to automate this transformation process to the highest possible degree, special folder structures have been introduced, as depicted in Figure 4.9. For every translation language a separate folder named after its corresponding ISO 3166-1 ALPHA-2 code was created ISO [2010]. The respective translation files were then placed inside their parent language folder. The language folders themselves were placed inside a root container named *data*. The Alexik HTML Translation Extractor then parsed this root folder for available translation languages and processed them sequentially.

³⁴<http://www.kerstner.at/alexikhtmlte/>

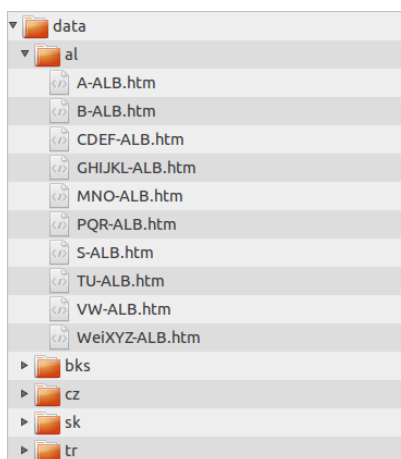


Figure 4.9: Folder Structure used by ALEXIK HTML Translation Extractor

4.2.2.8 TextTools

TextTools³⁵ is a command line based tool that provides a set of handy general-, as well as special purpose text manipulation functions. The main idea behind this tool was to provide easy but yet sophisticated means to efficiently process large sets of plaintext data. It is written in C++, based on Microsoft's .Net Framework³⁶. Hereby, special attention was given to the choice of the data structures, as well as algorithms used in order to provide a fast and reliable framework. Furthermore, in order to chain commands, the *Hook* design pattern by Schmaranz [2004] has been implemented, which is a modified version of the *Command* pattern proposed by Gamma et al. [1994]. Hereby, the Hooks are generated by a *Product Factory* Schmaranz [2004].

As depicted by Figure 4.11, the internal processing activities are designed as a pipeline, which consists of the following basic steps:

1. *IN*: Load source files specified.
2. *EXEC_1*: Assemble internal data structure (linked list) based on content from files loaded in step *IN*.
3. *EXEC_2*: Carry out actions specified by the command line switches, while paying attention to possible ordering of these commands.
4. *OUT_1*: Write (interim) results to (interim) output file(s) specified, or respective default file if none specified.
5. *OUT_2*: Display statistical output and quit.

Apart from general purpose functionality, such as replacing and extracting data, it also provides a selection of special purpose functions, such as *extraction*, *rotation*, *sort* and the *removal of duplicates*. Hereby, especially the sort functionality has proven to be very helpful, due to the fact that linguistic data contained in Word documents did not have to be imported into a spreadsheet application, such as Excel, to actually sort it. Hence, once the Microsoft Word data was exported as plaintext, it could then be sorted and merged accordingly using TextTools. Appendix B presents a more detailed description of the functionality provided by TextTools.

³⁵<http://www.kerstner.at/texttools/>

³⁶<http://msdn.microsoft.com/en-us/netframework/default.aspx>

A, a	
der Aal (= ein Fisch), die Aale	úhoř M (ryba), ~i
das Aas , die ~e (= ein toter Tierkörper)	mřšina F, mřšiny (= tělo mrtvého zvířete)
ab heute	ode dneška
ab und zu (manchmal) (~ gehe ich schwimmen.)	tu a tam (někdy) (~ chodím plovat.)
ab bau en : Sie baute die Ritterburg ab ; hat abgebaut	zbořit : <i>Zbořila</i> rytířský hrad.
ab bei ßen : Er biss die Wurst ab ; hat sie abgebissen	ukousnout : <i>Ukousl</i> salám.
ab bie gen : Das Auto bog nach links ab ; ist abgebogen	odbočit : Auto <i>odbočilo</i> doleva.
die Ab bil dung , die ~en	vzdělání N, ~
ab bre chen : Er brach den Ast ab ; hat abgebrochen	zlomit : <i>Zlomil</i> větev.
der Ab bruch , die ~brüche; der Abbruch des Spiels	přerušení N, ~; přerušení hry
das Abc	abeceda F, abecedy
der Abend , die ~e	večer M, ~y
am Abend (~ gehen wir heim.)	večer (~ jdeme domů.)
das Abend es sen , die ~	večeře F, ~
das Abend kleid , die ~er	večerní šaty M Pl.
Abends : <i>Eines Abends</i> kam die Katze nach Hause zurück	večer : <i>Jednoho večera</i> se kočka vrátila domů.
abends : Sie kam immer abends um 20 Uhr an.	večer : <i>Přišla vždy v osm hodin večer.</i>
der Abend spa zier gang , die ~gänge	večerní procházka F, ~ procházky
das Aben teu er , die ~	dobrodružství N, ~
das Aben teu er buch , die ~bücher	dobrodružná kniha F, dobrodružné knihy
der Aben teu er ro man , die ~e	dobrodružný román M, dobrodružné romány
aber : Ich will ~ nicht!	ale : Já ~ nechci!
ab fah ren : Du fährst heute nach Wien ab ; er fuhr ab ; ist abgefahren	odjet : <i>Dnes odjždíš</i> do Vídně; <i>odjel</i>

Figure 4.10: Excerpt from a German-Czech Microsoft Word Translation Document

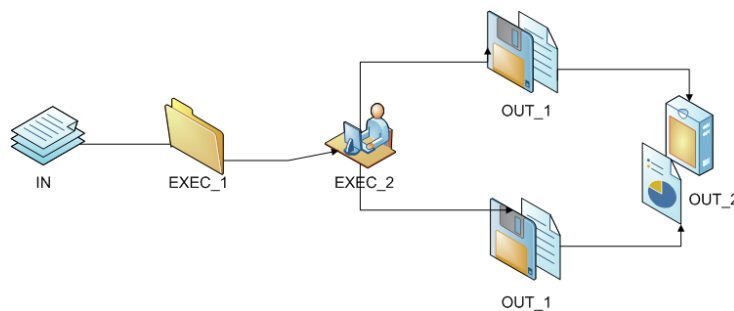


Figure 4.11: TextTools Processing Pipeline

4.2.2.9 phpMyAdmin

As already stated in section 4.2.1, one of the main goals for the dictionary module was to provide an administration interface for unified data management. Instead of developing an additional backend application, *phpMyAdmin* was chosen as the primary dictionary data management tool, since it provided more than the necessary functionality required. Furthermore, with regard to the exercises module described in Section 4.3, using *phpMyAdmin* could also be used to manage the data for the exercises. Thus, overall *phpMyAdmin* offered an elegant way overall to outsource functionality to a well-known, commonly-used, heavily tested and yet freely available database management backend.

4.2.3 Implementation

Prior to the beginning of the actual implementation, the tools and technologies described in section 4.2.2 had to be acquired to set up the development environment. Due to the fact that we wanted to actively test and promote platform independence, development was done under Windows XP, as well as Gentoo Linux³⁷, using different browsers and versions. Consequently, the development environment chosen should represent a self-contained distribution package, that could easily be transferred between different

³⁷<http://www.gentoo.org/>

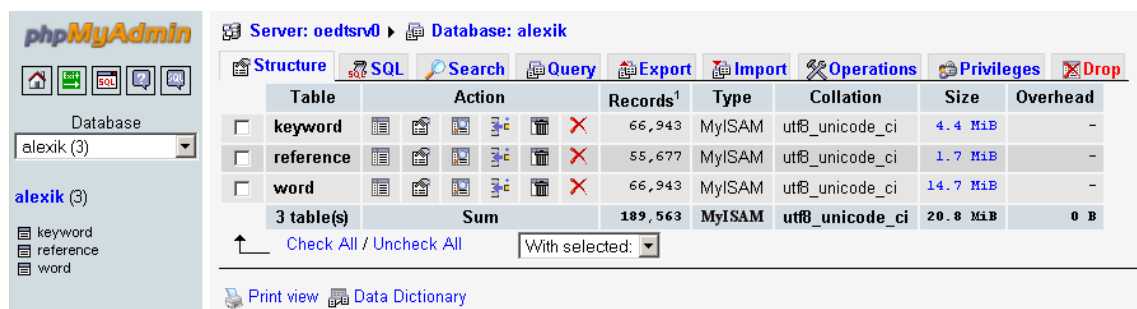


Figure 4.12: Unified Data Administration Backend Tool *phpMyAdmin*

PC setups.

4.2.3.1 Development Environment Setup

Choosing an appropriate development environment heavily depends on the technologies and tools used for the actual implementation. Thus, based on the ZK Ajax framework described in section 4.2.2.1 and the underlying Java Servlet Pages representing the primary software package configuration chosen, an Integrated Development Environment (IDE) was required that supported the development of Java-based web applications. Furthermore, it should provide means to manage web server instances, run unit tests and integrate subversion functionality to enable revision control amongst developers. Finally, since different operating systems were to be used for the development to actively promote testing of Wörterwelt's platform independence, the IDE itself should be supported by these setups. Although there exists a vast range of different IDEs for developing Java-based web applications, such as Eclipse and JDeveloper, *NetBeans IDE*³⁸ was chosen due to its great selection of plugins, flexibility, cross-platform support, as well as previous positive experiences.

First of all, in order to be able to compile and run Java code, the Java Development Kit (JDK) version 1.4 and higher was installed, followed by the setup of the *J2EE*³⁹ Apache Tomcat server described in section 4.2.2.2. Since the default settings were used for the development environment, Apache Tomcat was listening on port 8080 for incoming requests: *http://localhost:8080/*.

Apart from the servlet-container, two additional plugins were required, *MySQL Connector/J* and *log4j* as described in section 4.2.2.5. Plugins for Apache Tomcat can be installed using two different approaches, either *globally* for all projects, or *project-specific*. In this setup, it was chosen to include required plugins in the project source itself, in order to be able to distribute them together with the final implementation, thus keeping end-users from having to download and install them manually. The MySQL Connector/J JDBC driver plugin is required for the dictionary module to be able communicate with the MySQL server, whereas *log4j* has been used for the purpose of providing a unified way of logging events, as well as for debugging activities.

To host Wörterwelt's linguistic data, the MySQL server was installed. Additionally, *phpMyAdmin* was set up as the primary data management tool. Figure 4.12 depicts the database setup chosen for the dictionary module.

Finally, the IDE was set up. Apart from the core installation, two additional plugins were required to satisfy the prerequisites stated above, *SVN* and *JUnit*.

³⁸<http://www.netbeans.org/>

³⁹Java Platform Enterprise Edition, see <http://www.oracle.com/technetwork/java/javasee/overview/index.html>

4.2.3.2 Requirements

Based on the overall aims of the e-learning platform described in section 4.1, as well as the dictionary module's goals and tasks described in section 4.2.1, a more detailed description of the technical and functional requirements needed to be made prior to the beginning of the implementation. Hereby, the technical requirements can be best understood by looking at the basic flow of events during a dictionary *search request*. Initially an Ajax based *POST request* is sent to the server, including the search query as well as additional information, such as the source language and the translation language(s) selected through the UI. Once received, the server interprets the search query request, which may include wild-card characters to perform extended searches. Based on the search query specified, matching entries are retrieved from the database for all the various translation languages. Following that, the server assembles and sends a *XML response*, containing the result payload data. Once the client receives the XML response through a *callback* facility, it accordingly updates the user interface.

Thus, the dictionary module pursues three main objectives. First, it provides a *rich, desktop-like UI*, that, using Ajax technology enables a responsive user experience without the need to reload the frontend presentation layer on each request, while still relying on common HTTP technology. Secondly, it *encapsulates the business logic* entirely on the server side, thus minimizing the required client side code, and finally, it uses a *unified, XML-based communication protocol* to exchange data between the client and the server side.

Apart from the technical requirements, the dictionary module's functional requirements can be broken down again into three separate categories. Firstly, users are required to select one *source* and many arbitrary *translation languages*, on which the search query will be based. Secondly, through an *auto-completing search box*, users are able to submit their search queries, which will finally be rendered as rows in a *dynamic results grid*.

From the functional point of view, the dictionary module serves as the central point for searching words and phrases in all languages available to the e-learning platform. Consequently, the search functionality's scope of operation is two-fold. First, it should be possible to search for a word/phrase in a particular language and get all matching items. This represents the basic case of querying entries in the dictionary (*1:1 search*). Additionally, if requested by the user, any corresponding entries for the remaining translation languages should also be returned. This represents the dictionary's translation functionality (*1:n search*). In short, the basic mode of operation is to search for an entry in the source language, while automatically retrieving corresponding results of all selected translation languages.

The search facility itself should be *case-insensitive* and support *wild-card characters* for more complex search queries. Search terms entered should be automatically expanded through an *auto-completing search box*. Search results should be displayed row-wise using a *dynamic grid* that provides the ability to select entries through a context menu. Each column should represent a certain language, containing its corresponding result entries. In addition, users should be able to change the columns' order through an easy-to-use *drag&drop* mechanism.

Moreover, as Schwarz et al. [1983] found out, inexperienced users especially prefer *paging representations* instead of long, scrollable text to better grasp the contents. Furthermore, Piolat et al. [1997] demonstrated that users who navigated through data via pages built better mental models and thus, are able to locate relevant information more easily. Apart from these psychological advantages of paging in comparison to scrolling, optimizations concerning the processing of search queries could be achieved, too.

Certainly then, the dynamic grid should also support *paging* of results. Using this mechanism, a limited number of entries were to be displayed per result page, limiting the grid's visual space requirements, and simultaneously reducing the cost of processing search queries. Together with a mandatory result navigation, users should then be able to *browse through the result set*. Although Bernard et al. [2001] found out that when using search based results a limit of fifty links per page was preferred by most users,

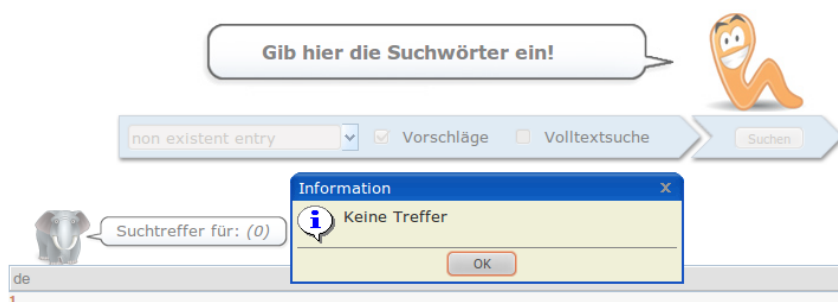


Figure 4.13: Dictionary Module User Interface No Results Message

a default setting of *ten entries* was chosen. This decision is based on the idea that when using multiple translation languages, the presented columns might overwhelm especially younger, inexperienced users. In fact, most of today's popular search engines such as Google and Yahoo use a setting of around ten entries. The overall goal then was to present result sets in a number of information pieces Bernard et al. [2001], that are perceived to be manageable, valuable and easily found.

4.2.3.3 User Interface

Based on the requirements presented in section 4.2.3.2, the design of the graphical user interface pursued three main objectives, with respect to the level of usability for the target user group. First of all, the functionality provided by the dictionary module should be *easily accessible* and *easy to use*. Furthermore, the final results table should be *flexible* in the selection and ordering of the translation languages in corresponding columns. Finally, the UI should process commands in a timely fashion and also, through Ajax technology act like a desktop application, leading to a high *responsiveness*.

Figure 4.14 shows the final version of the user interface developed for the dictionary module. It is separated into three main areas, with regard to the three functional requirements specified in section 4.2.3.2. The first functional block represents the *language selection*. On the left, users are able to select the source language; on the right they can arbitrarily select from available translation languages. When a source language is selected, that language is disabled for translation entries on the right-hand side. The vice versa occurs when selecting translation languages.

Based on the language selection, the results table is then updated to reflect the changes made. By default, the source language is displayed as the first column on the left and translation languages are appended consecutively according to the order of selection on the right. The second functional component, the *auto-completing search box* is updated too, depending on the source language selected. Apart from the auto-completion facility, the search box also enables users to do *full-text searches*. When this option is activated all translation entries are returned containing the search query term, otherwise, a *keyword search* is done as explained in section 4.2.3.5. In the case that no matching entry could be found, the message shown in Figure 4.13 will be displayed.

The third and final functional block is represented by the *dynamic results table*. Based on the results returned by a search query, the results table is updated respectively. It automatically splits the result set into pages, which limits the amount of data returned for a search query and simultaneously increases the application's overall responsiveness. Users are then able to browse through the result set by using the navigation bar attached to the bottom of the table.

The grid's columns can be reordered using the drag&drop facilities provided. For instance, the Turkish translation column shown in Figure 4.14 could be easily placed to the left of the Slovakian. Additionally, translation languages can be added or removed from the results table at any time using the aforementioned language selection block. Finally, the source language can also be changed at any time

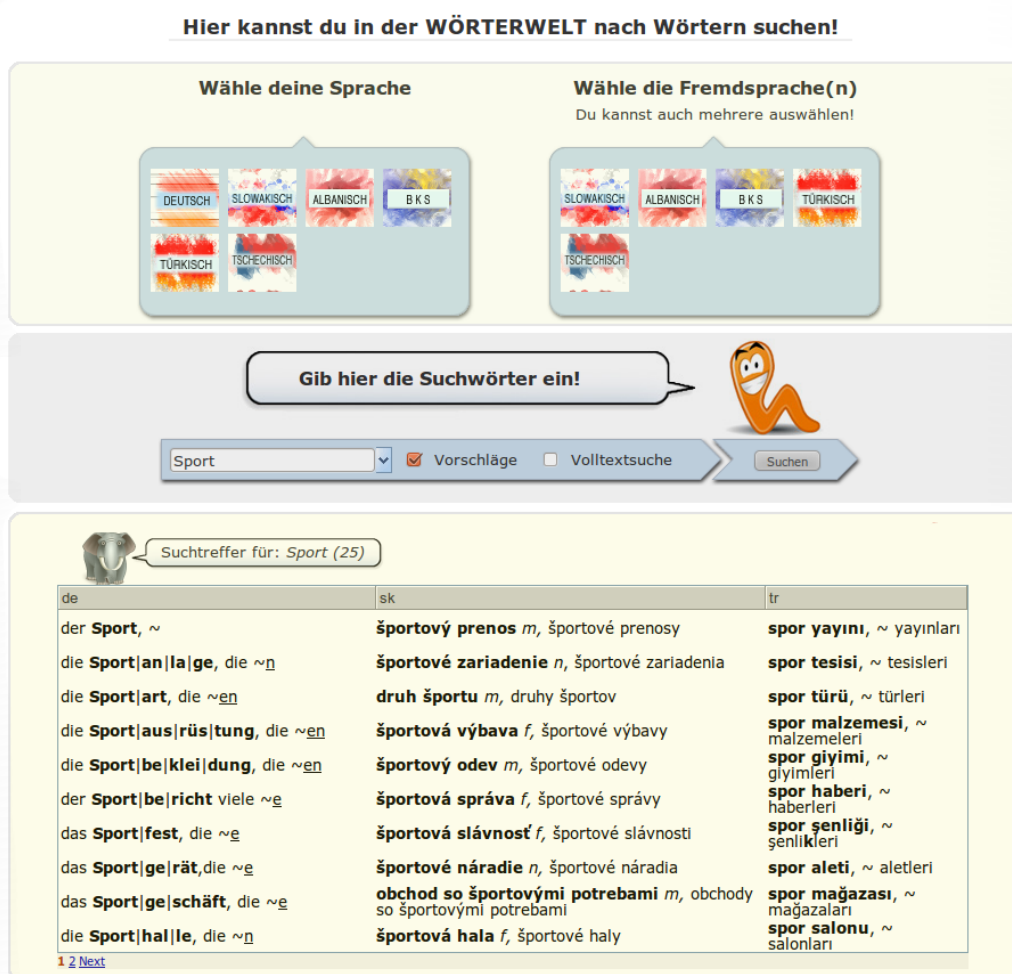


Figure 4.14: Dictionary Module User Interface Functionality Overview

during a user's session.

4.2.3.4 Data Schema

In order to achieve a slim and yet powerful dictionary data format, special formatting and syntax rules have been developed and applied during the generation of the language corpus Muhr [2010]. This section gives a brief overview of the most important and most commonly used syntax and formatting rules. Based on this description, the dictionary data schema developed for the practical implementation will be presented in greater detail. Further information on the specifics of the syntax and formatting definitions can be found at the project homepage.

```
1 German source entry → Translation entry
```

Listing 4.3: Dictionary Data Syntax Rule Schema

Overall, dictionary data follows a consistent schema, as shown in Listing 4.3. Entries basically consist of two parts, separated by the language delimiter symbol →. Hereby, the left-hand side of the expression represents the German source entry. The corresponding translation entry is displayed. For the sake of consistency, the examples provided below exclusively show German to Czech translation entries.

1. Symbol →

This symbol serves as delimiter for separating German source entries from corresponding translations.

Example:

1 das **Bei**|spiel, die ~e → **příklad** M, příklady

Listing 4.4: Example using the → symbol

2. Symbol |

This symbol separates words into their corresponding syllables. Example:

1 der **Sach**|**un**|**ter**|**richt** → **prvouka** F

Listing 4.5: Example using the | symbol

3. Symbol ~

The ~ symbol denotes the start of the noun's plural form. There are several different scenarios possible in this case:

- if the plural- matches the singular form only the plural's article together with the ~ symbol will be shown (Listing 4.6).
- singular and plural forms differ:
 - if only the ending characters (syllable) are different the plural form of the syllable will be shown (Listing 4.7).
 - otherwise, instead of the ~ symbol the entire word will be shown with the differential characters marked in bold letters (Listing 4.8).
- if no plural form exists no article and therefore no ~ will be specified (Listing 4.9).

1 der **Sä**|**ge**|**ar**|**bei**|**ter**, die ~ → **pracovník na pile** M, **pracovníci** ~

Listing 4.6: Example using the | symbol where plural matches singular form

1 das **Au**|**to**, die ~s → **auto** N, **auta**

Listing 4.7: Example using the | symbol where plural form partially differs from singular form

1 der **Saft**, die **Säfte** → **šťáva** F, **šťávy**

Listing 4.8: Example using the | symbol where plural differs singular form

1 das **Saat**|**gut** → **osivo** N

Listing 4.9: Example using the | symbol where no plural form exists

4. Symbol ()

Brackets enclose homonyms. Example:

1 das **Steu**|er, die ~ (=Lenkrad) → **volant** M, y; **kormidlo** N, kormidla

Listing 4.10: Example using the () symbol

5. Symbol :

Colons denote the start of descriptive examples, whereas multiple examples are separated by semicolons (;). Example:

1 **aus**|räu|men: Er *räumt*e das Regal *aus*; *hat* *ausgeräumt* → **vyklidit**: *Vyklidil* polici

Listing 4.11: Example using the : symbol

6. Symbol :=<=:

This symbol denotes language related differences. Example:

1 **fass**|te nach: **D** :=<=: **A** greifen nach: Jeder **fasste nach** der Hand des anderen.

Listing 4.12: Example using the :=<=: symbol

7. Formatting Rule **bold**

Boldfaced words primarily define nouns, or parts of it. Furthermore, boldfaced words also represent an entry's *main terms* that will be used for the keyword search described in section 4.2.3.5. Example:

1 **ab** heute → **ode** dneška

Listing 4.13: Example using the **bold** Formatting Rule

8. Formatting Rule underline

Underlined words/characters denote a term's plural form that differs from the singular form, as well as highlights tense differences, as shown in Listing 4.11. Example:

1 die **Auf**|schrift, die ~en → **nápis** M, ~y

Listing 4.14: Example using the underline Formatting Rule

9. Formatting Rule *italics*

This formatting rule marks additional significant differences, such as homonym relations, as well as tense differences. Example:

1 *das* **Steu**|er, die ~ (=Lenkrad), aber: *die* **Steuer**, die ~n (=Abgaben)

Listing 4.15: Example using the *italics* Formatting Rule

It must be born in mind that the enumeration provided above merely serves as an overview of the most important syntax and formatting rules developed and used by the linguists throughout the ALEXIK project. Various additional aspects, such as the order of informative and significant words pre-/succeeding main terms are not covered in this thesis. The official documentation on the project's homepage is available for further information.

A dictionary data schema was designed corresponding to these linguistic rules. As described in Section 4.2.2.7, each dictionary entry consists of three distinct parts, each having a specific format and purpose, the *HTML formatted representation*, the *plaintext version* and the *keywords* used for the search.

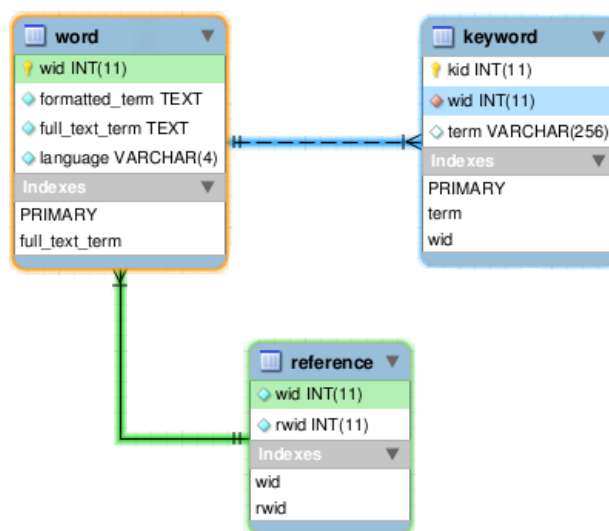


Figure 4.15: Dictionary Module Database Schema

Whereas no formatting and syntax rules have been applied to the plaintext version and keywords used for the search, special attention has been drawn on the entry's HTML formatted presentation. The schema shown in Listing 4.16 was defined to enable mapping of syntax and formatting rules used in the original Microsoft Word translation documents described in Section 4.2.2.7 to the corresponding HTML markup code used by the dictionary module. For this each dictionary entry must be enclosed in a `<p>`-tag using the CSS-class *record*. Words contained in the entry must further be embedded into ``-tags, identified by the *word* CSS-class. Inside these word entries, the following additional styling-tags are allowed: ``, `<i>` and `<u>`. The special purpose directive `` denotes so-called *main terms* that will be used as keywords for the keyword-based search presented in section 4.2.3.5.

```

1 <p class="record">
2   <span class="word">das <b>Bei</b>|<b>spiel</b>, die ~<u>e</u></span>
3 </p>

```

Listing 4.16: Example of a Formatted Dictionary Entry

Although nesting of words as well as definition of additional styles using the `style` directive is strictly prohibited, the nesting of the aforementioned allowed styling tags inside *word* entities is possible. By using this schema, dictionary entries can be easily traversed using the corresponding DOM functionality.

4.2.3.5 Database Schema

A well-designed database schema represents one of the key factors for a well-performing, flexible software application. With respect to the dictionary data schema, this section presents the database schema used by the dictionary module as depicted in Figure 4.15, which basically consists of three tables: *Word*, *Reference* and *Keyword*.

Word

The table *Word* contains available translation entries, identified by a unique word id (*Word.wid*) and the corresponding language (*Word.language*), specified using ISO 3166-1 ALPHA-2 abbreviation format ISO [2010]. *Word*-entries are not limited to single terms, in fact, most entries represent complete phrases.

Furthermore, in order to be able to visualize the formatting and syntax rules defined in Section 4.2.3.4 as well as to do full text searches, two separate columns have been introduced: *Word.formatted_term* and *Word.full_text_term*. Whereas *Word.formatted_term* contains the nicely formatted entries used for the presentation layer, as for instance shown in Listing 4.16, *Word.full_text_term* represents the plaintext counterpart that can be used for full text searches.

Keyword

The *Keyword* table serves as an optimized listing of *main terms* extracted from the corresponding *Word* entries. It can be seen as a lookup table for *words* representing a *many-to-one relationship* to the *Word* table that connects arbitrarily many keyword-terms (*Keyword.term*) to the corresponding *Word* through a unique keyword id (*Keyword.kid*).

As previously mentioned in Section 4.2.3.4, keywords represent a *Word's main terms* that are specially formatted using ``-tags. In case no main terms exist, the entire full text version of the corresponding *Word* will be used as keyword, resulting in the relation shown in Listing 4.17. Thus, the worst case scenario possible is a keyword entry is identical to the *Word's* full text term. To clarify the idea behind this optimization, the *keyword* extracted from the *Word* entry shown in Listing 4.16 is *Beispiel*.

```
1 {keyword1, ..., keywordn} ⊆ Word.full_text_term
```

Listing 4.17: Keyword Word Relation

Using this approach, a significant amount of costly full-text searching is eliminated by an optimized keyword-based lookup table, which additionally is useful for the auto-completion functionality described in Section 4.2.3.2.

Reference

The *Reference* table represents the *one-to-many relationship* between *Word* translation entries. It consists of two columns, *Reference.wid* and *Reference.rwid*. Whereas *Reference.wid* represents the German source *Word*, *Reference.rwid* references available translations entries. This design minimizes the effort required to manage many-to-many relations, since German source entries always serve as the primary lookup value to determine available translations.

4.2.3.6 Architecture

With ZK's server-centric architecture described in Section 4.2.2 in mind, this section presents the software architecture developed for the dictionary module, which can be separated into two main parts, represented by the Java packages

- `edu.iicm.alexik.dict`
- `org.alexik.dict`

The `edu.iicm.alexik.dict` package depicted in Figure 4.17 represents the database abstraction layer. The `org.alexik.dict` shown in Figure 4.18 incorporates the business logic and the presentation layer functionality. Prior to discussing the business logic, the database abstraction layer will be presented. Here dictionary entries are represented as collections of `Words` that can be managed through the corresponding object-relational mapping (ORM) abstraction layer `WordDAO`. According to the requirements specified in section 4.2.3.2, a MySQL specific implementation of the ORM layer has been introduced, called `WordDAOImpl`. The `WordDAO` entities are generated by the `WordDAOFactory`, implemented using the Product Factory design pattern Schmaranz [2004]. Database entries represented

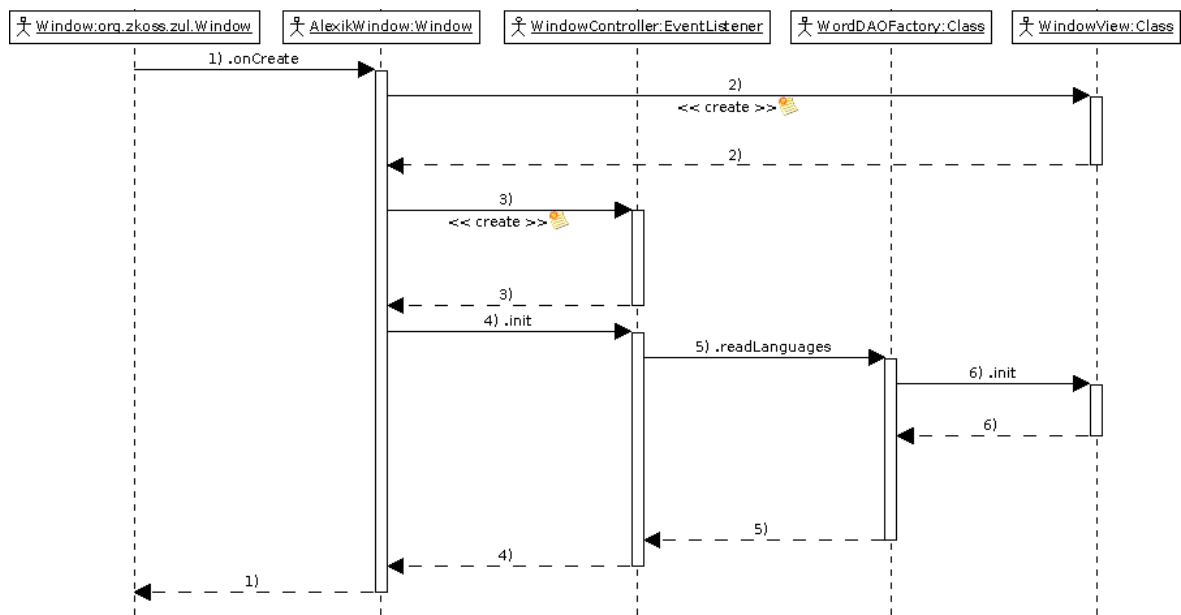


Figure 4.16: Dictionary Module Initial Page Load

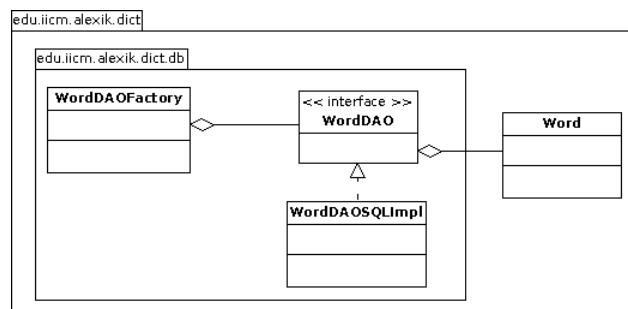


Figure 4.17: Dictionary Module Database Abstraction Layer

by the corresponding `Word` objects can then be retrieved through the search functionality provided by `WordDAO`. Apart from the search facilities, the ORM abstraction layer also provides means for requesting available translation.

The business logic layer shown in Figure 4.18 is based on the model-view-controller design pattern (MVC, Krasner and Pope [1988], Gamma et al. [1994]), which conforms to the design guidelines for ZK based applications Potix Corp. [2010d]. It is built upon `AlexikWindow`, an abstraction of ZK’s `Window` base class, which can be regarded as the main container for ZK components, such as the dictionary module’s dynamic grid, represented by the respective implementation of the `ResultsTable` interface. Based on ZK’s architecture, `AlexikWindow` can be managed via the backend side through manipulation of the corresponding Java classes, as well as through the corresponding scripting-codes on the client side, allowing dynamic changes to the presentation layer from the both end points of the dictionary module.

Thus, `AlexikWindow` can be seen as the primary container for including presentation layer functionality. Applying the MVC pattern, it is composed of `WindowController` and `WindowView`. The missing model in this pattern is represented by the appropriate implementation of the `ResultsTable` interface, which contains the majority of the module’s business logic. As depicted in Figure 4.18, two different implementations of the `ResultsTable` interface have been developed, represented by the Java packages

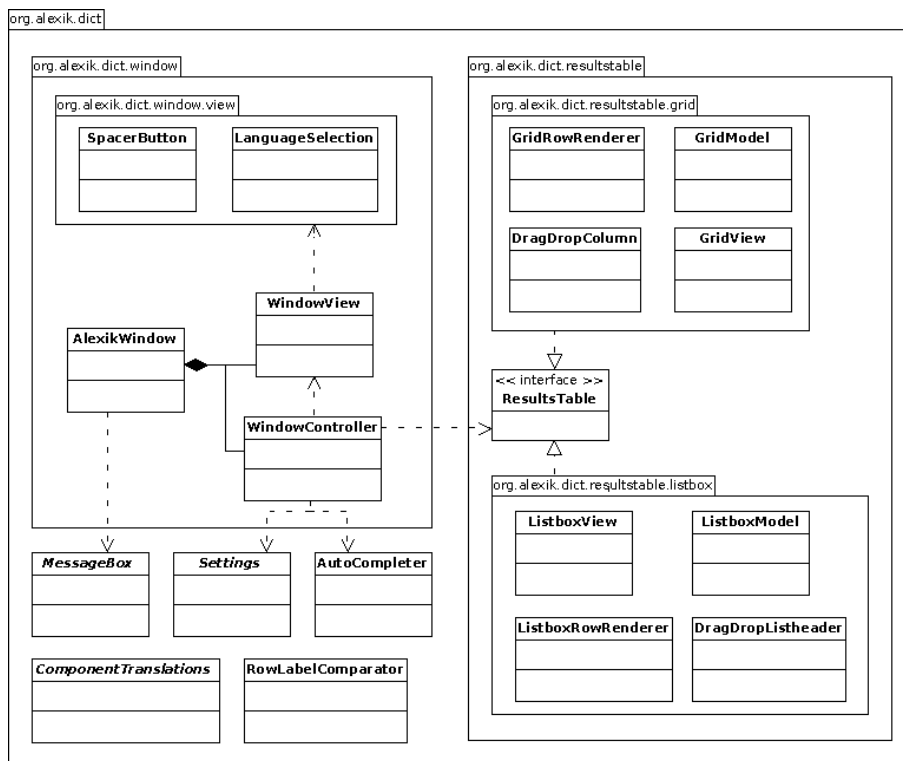


Figure 4.18: Dictionary Module Business Logic Layer Architecture

- `org.alexik.dict.resultstable.grid`
- `org.alexik.dict.resultstable.listbox`

Although both implementations provide the same functionality, the version using ZK's `Listbox` was chosen as primary development foundation, due to the better column presentation facilities provided through the corresponding renderer object. `ResultsTable`'s implementations have also been designed according to the MVC pattern. So in the case of the `Listbox` implementation, the corresponding model and view are represented by `ListboxModel` and `ListboxView`. The controller functionality has been incorporated in the respective parent container, `AlexikWindow`. Additionally, in order to minimize the view's footprint, the `ListboxRowRenderer` has been introduced to handle the result table's ordering of columns through drag&drop events, as well as providing the paging-mechanism described in section 4.2.1.

As previously described in section 4.2.2.1, ZK-based applications differentiate between two types of requests, the *initial page load* and *spontaneous, event-based Ajax update-requests*. Whereas Figure 4.16 depicts the basic flow of events during initial page loads, Figure 4.19 shows the fundamental step involved in handling spontaneous update events, once the application has been fully initialized. Respectively, during the initial page load of the dictionary module several steps are required:

1. creating a new *AlexikWindow* instance
2. initializing the corresponding MVC components and dependencies
3. determining available languages via database abstraction layer
4. updating the view accordingly

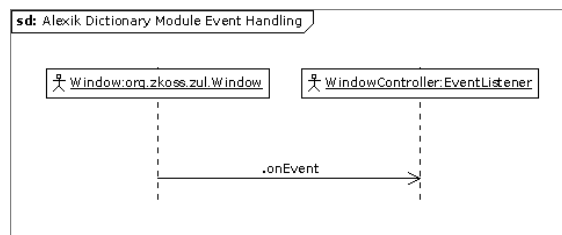


Figure 4.19: Dictionary Module Event Handling

On the other hand, Figure 4.19 which illustrates the processing of events. Instead of showing the entire range of event types and their corresponding actions, only the base event *onEvent* is included in the sequence diagram. The total set of events handled by the `WindowController` includes:

1. update of source language
2. update of translation language(s)
3. *onPaging*
4. *onRightClick*
5. *onSelect*
6. *onDrop*
7. *onCheck*

Here the first two events provide means for changing the source language and setting an arbitrary number of available translation languages. Changing the source language automatically triggers the `AutoCompleter` to update the search language used for the auto-completing search box. The *onPaging*-event handles the functionality of browsing through the result set of a search query. Users are able to either select specific pages, or leaf through the result set using the *forward* and *back* links. The *onRightClick*-event is used to load a context menu, based on the selection of a specific term from the results list. In future versions, it should provide additional functionality, such as using the selected item as a template for search queries or invoking corresponding exercises from the exercises module described in section 4.3.

The *onSelect*-event was introduced to handle the process of selecting specific entries from the results table. In future versions, this event might also invoke a contextual menu incorporating further functionality, such as audio playback of the term selected. The *onDrop*-event is used to handle drag&drop actions when users drag available translation languages from the results grid and drop them at different horizontal locations to reorder the grid, as described in section 4.2.1. Finally, the *onCheck*-event determines whether the full text search or the keyword search⁴⁰ should be used and updates the presentation layer accordingly.

In conformance to the MVC design pattern, once an event-type has been identified, it is delegated to the respective methods. Unknown events will be logged⁴¹, as well as displayed to the user through the `MessageBox`. The update process itself is a sequential combination of retrieving and assembling data through the model and updating the view correspondingly.

⁴⁰see Section 4.2.3.5

⁴¹see Section 4.2.2.5

4.2.3.7 Standards Used

The set of standards used for the dictionary module basically fall into two main categories: *Data management* and the *presentation layer*. Access to data is provided using SQL through the database abstraction layer described in Section 4.2.3.6, as well as the administration backend *phpMyAdmin* presented in Section 4.2.2. SQL statements used throughout the dictionary module were based on the ISO/IEC 9075-14:2008 standard ISO [2008]. Furthermore, to enable internationalisation (i18n), *UTF-8* was chosen as the default character encoding standard for dictionary data Yergeau [2003].

The presentation layer comprises an entire set of standards, including XML, XHTML, CSS and JavaScript, that is either automatically generated by ZK's Loader described in Section 4.2.2, or injected through the corresponding frontend markup code. Consequently, the exact standards used for the presentation layer heavily depends on the version of ZK deployed, as it serves as a facade for generating corresponding client side code. ZK uses XHTML 1.0 Transitional by default, resulting in a document type definition (DTD) shown in Listing 4.18.

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">

```

Listing 4.18: ZK Default Document Type Definition

Unfortunately, no exact information could be found on the JavaScript version used by ZK. Thus, determining the corresponding standards version is rather tricky, although there are basically only two possibilities, the fourth or the fifth edition of the ECMA-262 standard. JavaScript, which actually is a synonym for the *ECMAScript scripting language*, is defined through the ECMA-262 standard ECMA [2009]. As of this writing, the standard is in the fifth edition. Whereas the fourth edition has already been approved by the ISO as *ISO/IEC 16262* ISO [2002], the fifth edition is still pending for approval as a replacement edition.

Moreover, it is interesting to note that ZK itself uses additional client side frameworks, such as *script.aculo.us*, making it even more difficult to determine common standards versions. Nevertheless, an educated guess would be that one of the latest versions are implemented, such as JavaScript 1.5 and greater. This also conforms with Mozilla Developer Network [2010b], stating that browsers that do not support at least JavaScript 1.5 are very rare today, since it has already introduced back in 1999. Thus, in order to profit from the vast range of enhancements since JavaScript version 1.5, a more current version will most likely be used. For instance, as of this writing, Mozilla Firefox incorporates JavaScript version 1.8.2 Mozilla Developer Network [2010a].

The same problem applies to the CSS code generated by ZK. Furthermore, due to the fact that styling code can also be referenced from within ZUL presentation layer files, a mixture of different standards versions is possible. To circumvent this possibility, styling code had to conform to the CSS 2.1 specification W3C [2002], with forward compatibility to CSS 3 W3C [2001] in mind.

The remainder of this chapter is dedicated to presenting the second part of the e-learning platform Wörterwelt, the exercises module, thus completing the functionality required.

4.3 Exercises Module

The exercises module represents the second integral part of the e-learning platform Wörterwelt.

Apart from the dictionary functionality described in section 4.2, the exercises module adds facilities for interactive language learning. Designed and implemented by David Wolf in the course of his Bachelor thesis at the Technical University of Graz, it incorporates a variety of *exercise base types*, including cloze text, crossword, drag and drop, hotspots and multiple-choice. Due to the flexible architecture described

in Section 4.2.3.6, further exercise types can be easily added to the system. Prior to the presentation of the actual implementation in Section 4.3.3, Section 4.3.1 presents the module's goals and tasks, followed by a discussion of the tools and technologies incorporated. Afterwards, Section 4.3.3.5 presents the module's underlying architecture, whereas Section 4.3.3.6 discusses the standards used for the implementation.

4.3.1 Goals and Tasks

Preceding the actual implementation, the overall goals and tasks of the exercises module were defined. Like the dictionary module, the main goals for the exercises module were deduced from the original project proposal, combined with further user and functional requirement specifications. Based on previous experience developing web applications, additional goals, as well as tasks could be identified. From the technical perspective, most of the main goals collected through the requirement specification process resembled those of the dictionary module described in Section 4.2.1.

Most importantly, the implementation should be based on *common web standards*. Based on an *i18n-enabled presentation layer*, a language independent user interface should be developed. Moreover, a *lightweight and standards-based client-server exchange protocol* should be used. To enable a rich user interface (RIA), an *Ajax-based presentation layer* should be deployed, including interactive language exercises, using a broad range of different exercise types. Additionally, through a *session management layer*, users should be able to progress through exercises contained in a *learning course*.

For a unified exercise management, exercises should be organized in *learning packages*, that are based on a previously defined didactic-relevant exercise topology. Due to the fact that most of the business logic is contained in the implementation of the exercises, a *widget-based client side framework* should be used, in order to *promote rapid development*. These learning packages are Wörterwelt's approach to digital learning objects and meta structures previously described in section 2.2.2.

On the contrary, a *slim server side architecture* should be developed, that would provide the aforementioned session management and database layer functionality. Similar to the dictionary module, the tools and technologies used for the exercises module should be *platform independent*, and support *online* as well as *offline* execution. Respectively, an administration interface for *unified data management* should be provided.

Based on the main goals listed above, the resulting tasks were identified. Again, the same approach as for the dictionary module has been used. Since both modules should be integrated into the e-learning platform, as depicted in Figure 4.1, one of the major objectives was to *identify overlapping tasks*, as well as determine tools and technologies required by both modules. The following list represents the main tasks of the exercise module:

- asynchronous processing of GET and POST requests
- handling of unified data format for client-server communication
- handling of multi-byte character encoded data
- serving template and Ajax based presentation layer to enable rich and responsive user experience, including support for i18n
- providing widget-based, interactive exercises, including feedback facilities using standardized format and architecture
- providing SQL-based data storage

Once the goals and tasks were identified, tools and technologies providing the required functionality could be determined. Section 4.3.2 covers the combination of tools and technologies used for the exercises module in greater detail.

4.3.2 Technology and Tools

The driving technologies for the exercises module are based on the *PHP* server side programming language, as well the feature rich, client side JavaScript framework called *Dojo Toolkit* for standardized interaction between the client and server sides. The following list provides an overview of server side technologies incorporated in the exercises module, the so-called *web-stack*:

- Apache 2.0 Web server⁴²
- MySQL 5 Database server⁴³
- PHP 5 Server-side scripting language⁴⁴

Based on this web-stack, the client side environment has been built, which is comprised of the *Dojo Toolkit*, an Ajax-based JavaScript framework and bundled with a distinct selection of proprietary JavaScript classes, as well plain HTML and CSS code required for the presentation layer. Consequently, the server side code is entirely written in PHP and the client side code is a mixture of several programming, as well as markup languages. Furthermore, as described in Section 4.3.3.3, *JSON* was chosen as the primary client-server data exchange format.

4.3.2.1 Dojo Toolkit

“Over time, the job of the DHTML hacker has changed. We know most of the tricks that we can expect a browser to do, and where there is overlap between browsers, we’ve probably already exploited it. . . just look at the depth and diversity of modules in Dijit and DojoX.”

[Russell [2008]]

JavaScript in its pure form provides web developers with a variety of facilities to turn static HTML content into rich, desktop-like applications Wenz [2007]. Unfortunately, due to its nature as a client side scripting language, browser support and implementation details differ greatly between current vendors.

The root of this problem can be tracked back to the year 1995 when Netscape introduced *LiveScript*, a client side scripting language that resembled the syntax of Sun’s *Java* Wenz [2007]. Due to marketing reasons, the name was soon changed to *JavaScript*. Although Netscape’s implementation of JavaScript was very limited, it became popular very swiftly.

Netscape’s competitor Microsoft also had realized JavaScript’s potential and planned to integrate it into Internet Explorer version 3. As a result of conflicting licenses, they had to name their port of JavaScript *JScript*. From this point onwards, Netscape and Microsoft competed fiercely for dominance in usage shares, resulting in the infamous *browser-wars* Wenz [2007]. By adding JavaScript/JScript features in their proprietary implementations, the compatibility gap kept increasing. Whereas Netscape used the ECMA-262 standard as a template, Microsoft introduced its own standards, such as the DOM Wenz [2007].

Consequently, web developers striving for browser-independent compatible web application were left with the hassle of filling this gap. Due to this fact, a broad range of JavaScript frameworks emerged over the last couple of years that all share a common goal: hide browser-incompatibility hassles by providing browser-independent functionality.

The list of today’s most popular JavaScript frameworks includes for example the Dojo Toolkit, JQuery, Prototype, script.aculo.us Yahoo UI and Google’s Web Toolkit. The final choice for the best

⁴²<http://httpd.apache.org/>

⁴³<http://www.mysql.com/>

⁴⁴<http://www.php.net/>

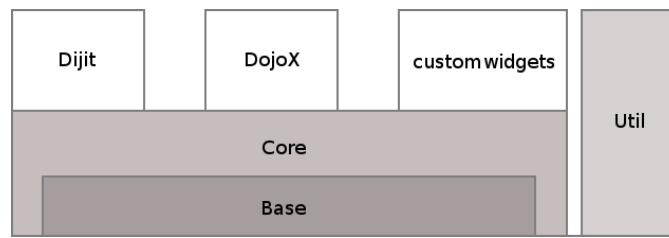


Figure 4.20: Dojo Toolkit Architecture [reproduced from Russell [2008]]

fitting framework heavily depends on the application’s requirements. For instance, some libraries such as *Prototype* are specifically targeted at providing fundamental, browser-independent functionality, whereas *Dojo* or *script.aculo.us* additionally provide rich presentation layer functionality.

This distinction becomes evident by looking at JQuery, which has been split into two separate libraries: *JQuery* and *JQuery UI*. Hereby, *JQuery UI* represents an extension of JQuery, which provides an abstraction layer for low-level interaction and animation, advanced effects and high-level and themeable *widgets* JQuery Project and the JQuery UI Team [2011]. On the other hand, the foundation library *JQuery* itself represents a concise JavaScript library with a very small footprint, specifically targeted at simplifying HTML document traversing, event handling, animation and Ajax interactions The JQuery Project [2011].

Since additional presentation layer functionality was required for the exercises, a library incorporating both feature sets was mandatory. Apart from *JQuery UI*, the *Dojo Toolkit* represents another popular JavaScript framework. As shown in Figure 4.20, it is comprised of five major components: *Base*, *Core*, *Dijit*, *DojoX* and *Util*. Optionally, developers are free to implement *custom widgets*. In the following, these components are going to be discussed in greater detail.

Base

Russell [2008] refers to the *Base* component as an “ultra-compact, highly optimized library that provides the foundation for everything else in the toolkit”. Respectively, if Dojo was an operating system *Base* would represent the *kernel*. Apart from providing developers with facilities to handle Ajax requests, *Base* incorporates a complete packaging system, tools for managing inheritance hierarchies, querying DOM nodes through CSS3 selectors, as well as standardized functions for handling DOM events Russell [2008]. It represents the foundation layer onto which other components are built.

In its function as kernel, *Base* is also responsible for *bootstrapping* Dojo-enabled applications, by including required files identified through the packaging system, detecting the browser’s environment, “smoothing out” browser incompatibilities and finally loading Dojo’s namespace *dojo.**. For as Dojo has been designed as a very loosely coupled combination of different modules, *Base* can be deployed separately. Thus, developers seeking a toolkit to for instance browser-independently handle Array-traversals, are free to only include Dojo’s *Base* component in their applications.

Although *Base* might seem very simple at first, it provides a rich set for many standard operations needed in JavaScript development Russell [2008].

Core

Core serves as an extension of *Base*. It incorporates functionality that has not been deemed universal enough to be included in *Base* Russell [2008]. The following list serves as an overview of additional facilities provided by *Core* Dojo Foundation [2010b]:

- parsing widgets (*dojo.parser.**)
- advanced animation effects (*dojo.fx.**)

- uniform data access layer (*dojo.data.**)
- drag & drop (*dojo.dnd.**)
- internationalization (*dojo.i18n.**)
- localization (*dojo.number.**, *dojo.date.**, ...)
- handling of back-button event (*dojo.back.**)
- managing cookies (*dojo.cookie.**)

Although the boundary between the modules of Core and Base might not be fixed, a differentiation between these two can be made by examining how modules are included in applications Russell [2008]. Since Base is represented by a single file called *dojo.js*, Core modules such as the parser (*dojo.parser.**) must be explicitly included, externally to Base. Consequently, in contrast to functionality provided by the Base's root namespace *dojo.**, Core facilities usually appear in lower-level namespaces, such as *dojo.parser.**.

Whereas Base and Core represent Dojo's foundation providing standardized, browser-independent, everyday functionality, *Dijit*, *DojoX* and *custom widgets* represent further extensions.

Dijit

Dijit, which is short for *Dojo widget* represents an extensive widget library, ranging from calendars to WYSIWYG⁴⁵ editors to progress bars. According to Dojo Foundation [2010b], *Dijit* is Dojo's user interface library that requires Core and resides in its own *dijit.** namespace. By conforming to commonly accepted accessibility standards, such as *ARIA*⁴⁶, it provides developers with flexible and yet powerful, ready-to-use components Russell [2008].

Personal experience has shown that most often little or no effort is required to customize or include *Dijits* in web applications. In case changes to existing *Dijits* are required or a new widgets have to be implemented, developers are free to create custom widgets by either abstracting from an existing templates or implementing the interfaces provided.

Since *Dijit* is built directly upon Core, it inherits the same strong testament to integrity Russell [2008]. Consequently, the same thoroughly tested *building blocks* are used for all newly-developed widgets. will be used. Furthermore, due to *Dijit*'s interface declarations, custom widgets are highly portable. Since they follow the *write-once-use-often* system, they can be included arbitrarily often by using the *dojoType*-tag, as shown in Listing 4.19. Line 1 presents an example for the built-in *TextBox*, whereas Line 2 shows an example for a custom widget, which is available through the namespace *my.custom.**.

```
1 <input dojoType="dijit.form.TextBox" />
2 <input dojoType="my.custom.TextBox" />
```

Listing 4.19: Example for including *Dijits* using the *dojoType* Directive

Due to its vast range of built-in *Dijits*, a finer grained categorization has been introduced. According to Russell [2008], *Dijits* can therefore be divided into three categories: *general purpose*, *layout* and *form widgets*. Whereas general purpose widgets include dialogs and progress bars, the set of layout widgets consists of tab- and border containers, as well as accordion panes. Probably the most used category are the form widgets, which represent highly enhanced versions of classic form elements, such as buttons and input fields.

⁴⁵What You See Is What You Get ADBH Web [2010]

⁴⁶Accessible Rich Internet Applications, see <http://www.w3.org/WAI/intro/aria>

DojoX

DojoX represents Dojo's approach for the development of extensions to the toolkit Dojo Foundation [2010b]. It serves as a collection of subprojects officially known as *Dojo Extensions*, but in reality it is often called *Extensions and Experimental*, as a reference to DojoX's sub-categories Russell [2008]. Whereas *extensions* represent stable and mature widgets and resources not suitable for Dijit and Core, *Experimental* projects incorporate widgets that are either unstable or highly volatile and thus cannot be included in Core or Dijit.

Consequently, according to Dojo Foundation [2010b], DojoX serves two primary purposes. First, it is a repository for more stable and mature extensions and secondly, it also acts as a testbed for Experimental code. Furthermore, it is managed by distinct set of subprojects, having at least one module, a sponsor and a clear mission statement.

Russell [2008] concludes that DojoX strives for a *sensitive balance* for critical issues that are central to any community-supported open source project, meaning that although not all subprojects might be perfectly accurate, when it comes to meeting the accessibility and i18n initiatives defined by Dijit, still a considerable amount of DojoX projects are part of real-world applications.

Custom Widgets

As previously mentioned, writing *custom widgets* is achieved by either abstracting from existing Dijits, or by introducing new widgets by implementing the base interface. Thus, widgets must meet specific interface declarations. Dojo Foundation [2010c] states that all widgets in Dijit and DojoX are built on top of the *dijit._Widget* base class.

Furthermore, a differentiation between plain and *templated* widgets, provided by *dijit._Templated* has to be made. Whereas *dijit._Widget* represents the most basic widget possible that is required to construct its own DOM tree for presentation and interaction, *dijit._Templated* is a more enhanced version that takes a reference to a HTML fragment (i.e. the *template*) and automatically builds the corresponding DOM information. Previous experiences have shown that in most cases *dijit._Templated* will be used in favor of *dijit._Widget*, due to its pre-existing base functionality.

Util

The *Util* package rounds up Dojo's functionality. It incorporates a collection of utilities that facilitate *code management* and *testing* Russell [2008]. The list of utilities includes the *Dojo Objective Harness* (DOH⁴⁷), a JavaScript unit-testing framework developed by Dojo's community and build tools for creating custom Dojo versions. DOH represents Dojo's attempt to solve the complexities of testing JavaScript code Dojo Foundation [2010a]. It is one of few currently available unit-testing frameworks that supports testing of asynchronous (Ajax) functions.

Due to its ability to test an application's visualization, Jurkiewicz and Walter [2008] refer to DOH as JUnit⁴⁸'s counterpart for Web 2.0 user interfaces. DOH's main aim was to be both flexible and extendable and yet still compatible with as many different environments as possible. Apart from supporting a wide range of browsers, DOH also can be executed in non-browser environments, such as Rhino⁴⁹. Since DOH is not specifically coupled to Dojo Russell [2008], it can also be used as a general purpose JavaScript testing framework.

Dojo's build tools make up the second part of the utilities. Their primary purpose is two-fold. Firstly, JavaScript code can be structured into so-called *layers* that serve as a bundled collection of JavaScript files, and secondly, using ShrinkSafe⁵⁰, generated JavaScript code can be shrunk to the minimum size required for optimization. As such, ShrinkSafe itself is a patched version of the aforementioned Rhino JavaScript engine Russell [2008].

⁴⁷<http://dojotoolkit.org/reference-guide/util/doh.html>

⁴⁸<http://www.junit.org/>

⁴⁹<http://www.mozilla.org/rhino/>

⁵⁰<http://shrinksafe.dojotoolkit.org/>

4.3.2.2 PHP

There are various reasons for selecting *PHP* as the server side programming language. Most importantly, PHP has become the web scripting language of choice since its introduction in 1995 as *PHP/FI* The PHP Group [2011b], due to its simplicity, ever-evolving functionality and excellent performance Alshanetsky [2005]. It provides an extensive range of pre-defined functionality The PHP Group [2011a], thus enabling rapid development of web applications.

Moreover, through a vast collection of *PECL*⁵¹ extensions, PHP's functionality can be further enhanced. Furthermore, with the advent of PHP 5, object oriented programming model support has been introduced using the Zend Engine 2.0 The PHP Group [2011c], which provides better data and functional encapsulation facilities. Finally, it is fully compatible with the Apache Webserver, as well as the MySQL database server used by the exercises module's web-stack configuration described in Section 4.3.2.

4.3.2.3 JSON

Searching for web-based client-server communication formats reveals two prominent acronyms:

- *Extensible Markup Language* (XML)
- *JavaScript Object Notation* (JSON)

JSON, represents a lightweight, language-independent and plaintext data interchange format, which is derived from the *ECMAScript* Programming Language Standard Crockford [2006], described in Section 4.2.3.7.

JSON provides a set of simple formatting rules that are used for *portable* representations and serialization of structured data Crockford [2006]. Listing 4.20 provides an exemplary JSON representation of image meta-data.

```

1 {
2   Image: {
3     Width: 800,
4     Height: 600,
5     Title: "View from 15th Floor",
6     Thumbnail: {
7       Url: "http://www.example.com/image/481989943",
8       Height: 125,
9       Width: "100"
10    },
11    IDs: [116, 943, 234, 38793]
12  }
13 }
```

Listing 4.20: Example of JSON Notation [Crockford [2006]]

Basically, *object* definitions need to be embedded inside curved brackets (`{}`, see Line 1), whereas *arrays* are defined through square brackets (`[]`, see Line 11). Data entity labels, as well as their corresponding values can be optionally surrounded by quotation marks (`"`), as shown in Line 8 and 9. Hereby, data entity labels are separated by colons (`:`) from their corresponding values. Since values are not limited to primitive datatypes, nesting is possible, as shown in Line 2 and 6. Finally, data entities are separated by commas (`,`), as shown in Line 3.

Based on the goals and tasks identified, as well as the tools and technologies selected, Section 4.3.3 presents the exercises module's implementation details.

⁵¹PEAR (PHP Extension and Application Repository) Extended Code Language, see <http://pecl.php.net/>

4.3.3 Implementation

Preceding the actual implementation, the tools and technologies described in Section 4.3.2 were set up. In conformance to the e-learning platform's overall objective as a platform and browser independent web application, the respective development environment settings were determined. Thus, prior to the presentation of the implementation details in Section 4.3.3, the development environment will be discussed, followed by a more detailed description of the module's requirements in Section 4.3.3.2.

4.3.3.1 Development Environment Setup

One of the main objectives regarding the development environment used for the exercises module was to reuse the settings from the dictionary module described in section 4.2.3.1. Several adjustments were made corresponding to the tools and technologies selected. In difference to the dictionary module, PHP was selected as the server side scripting language. Furthermore, the Dojo Toolkit described in Section 4.3.2.1 was chosen as the client side JavaScript framework. Consequently, the targeted IDE supported PHP and JavaScript, as well provided means to manage web server instances, run unit tests and integrate subversion functionality to enable revision control amongst developers. Based on the settings used for the dictionary module, NetBeans IDE was also chosen as the IDE for the exercises module.

Prior to configuring the IDE, the web-stack's components were set up. *XAMPP* has been used as web-stack, which is available for a broad range of popular operation systems, such as Windows and Linux Seidler [2011]. *XAMPP* was installed locally using the default settings. Thus, once started, the Apache server was listening for incoming connections on port 80 and the MySQL database server on port 3306.

The concluding step to follow the web-stack's configuration was the setup of the NetBeans IDE. In order to enable PHP support, the *PHP plugin* was installed, as well as the MySQL database connection configured through NetBeans' database services tab.

4.3.3.2 Requirements

Based on the goals and tasks defined in Section 4.3.1, a more refined requirements specification was created. As previously mentioned, the exercises module should serve as the e-learning platform's primary facility for interactive, adaptive language exercises. Therefore, the main focus was on two objectives. First to provide a flexible and powerful, yet simple to use user interface, specifically tailored for younger users, and second to reuse data previously assembled for the dictionary module, as discussed in Section 4.2.3.4.

The core functionality required by the exercises module can be separated into the following categories:

1. users should be able to select a *source language* they would like to practice,
2. based on the source language users should be able to choose the corresponding linguistic area.

Based on these two primary selections, the system should then provide users with a list of matching *learning packages* (LP), out of which they should be able to either select single exercises, or entire learning packages. Once a selection has been made and the *course* has been started, the system should take care of handling this practice session by providing means for *browsing* through the set of exercises, as well as keeping track of the overall progress.

To this effect, a multitude of different exercise types should be available, independent of the linguistic area, such as cloze-text, crossword, drag&drop, hotspots, as well as multiple-choice. Moreover, the exercises should be designed and implemented as *templates*, so that they can be loaded separately into

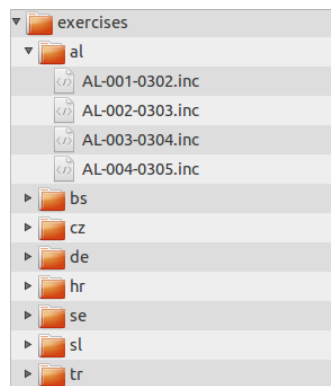


Figure 4.21: Exercise Templates Folder Structure

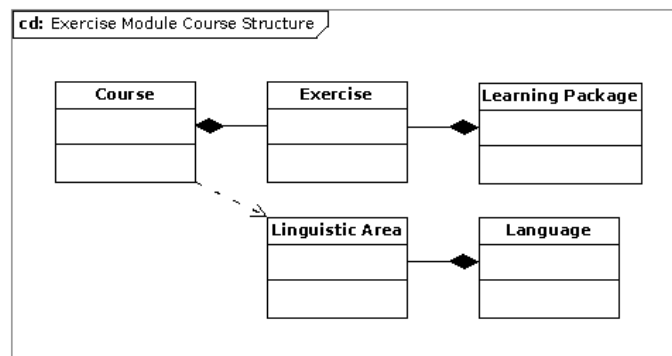


Figure 4.22: Exercise Module Course Structure

the system. Using a *standards-based* architecture, checking for results, as well as collecting feedback on the user's progress should be made possible.

Based on the aforementioned session handling mechanism, reloading of the exercises module in the browser should automatically reinitialize the session to the previous state. Consequently, reloading of the browser's window should at no time alter or even reset any previous progress made. Additionally, while doing the exercises, users should at all times be able to request *feedback*, including hints concerning the correctness of answers provided. Thus, the feedback mechanism should help users if they get stuck, or provide means for interactively checking their answers. Finally, during the testing phase of the implementation there should also be an option for submitting feedback to the developer team.

Consequently, the required core functionality can be summarized by six basic categories. First, users need to select the language to practice, as depicted in Figure 4.23. Afterwards, the corresponding exercises contained in learning packages need to be selected. In the following, users progress through the course. They are free to request feedback on their overall progress, as well as hints for the correctness of answers at any time. Finally, in case the browser window was reloaded, users should be able to continue their learning session based on previous progress. Figure 4.23 depicts the first three steps involved in the process of selecting the course's contents. Thus, based on the exercise/learning package selection, users should then be interactively guided through the learning course. A more detailed description of the user interface will be given in Section 4.3.3.4.

4.3.3.3 Data Schema

In order to reflect the exercise topology defined by the linguists, a standards-based and well-structured data schema had to be designed. Moreover, an appropriate container format was required for creating temporal exercise sequences, presented in *learning courses*. This section presents the data schema chosen, with respect to the e-learning standards discussed in Chapter 3.

As depicted in Figure 4.22, courses represent distinct collections of exercises, that are further embedded in learning packages. Although courses may only consist of single exercises, the standard case would be the inclusion of entire learning packages. Apart from the actual exercises, courses also include further *meta-data*, such as the language selected, as well as the linguistic area, conforming to the aforementioned exercise topology. Thus, courses basically serve as a mental model, representing sets of exercises contained in learning packages.

In this implementation, the list of available learning packages is published by the server side through a single JSON encoded file, *exercises.json*. Thus, prior to the actual course selection process, the client side needs this exercise meta-information file, in order to be able to build and update the course selection user interface. Based on JSON's syntax rules defined in Section 4.3.2.3, Listing 4.21 presents an exemplary Albanian spelling exercise taken from *exercises.json*.

```

1 {
2   dateiname: "AL-006-0307",
3   sprache: "al",
4   sprachbereich: "Rechtschreibung",
5   lernbereich: "Rechtschreibung 01: Fehler korrigieren 1",
6   lernpaket: "LP01: Rechtschreiben 1: Häufige Fehler üben",
7   lpindex: "1",
8   lernstufe: "leicht"
9 }

```

Listing 4.21: Exercise JSON Data Schema

Based on Listing 4.21, the following list describes the entities used by the data schema:

- **dateiname:** represents the filename of the exercise template. Filenames follow the mandatory specified in Listing 4.22. Hereby, EX_{idx} denotes the exercise's numerical index in the corresponding language set by field *sprache*, whereas EX_{g_offset} represents the exercise's global numerical offset in the entire set of available exercises. Figure 4.21 depicts the folder structure used by the exercises module to store exercise templates.
- **sprache:** specifies the exercise's language, represented by the respective ISO 3166-1 ALPHA-2 abbreviation code ISO [2010].
- **sprachbereich:** denotes the linguistic area to be practiced. Note that due to the heterogeneous nature of languages, chances are that only selected linguistic areas are available as exercises for certain languages. As of this writing, the following linguistic areas are available:
 - spelling
 - grammar
 - word meaning
 - word formation
 - vocabulary training
 - communication training

- **lernbereich:** defines the learning type, represented as a subset of the linguistic area. For instance, for the linguistic area *vocabulary training* a possible learning type would be *practicing words* by naming all components of an apple, whereas *building sentences* by substituting subjects with pronouns would be a possible combination for *grammar* related exercises.
- **lernpaket:** references the learning package containing the exercise referred to by *dateiname*. This reflects the idea shown in Figure 4.22, that learning packages represent collections of exercises for specific languages.
- **lpindex:** this entity represents the exercise's index in the learning package defined by *lernpaket*. This field serves as an ordering criterium for exercises contained in learning packages, starting at 1.
- **lernstufe:** this field specifies the exercise's level of difficulty, ranging from easy, over medium to hard.

```
1 <ISO 3166-1 ALPHA-2 code52>-<EXidx>-<EXg_offset>
```

Listing 4.22: Exercise Template Filename Convention

Note at this point that the exercise meta-information shown in Listing 4.21 does not include the actual exercise type. The exercise type and its actual functionality is provided through the corresponding template, represented by *dateiname* in Listing 4.21, leading to the loosely coupled architecture described in section 4.3.3.5.

4.3.3.4 User Interface

With regard to the e-learning platform's overall requirements specified in Section 4.1, the exercises module extends the platform with interactive exercises, represented by a standards-based, easy to use, clearly-structured, course-based and responsive graphical UI.

The exercises module is composed of two user interfaces, one for *assembling courses* and another for displaying the actual *exercises*. Figure 4.23 depicts the exercise module's user interface for selecting course contents. In conformance with the dictionary's UI⁵³, as well as the functional requirements specified in Section 4.3.3.2, it has been separated into three distinct areas. The topmost functional block denotes the *language selection*. Based on the language selected, the second functional block, referred to as the *linguistic area selection*, presents the user with a selection of available linguistic areas to choose from. As previously mentioned in Section 4.3.3.3, due to the heterogeneous nature of languages, there is a chance that only a limited number of linguistic areas is available for certain languages. Finally, the third functional block lists the matching learning packages, ordered by their numbers.

In order to assemble learning courses, users are free either to select entire packages by using the double arrow symbol on the right hand side, or to execute single exercises contained in it. Clicking on a learning package toggles the visibility of its contents. When expanded, incorporated exercises are displayed, order by their level of difficulty.

Once course contents have been selected, the user is redirected to the *exercise UI*. Note that since the exercises module incorporates a multitude of different exercise types, Figure 4.24 merely serves as an exemplary representation of the exercise UI. Although exercise templates conform to a basic layout template, the final representation might differ. Basically, the topmost section describes the learning type, together with a descriptive text. The main part is composed of the exercise itself, incorporating a header

⁵²ISO [2010]

⁵³see Section 4.2.3.3



Figure 4.23: Exercise Module Course Selection User Interface

and a body section. Whereas the header section serves as a container for exercise related components, such as draggable words in a drag&drop exercise, as shown in Figure 4.24, the body part represents the exercise's core, containing the main functionality.

Apart from the core functionality, this UI additionally provides special *feedback facilities* that have been added in the course of the second development phase, based on usability data gathered during a formal experiment. Chapter 5 presents a thorough description of the feedback facilities, as well as the formal experiment conducted.

4.3.3.5 Architecture

As shown in Figure 4.25, the exercises module is based on a loosely coupled client-server architecture. Whereas the majority of the business logic is incorporated in the client portion, represented by the set of exercise type implementations, the server side merely serves as an infrastructure to retrieve exercise templates and manage course sessions.

In order to provide a unique entry point to the server side, it has been implemented using the front controller design pattern Fowler [2002]. Based on the command specified, the front controller delegates the action to the corresponding submodules, represented by the *exercise selection*, as well as the *exercise execution* components depicted in Figure 4.25. These submodules conform to the graphical frontends described in Section 4.3.3.4. Hence, whereas the exercise selection portion provides facilities to assemble learning courses, the functionality to display and execute exercises is provided by the exercise execution module. The ability to manage courses is provided by the *session management* component, which uses PHP's built-in session facilities⁵⁴ to handle learning sessions.

⁵⁴<http://www.php.net/manual/en/book.session.php>

6. pád: jednotné číslo
6. Fall: Singular

Přiřaď k obrázkům správné výrazy! Myší přetáhni výrazy k obrázkům!
Ordne den Bildern die richtigen Phrasen zu! Ziehe die Phrasen zu den Bildern!

v parku na zahradě na stole v knize v divadle v moři na židli ve škole v pokoji










		
		
		

Figure 4.24: Exercise Module Exercise User Interface

As shown in Figure 4.25, there exists a mutual connection between the client and server side. The server side's submodules include the client side's presentation libraries, while the client side uses the *API* provided by the server's front controller. Hereby, the API boils down to two primary functions: either the entire learning package catalog, or specific exercises can be requested in the form of JSON encoded data.

The aforementioned loosely coupled architecture becomes more obvious when taking a closer look at the client side. Although the server side maintains the catalog of available exercise templates, as depicted in Figure 4.25, the actual implementation of the corresponding exercise type is provided through the client side's base exercise type implementation. For instance, when requesting a drag&drop based exercise from the server side, the corresponding template will be returned, which refers to the actual JavaScript-based exercise implementation residing on the client side. To illustrate this idea, Appendix C lists an exemplary drag&drop based exercise template.

The chosen architecture enables linguists to design exercises independently from the exercise type, since all base exercise types must implement the exercise interface shown in Figure 4.25. Thus, exercise templates located on the server side merely define the data and exercise type to be used for the presentation layer without making an assumption about the actual implementation provided by the client side.

4.3.3.6 Standards Used

In reference to the standards used by the dictionary module explained in Section 4.2.3.7, the standards incorporated in the exercises module can also be divided into two categories: *data management* and the *presentation layer*.

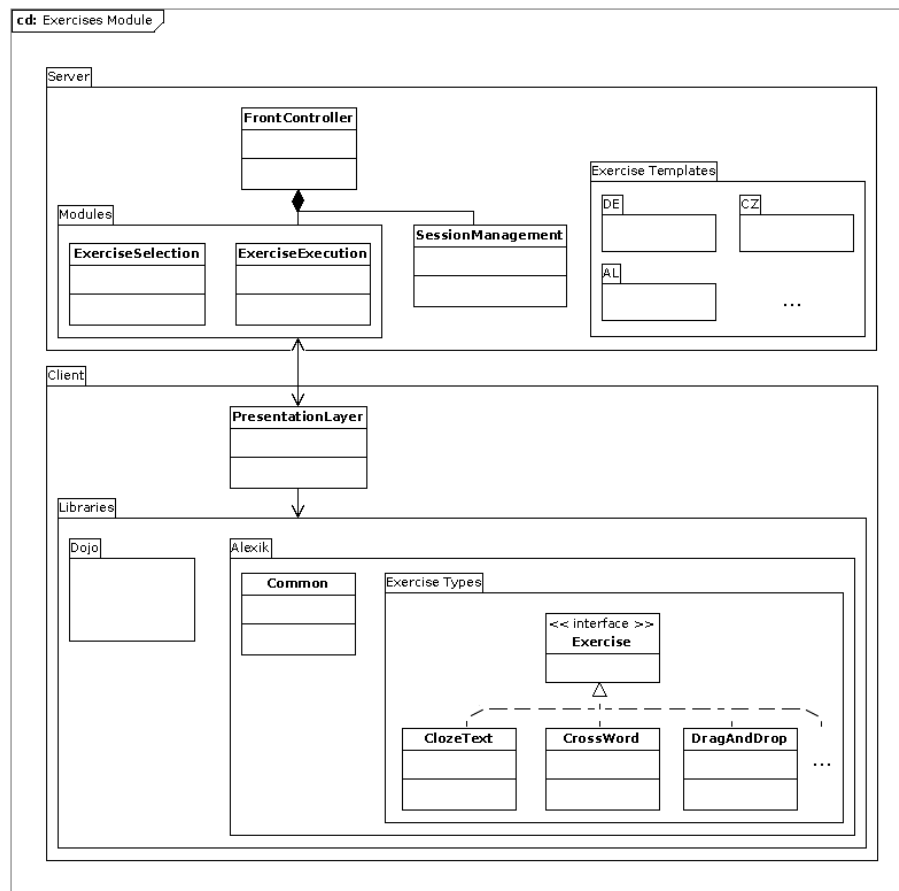


Figure 4.25: Exercises Module Architecture

Whereas the same presentation layer standards have been used, further distinctions have to be made concerning the data management. Based on the data scheme presented in Section 4.3.3.3, additional metadata had to be introduced to describe learning objects available to the exercise module. In reference to current e-learning standards discussed in Chapter 3, the goal was to minimize the overhead for defining meta information, and instead, concentrate on web technology related standards, such as HTML, JavaScript and CSS conformance.

As a consequence, we chose the proprietary format for learning material described in Section 4.3.3.3, as opposed to implementing SCORM or QTI. This decision was made due to the fact that we would have been also forced to implement the corresponding LCM functionality required, which was clearly out of scope for this project. Thus, instead of using metadata described in Section 2.2.2.1, we introduced a minimal, yet flexible learning package structure that enabled the exercise module to easily retrieve matching learning material based on the selection made. The selection process and the corresponding required data is explained in more detail in Section 4.3.

Chapter 5

Feedback

“Everybody needs feedback, and it’s a heck of a lot cheaper than paying a trainer.”

[Louis [2006]]

Feedback plays an important part in software development processes. Not only does it provide developers with valuable test and end user-information, but also allows the detection of problems at early stages. Unfortunately, there is little buffer space in finding the right frequency for feedback requests. Out of experience, users tend to get irritated when asked to provide feedback too often, especially when exposed to “buggy” software. The right amount of feedback rounds can therefore be rather difficult to determine. Frequency heavily depends on the complexity of the software under test, as well as the fact that it should not become a burden for users to actually provide feedback.

For this project it was decided to conduct a single feedback round in the form of a simplified formal experiment. As well as gathering usability information from a selected group of representative test users through a series of simple tasks, detection of comprehension barriers regarding different linguistic proficiencies would be of interest. The outcome of this formal experiment should then provide enough information for the second development phase of the project, which was dedicated to improving the system’s usability, as well as debugging. In the end, this approach turned out to be very valuable.

This chapter starts with a presentation of the feedback round conducted between the two development phases in the form of a formal experiment in section 5.1. Based on the feedback information gathered, Section 5.2 discusses the lessons learned and furthermore presents the measures taken to improve the e-learning platform *Wörterwelt*.

5.1 Formal Experiment

Between the two development phases of the ALEXIK project a feedback round was conducted in the form of a formal experiment in order to gather usability information from a representative group of selected test users.

Since the e-learning platform was specifically targeted at younger people, twenty fourth graders of St. Andrä elementary school in Graz kindly volunteered to participate in this usability study. In the following, the test methodology deployed for this formal experiment will be presented in greater detail, including the underlying test procedure in Section 5.1.1, a more detailed description of the test users in Section 5.1.2, the test environment in Section 5.1.3, the set of test tasks in Section 5.1.5 and finally, the concluding feedback questionnaire in Section 5.1.6.

Hardware	HP Pavilion PC, Intel Pentium 4 2Ghz Processor, 1 GB RAM
Operating System	Windows XP Professional SP3
Web Browser	Mozilla Firefox 2
Connection	Cable 100mbps
Monitor Resolution	1024x768
Monitor Size	19" TFT
Peripherals	HP Keyboard and Mouse

Table 5.1: Hardware and Software Environment used for the Formal Experiment

5.1.1 Test Procedure

Test participants were engaged a total of ten tasks with the e-learning platform. A between groups test design was used. Before the test participants were given their tasks, they were allowed to experiment with the system. In conclusion they were given the feedback questionnaire shown in Appendix A, to gather subjective rating data. Irrespective of common formal experiment practices, our test users were not recorded during their test runs.

Test operators were present to observe carefully and detect any difficulties, or simply to provide help when needed. Also, as opposed to traditional formal experiment setups, questions asked by test participants during their test runs were answered immediately. This was a precautionary measure taken to avoid overstraining or intimidating our young test participants.

In order to provide test participants with the best possible way to express their subjective ratings and thoughts, the feedback questionnaire also contained a concluding free text area where they could express wishes or complaints about the system. In addition, at the end of each test run test participants were engaged in a short concluding interview which provided a final opportunity to comment on their overall experience.

5.1.2 Test Users

Wörterwelt is specifically targeted at younger users and a representative selection of test users was acquired. Twenty fourth graders of St. Andrä elementary school in Graz volunteered as test participants for this formal experiment. They were assigned to one of the following groups, according to their linguistic proficiency:

- green
- yellow
- orange
- red
- black

The group colors represent the corresponding language ability, ranging from good (green) to bad (black). The individual participant's linguistic proficiency was determined by a language teacher in advance, to determine the appropriate language exercises to be used in the formal experiment. Depending on group membership, participants were given the appropriate tasks.

Die Aussprache und Schreibung von Wörtern mit „V“ und „F“

Lies die Erklärungen durch, wenn du sie verstanden hast, schreibe den gesuchten Namen in die Tabelle!

- Eine giftige Spinnenart ist die →
- Jemand, den du sehr gerne hast und mit dem du in deiner Freizeit etwas unternimmst, ist ein →
- Vater, Mutter und Kinder sind eine →
- Ein Tier, das fliegen kann, ist ein →
- Ein rundes Spielzeug, mit dem man Tore schießen kann, ist ein →
- Ein großes Haus, in dem Maschinen und Waren hergestellt werden, nennt man eine →
- Wenn ein Mann ein Kind bekommt, wird er →

Ergebnis

Gedrückt halten um richtige und falsche Übungen hervorzuheben!

Hilfe

Figure 5.1: Interactive Help System of the Exercises Module

5.1.3 Test Environment

The formal experiment was conducted by Prof. Muhr at the computer lab of the Austrian German Research Center in Graz¹. Table 5.1 shows the hard- and software setup used for the formal experiment. The tests were conducted using Mozilla's Firefox on Windows XP. In order to minimize page-loading times the e-learning platform was installed on a server in the local network.

5.1.4 Training

In order to accustom the participants to the e-learning platform, they were asked to browse the system prior to receiving their actual tasks. They were given fifteen minutes to browse freely through the system and even begin learning courses on their own. Questions asked during the training session were answered immediately. Once this training session was over, the test operator closed the browser to invalidate any active learning sessions previously started.

5.1.5 Tasks

Test users were asked to do a number of ten consecutive tasks in total. Although the e-learning platform already provided exercises and dictionaries for a multitude of different languages, in order to achieve comparable results all participants were asked to do the tasks using German as the source language. Using this approach, we were able to detect difficulties based on varying language proficiencies, as mentioned in Section 5.1.2.

Participants were allowed as much time as they needed to accomplish their tasks. Various question difficulty levels as well as available exercise types were tested. Participants were also allowed to provide subjective ratings via the feedback questionnaire any time during their test run.

Finally, the test operator tried to engage participants in a concluding interview. Participants were hereby asked to comment freely on their overall experience with the system. Furthermore, in case that a

¹<http://www-oedt.kfunigraz.ac.at/OEDTPORTAL/content/02kontakt/1Werwirsind.htm>

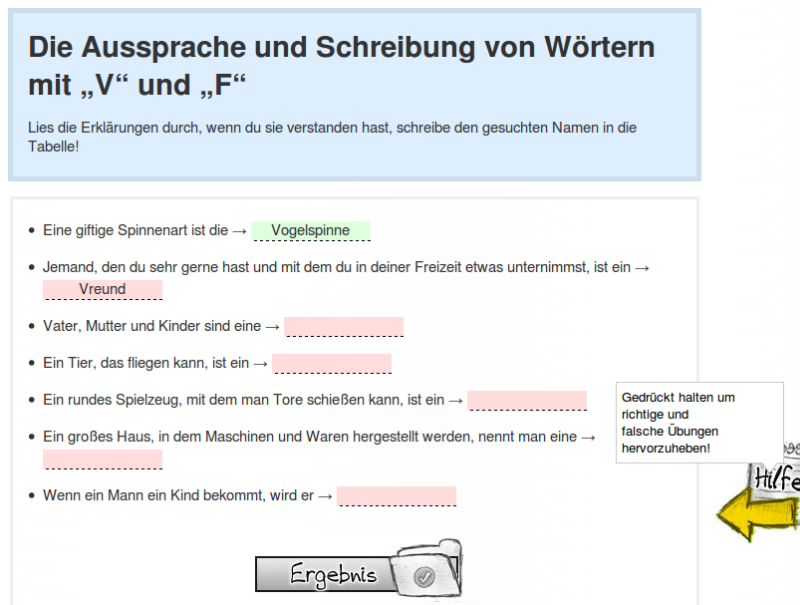


Figure 5.2: Interactive Help System of the Exercises Module highlighting answers

participant had used the feedback questionnaire's free text section, the test operator interviewed him/her more extensively about the written comments.

5.1.6 Feedback Questionnaire

Apart from the final interview, the feedback questionnaire shown in Appendix A represents the main source of subjective user ratings gathered during this formal experiment. As previously mentioned, participants were free to use the feedback questionnaire during their test runs, in order to be able to immediately rate their experiences while working with the system.

The questionnaire consisted of a short introductory text, a group indicator referring to the linguistic proficiency, four questions for each task and a concluding free text section, used for general comments on improving the system. The four questions per task were targeted at determining the test users' subjective rating of the corresponding exercise's *difficulty level*, *comprehension*, *learning potential* and overall *affection for exercise type*.

Except for one question a four point likert scale was used, ranging from *totally agree*, to *totally disagree*. For a single question a three-point likert scale was used. Overall, the feedback questionnaire was designed to be as simple as possible so as not to overwhelm the young participants with too much information.

5.1.7 Final Interview

At the end of each test run the test operator attempted to engage participants in short discussions. Simply by asking the question "How was it?" participants were encouraged to talk freely about their experiences. Furthermore, in case participants had made use of the free text section on the feedback questionnaire, they were questioned further about these ratings by the test operator.

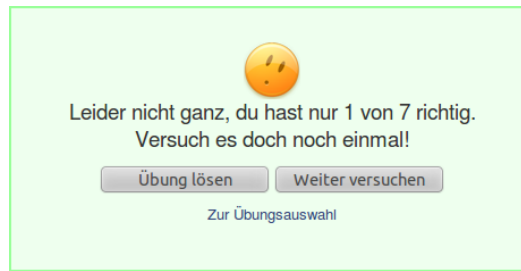


Figure 5.3: Interactive Help System Showing Incomplete Results

5.2 Lessons Learned

The formal experiment proved to be a valuable tool for detecting strengths and weaknesses of the e-learning platform. By observing twenty young participants, representative for the targeted end-user group, two major outcomes were deducted:

1. participants were able to interact with the e-learning platform without any problems regardless of their linguistic proficiency, and second,
2. additional feedback functionality was required to provide users with immediate help and progress information when using the exercises module.

Data from the feedback questionnaire indicates that all participants independent of linguistic proficiency experienced no severe difficulties doing the given tasks, resulting in a success rate of one hundred percent. Nevertheless, during the test runs, as well as the final interviews, a certain number of participants complained about the lack of feedback regarding their overall progress, as well as lacking assistance, particularly when confronted with difficult questions.

These criticisms conforms with the subjective ratings for the corresponding tasks. It was thus concluded that the exercises module needed an additional interactive help system. Furthermore, navigation through learning sessions in the system proved to be too complex, and an improved version includes the following list of requirements:

1. Include a navigation menu reflecting the progress of a learning session, displayed at the top and the bottom of exercise pages.
2. Provide an interactive help menu providing hints on the correctness of answers and help on exercises and that remains visible even when users scroll down the page.

A more detailed description of these additions to the functional requirements are presented in Sections 5.2.1 and 5.2.2.

5.2.1 Interactive Help

The interactive help system should provide users with help on exercises. It should be visible at all times on the exercise page, even when users scroll down the page. Figure 5.1 depicts the help system, showing its two main entry points. On the right hand side of the exercise page there is the primary *help button*, whereas the button at the bottom represents the *results button*.

The main help functionality is provided by the help button through a simple *onClick* event triggered by the user. To requests hints on the correctness of answers, users simply click and hold the help button,



Figure 5.4: Interactive Help System Showing Complete Results



Figure 5.5: Interactive Help System for the Exercises Module

highlighting correct answers in green and incorrect ones red, as shown in Figure 5.2. In this example the first answer is correct (“Vogelspinne”, German for tarantula), whereas the second is incorrect (“Vreund”, misspelled for German friend, i.e. “Freund”). Missing answers are treated as incorrect and will be marked red accordingly. Consequently, this help button also provides users with hints on the progress for the current exercise.

Additional support functionality is provided through the results button located at the bottom of the exercise page. The results button serves three primary functions:

1. show user’s progress for current exercise
2. provide auto-complete functionality
3. provide navigation through learning session

In case of incorrect or missing answers, users are presented the error screen shown in Figure 5.3 when the results button is pressed. Figure 5.4 depicts the screen shown when all answers are correct. The results button furthermore incorporates an *auto-complete* functionality, which automatically solves exercises by inserting correct answers. Thus, users can not only check their answers, they can have the help system solve entire exercises.

The third and final functionality provided by the result button is the navigation. Since users are allowed to switch between exercises contained in learning courses, a simplified version of the progress navigation bar has been incorporated.

5.2.2 Progress Information

The second improvement requested by test users was a better way to visualize overall progress information. Consequently, the main learning session navigation bar has been optimized and included in the header and footer section of exercise execution UI, as depicted in Figure 5.5. Users are now able to browse course contents by using the provided *back*, *forward* and *return to course selection* links, that are displayed depending on the current progress.

As opposed to classical paging navigation styles, the progress information bar does not display direct references to exercises contained. The design deployed should provide sufficient feedback on the users' course progress.

In total, conducting the formal experiment has proven very valuable for the project's second development phase. The feedback gathered confirmed the system's high level of usability, but also revealed insufficiencies in facilities for displaying help and progress information. Based on the feedback gathered from twenty young participants, the requirement specifications were updated and the e-learning platform Wörterwelt changed accordingly.

Chapter 6

Outlook

“Like any learning process, e-learning depends on effective communication of human knowledge, whether this occurs in a face-to-face classroom or across the Internet.”

[Bowles [2004]]

In the past decade, the e-learning sector has evolved to a flourishing market with a wide range of vendors and products. The need for standardized learning material to ultimately achieve interoperable e-learning systems is apparent. A multitude of organizations have become involved in the effort of combining existing diverse business interests to create flexible, yet sophisticated e-learning standards.

Two of the most prominent standards to date are SCORM, and QTI for test related activities. Their objective is to address every possible learning scenario, and due to this diversity they have evolved to complex structures that are costly to implement, creating several serious constraints, which are especially problematic for smaller projects.

6.1 General Trends

Despite of the complexity associated with state-of-the-art e-learning, it is still gaining popularity, as especially bigger companies and institutions have realized the potential of reusable learning material. It is important to note, that although the majority of the prominent e-learning systems, such as Blackboard or the open source alternative Moodle actively support existing standards, they do provide functionality using proprietary standards. This dual approach is based on the fact, that existing standards, such as SCORM are still in development, thus leaving room for interpretation.

The complexity and restrictive inherent conformity of current e-learning standard has started to experience criticism. One of the most prominent critics is Jim Groom, who coined the term “EduPunk”, as a counter-movement in electronic education. The name refers to Punk ideology of the 1970’s, a reaction against the conformity of e-learning standards. Other critics denunciate existing standards for their diminished pedagogical values. Since existing e-learning standards will most likely further increase in complexity, chances are these critical voices will become louder.

Event though SCORM is still yet evolving to become a de jure standard, there is a trend toward supporting and implementing it where ever applicable and monetary feasible. An interesting approaches to the problem would be to introduce service-oriented architectures that ultimately would make existing e-learning standards obsolete.

Finally, with the ever increasing volumes of learning material, the issue of intellectual property will play an integral part in the next few years. According to Naidu [2006], there already exists a tendency to adapt LOs for personal use. Consequently, strict rules must be defined to ensure proper handling of rights.

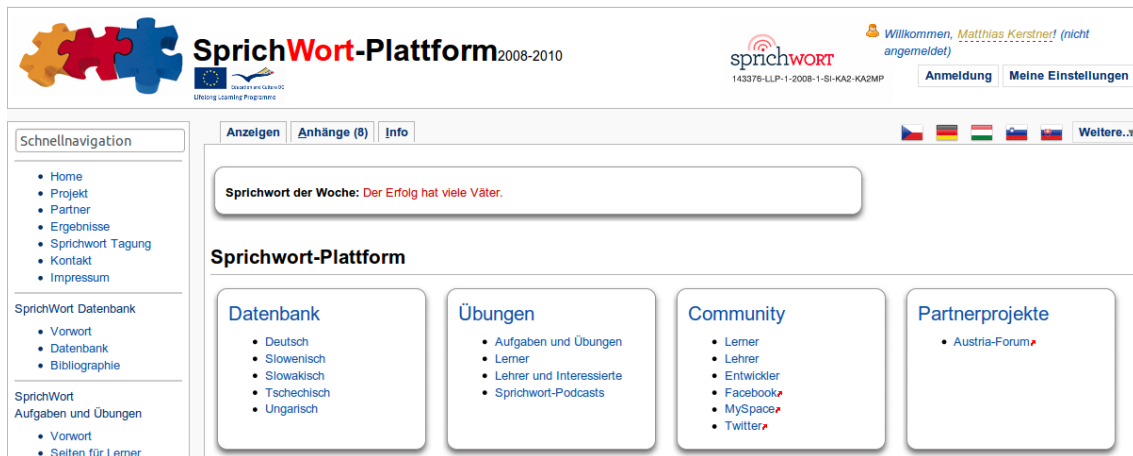


Figure 6.1: SprichWort-Plattform Welcome Page

6.2 Related Work

Based on the practical implementation of the e-learning platform Wörterwelt presented in this thesis, a subsequent project called *SprichWort-Plattform* has been developed. This section briefly introduces SprichWort-Plattform and compares Wörterwelt’s exercise module with the respective implementation in SprichWort-Plattform.

Figure 6.1 depicts SprichWort-Plattform’s welcome page. In contrast to Wörterwelt’s counterpart, users are able to choose from a wider variety of options to start the learning process, which are separated into four categories:

- *database*
- *exercises*
- *community*
- *partner projects*

In contrast to Wörterwelt, SprichWort-Plattform particularly focuses on proverbs, represented by its *database*. Furthermore, it is important to note that it does not offer the dictionary functionality provided by Wörterwelt as described in Section 4.2. Instead, it uses an alphabetical index to structure proverbs for the languages available.

The relation between these two e-learning platforms becomes obvious by looking at the exercises provided. Figure 6.2 depicts SprichWort-Plattform’s implementation of a cloze text exercise. By comparing it with Figure 5.2, similarities in the handling and visualization of user input become visible. In contrast to Wörterwelt’s exercises, SprichWort-Plattform does not provide means to select entire learning courses. Instead, users are required to manually select consecutive exercises from the corresponding lists. Thus, rather than “browsing” through the exercises selected for a linguistic area and language, as provided by Wörterwelt, users of SprichWort-Plattform need to manually iterate through a list of available exercises.

Like Wörterwelt, SprichWort-Plattform also includes special community features, targeted at learners and educators alike. In addition to the forum provided by Wörterwelt, SprichWort-Plattform also

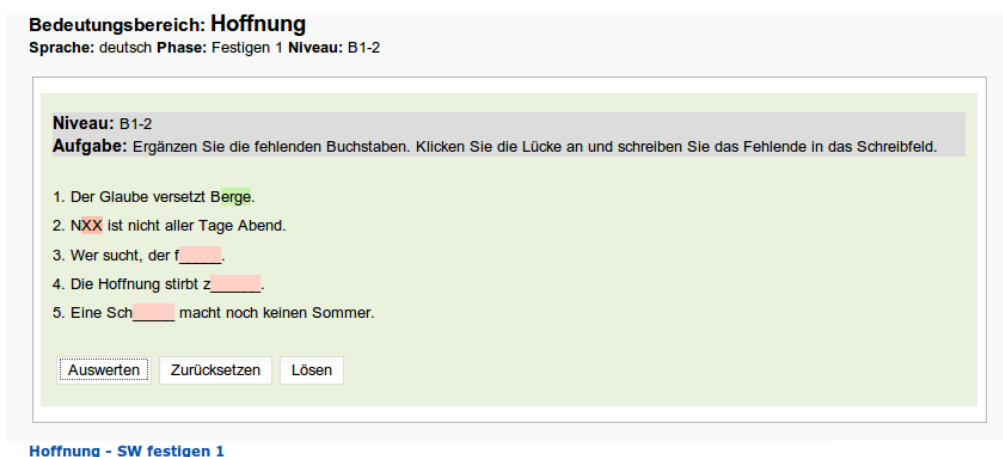
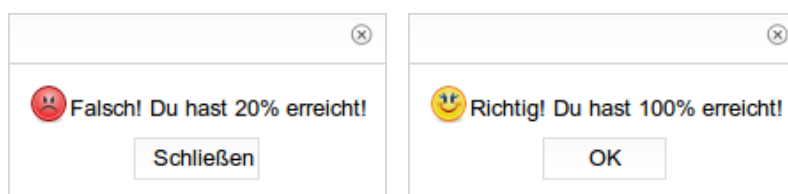


Figure 6.2: Example of SprichWort-Plattform’s cloze text exercises



(a) Sprichwort-Plattform Error Feedback (b) SprichWort-Plattform Success Feedback

Figure 6.3: SprichWort-Plattform Feedback Messages

refers to its Facebook¹, MySpace² and Twitter³ pages. Moreover, it also provides information on how developers can collaborate to create further learning material and improve the platform.

As opposed to Wörterwelt, SprichWort-Plattform is based on JSPWiki⁴, which includes a vast range of predefined plugins, such as a user management, as shown in the top right corner of Figure 6.1. Hereby, JSPWiki differentiates between core and contributed plugins JSPWiki [2009].

A more detailed description of SprichWort-Plattform’s architecture and implementation is provided by Christoph Portsch’s master thesis Portsch [2010].

6.3 Ideas for Future Work

Current e-learning standards are often too complex to be implemented by smaller projects. Furthermore, although current LMSs and LCMSs such as Moodle (section 2.3.1) provide rudimental support for standards such as SCORM and QTI, they often cannot reflect the flexibility of proprietary approaches. Consequently, proprietary LMSs oftentimes need to be developed alongside, instead of focusing on creating reusable learning material.

Fortunately, it seems that the standards committees have recognized problems inherent with complexity as demonstrated by the introduction QTI Lite, for instance. Maybe some time in the future SCORM will also undergo a process to reduce its complexity, or even provide a SCORM Lite.

¹<http://www.facebook.com>

²<http://www.myspace.com>

³<http://www.twitter.com>

⁴<http://www.jspwiki.org/>

Standards provide a motivating impulse to achieve consistent quality, reproducibility and conformity. Who could imagine a world today without TCP/IP or the HTML web standard? It is wiser to identify common interests than to “home-brew” implementations. Nevertheless, these can also become restrictive and too complex, eventually backfiring on their purpose.

Wörterwelt has proven that e-learning learning systems must not always blindly adhere to strict standards in order to achieve reusable learning materials. Concerning the exercises module, SprichWort-Plattform can be seen as the successor of Wörterwelt. Future work might relate to the missing dictionary functionality described in Section 4.2, which could be easily integrated into SprichWort-Plattform, as both are written using JSP.

Chapter 7

Concluding Remarks

In this thesis I have presented my research and practical work done in the field of e-learning and e-learning standards. With the practical implementation of the e-learning platform Wörterwelt it has been demonstrated that it is not mandatory to blindly adhere to current prominent e-learning standards such as SCORM or QTI. Using them as template instead provided the means to create a flexible e-learning architecture for using Wörterwelt's proprietary learning packages, which represent simplified counterparts of digital learning objects proposed by the respective standards.

Rather than being limited to a fixed number of exercise learning types and functionality provided by current standards conformant LMSs and LCMSs, Wörterwelt incorporates a wide range of exercise types based on a previously defined exercise topology for a broad range of translation languages. Furthermore, due to its flexible architecture and extensibility, additional exercise types can be added easily.

Apart from the exercises module, a dictionary module has been implemented for Wörterwelt that provides users with easy-to-use search functionality and presents results in a flexible, RIA dynamic grid. By the use of Ajax technology a desktop-like experience could be achieved.

In conclusion, it has been shown that focusing on web-standards instead of strictly adhering to complex e-learning standards, might provide the required flexibility to create learning-centered systems, rather than solely focusing on the technical specificities.

Appendix A

Feedback Questionnaire

Fragebogen

Liebe Kinder!

Danke, dass Ihr zu uns gekommen seid und uns helft, die Internet-Übungen zu verbessern!
Damit wir wissen, wie euch die Übungen gefallen haben, bitten wir euch, diesen Fragebogen auszufüllen, indem ihr die passende Antwort ankreuzt.

Ich/Wir gehöre/n zum Team

	grün	gelb	orange	rot	schwarz
Die Übung 1 war		ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich		gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass		man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 1		sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 2 war		ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich		gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass		man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 2		sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 3 war		ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich		gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass		man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 3		sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 4 war		ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich		gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass		man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 4		sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 5 war		ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich		gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass		man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 5		sehr gut	ziemlich gut	mittelgut	gar nicht gut

Figure A.1: Formal Experiment Feedback Questionnaire Page 1

Ich/Wir gehöre/n zum Team

grün	gelb	orange	rot	schwarz
Die Übung 6 war	ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich	gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass	man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 6	sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 7 war	ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich	gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass	man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 7	sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 8 war	ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich	gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass	man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 8	sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 9 war	ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich	gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass	man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 9	sehr gut	ziemlich gut	mittelgut	gar nicht gut
Die Übung 10 war	ganz leicht	leicht	mittel	schwer
Was ich tun soll, habe ich	gut verstanden	weniger verstanden		überhaupt nicht verstanden
Ich finde, dass	man viel lernen kann	einiges lernen kann	wenig lernen kann	gar nichts lernen kann
Ich finde die Übung 10	sehr gut	ziemlich gut	mittelgut	gar nicht gut

Was würdest du verbessern?

Figure A.2: Formal Experiment Feedback Questionnaire Page 2

Figure A.1 and A.1 depict the first and second page of the feedback questionnaire used in the formal experiment described in Chapter 5.

Appendix B

TextTools

TextTools represents a command-line based collection of text manipulation functions. It is capable of processing an arbitrary amount of plaintext source files. Listing B.1 shows an exemplary call of TextTools.

```
1 Texttools -w=my_output_file.txt -s=asc -e=; -ed=my_extracted_lines.txt
   source1.txt source2.txt
```

Listing B.1: Example TextTools Call

This call reads the source files *source1.txt* and *source2.txt*, whereas blank lines are excluded by default. Then it sorts the lines lexicographically ascending (*-s=asc*), which represents the default setting and can be turned off by using *-b=0*. Following this, TextTools extracts all characters from the lines until the delimiter *;* is found (*-e=;*). If the delimiter cannot be found the entire line is extracted. TextTools then writes the extracted lines to the output file specified by *-ef=my_extracted_lines.txt*. Finally, it writes the sorted lines from step 2 back to *my_output_file.txt*.

The help menu is displayed when calling TextTools without any flags, as shown in Listing B.2.

```
1 Texttools.exe:
2
3 usage: Texttools
4   [-d=0]
5   [-e=delim [-ed=dest ]]
6   [-o=delim [-od=dest ]]
7   [-p= [-ps=src] [-pd=dest ]]
8   [-r=expr [-rd=dest ]]
9   [-s=0|asc|desc [-sd=dest ]]
10  [-u=delim [-ud=dest ]]
11  [-w=c|dest|s]
12  [src_files ...]
```

Listing B.2: TextTools Help

Although active development of TextTools has been discontinued, the reader may consult the official documentation for further information, as well as download the latest version from the project homepage¹.

¹<http://www.kerstner.at/texttools/>

B.1 Statistics

The `-d` flag toggles the functionality to display statistical output. If `-d=0` is used no statistical output will be displayed. This functionality is turned on by default (`-d=1`).

B.2 Extraction

Extracts all strings from the source files, up to the delimiter specified (`-e=delimiter`). Use `-ed` to specify to destination filename, otherwise the corresponding default filename will be used, as described in section B.8.

B.3 Rotation

Rotates strings until the given delimiter is found (`-o=delimiter`). If no delimiter is specified the entire line will be rotated. Use `-od` to specify custom output filename.

B.4 Replacing

Replaces strings based on the *rules file* specified by the `-ps` flag. Use `-pd` to specify custom output filenames.

B.4.1 Replacement Rules

Rules must follow the format specified in Listing B.3.

```
1 exprold | exprnew [ | envbegin [ | envend ] ]
```

Listing B.3: TextTools Rules Format

Expressions can be replaced in the context of entire lines inside *environments*. Hereby, environments have special beginning and end delimiters. This can either be a single character or an arbitrarily long string. In case no environment is specified Texttools replaces all occurrences of *expr_{old}* by *expr_{new}*. When using environments the options specified in Listing B.4 are available.

```
1 [ | ( $ || BOL || delimbegin ) [ | ( $ || EOL || delimend ) ] ]
```

Listing B.4: TextTools Rules Environment Format

The `$` symbol represents an empty string. `BOL` denotes the beginning of a line, whereas `EOL` represents the end of a line. The following examples illustrate this concept.

1. `TEST|new` replaces all occurrences of `TEST` with `new`, which is the same as `TEST|new|BOL|EOL`
2. `TEST|new|BOL` replaces all occurrences of `TEST` with `new`, only if `TEST` is the first word of the current line.
3. `TEST|new|EOL` replaces all occurrences of `TEST` with `new`, only if `TEST` is the last word of the current line.

B.5 Sorting

This command sorts data specified by source files. There are four scenarios possible for this flag. First, if the flag is not specified or value equals 0, the list will not be sorted. Secondly, using `-s=asc` will sort the list lexicographic ascending, whereas `-s=desc` sorts it lexicographic descending. Otherwise an error will be displayed.

B.6 Removal of Duplicates

Removes all duplicate entries. Specify no predicate to remove all duplicate strings. Use `-ud=` to specify custom output filenames used for backing up lines removed.

B.7 Output

This flag specifies the general output filename to be used (`-w`). If it is not specified or an empty value is used, `DEFAULT_OUTPUT_FILE` is used. If `-w=s` is specified TextTools will automatically write the output to files starting with the line's leading character, i.e. `a-z.txt` and `other.txt`. Otherwise, the filename specified will be used. Use `-w=c` to print the output to the command line.

B.8 Default Filenames

The following list shows the default output filenames used by TextTools.

- `DEFAULT_OUTPUT_FILE` *output.txt*
- `DEFAULT_OTHER_FILENAME` *other.txt*
- `DEFAULT_EXTRACTION_FILE` *extracted.txt*
- `DEFAULT_UNIFICATION_FILE` *unification.txt*
- `DEFAULT_REMOVE_FILE` *removed.txt*
- `DEFAULT_ROTATION_FILE` *rotated.txt*
- `DEFAULT_RULES_FILE` *rules.txt*
- `DEFAULT_REPLACE_FILE` *replaced.txt*

Appendix C

Exercise Template

Listing C.1 represents a shortened version of a drag&drop based exercise template for Czech. Line 1 shows the base exercise type. Lines 5-11 denote the the exercise data definition. Following the references to the implementation files are the presentation layer definitions. Exercises need to have a task definition section (Line 16) and a container for the actual exercise's contents using the id *exercise*, as shown in Line 18.

```
1 <script type="text/javascript" src="lib/alexik/draganddrop.js"></script>
2
3 <script type="text/javascript">
4   function init() {
5     exercisePairs = [[langsam, "rychlý"],
6       ...,
7       [jung, "starý"]];
8
9     initDragAndDrop();
10  }
11
12  dojo.addOnLoad(init);
13 </script>
14
15 <div id="task-definition">...</div>
16
17 <div id="exercise">
18   <div dojoType="dojo.dnd.Source" class="dnd-source-field">
19     <span class="dojoDndItem">tenký</span>
20     ...
21   </div>
22
23   <table class="dnd-target-table">
24     <tr>
25       <td>pomalý</td>
26       <td dojoType="dojo.dnd.Source" class="source" jsId="langsam"></td>
27     </tr>
28     ...
29   </table>
30   ...
31 </div>
```

Listing C.1: Wörterwelt Drag&Drop Exercise Template Example

Bibliography

- ADBH Web [2010]. *T171 TMA3 - The Importance Of WYSIWYG*. <http://www.adbh.co.uk/t171/tma3.php>, Last accessed: 2011-03-03. (Cited on page 54.)
- ADL [2001]. *The SCORM Overview*. Advanced Distributed Learning. <http://xml.coverpages.org/SCORM-12-Overview.pdf>. (Cited on page 19.)
- AICC [2010]. *AICC Publications*. http://aicc.org/joomla/dev/index.php?option=com_content&view=article&id=64&Itemid=28#AGRs. (Cited on page 4.)
- AICC CMI Subcommittee [1998]. *Web-Based Computer-Managed Instruction*. www.aicc.org/docs/AGRs/agr010v1.pdf. (Cited on page 4.)
- AICC CMI Subcommittee [2004]. *CMI Guidelines for Interoperability*. <http://www.aicc.org/docs/tech/cmi001v4.pdf>. (Cited on pages 3 and 9.)
- Alshanetsky, Iliia [2005]. *php—architect’s Guide to Security*. First Edition. Marco Tabini & Associates, Inc., Toronto, Canada. ISBN 0973862106. (Cited on page 56.)
- Andrews, Keith [2010]. *Writing a Thesis: Guidelines for Writing a Master’s Thesis in Computer Science*. <http://ftp.iicm.edu/pub/keith/thesis/>. (Cited on page xi.)
- Austrian Federal Ministry for Education, Arts and Culture [2010]. *Schulbuchliste 0100. Volksschulen und Sonderschulen*. http://www.bmukk.gv.at/medienpool/18857/1011_sbl_0100.pdf. (Cited on page 25.)
- Bernard, Michael, Ryan Baker, Barbara Chaparro, and Marisa Fernandez [2001]. *Paging vs. Scrolling: Examining Ways to Present Search Results*. <http://psychology.wichita.edu/mbernard/HSEF.Paging.pdf>. (Cited on pages 40 and 41.)
- Blackboard Inc. [2004]. *Leading the Way on Standards-Based e-Learning*. http://www.blackboard.com/docs/AS/Blackboard_Whitepaper_Standards_QE.pdf. (Cited on page 17.)
- Bowles, Marc [2004]. *What Is Electronic Learning? Relearning to E-learn: Strategies for Electronic Learning and Knowledge*, pages 3–19. <http://search.informit.com.au/documentSummary;dn=825091693221992;res=IELHSS>. (Cited on pages 4, 5, 10, 11 and 73.)
- Brooks, Leslie Madsen [2008]. *Introducing Edupunk*. <http://www.blogger.com/introducing-edupunk>. (Cited on page 23.)
- Bush, Michael D. [2002]. *Connecting Instructional Design to International Standards for Content Reusability*. *Educational Technology*, 42(6), pages 5–13. ISSN 0013-1962. <http://arclite.byu.edu/digital/edtechscorm.htm>. (Cited on page 18.)

- Cantoni, Virginio, Massimo Cellario, and Marco Porta [2004]. *Perspectives and challenges in e-learning: towards natural interaction paradigms*. *Journal of Visual Languages Computing*, 15(5), pages 333–345. ISSN 1045-926X. doi:DOI:10.1016/j.jvlc.2003.10.002. <http://www.sciencedirect.com/science/article/B6WMM-4CJVC5G-1/2/c1b0405e36d7d67b1922c62b8c11b3da>. Image Understanding and Retrieval. (Cited on pages 9 and 15.)
- Chen, Henri and Robbie Cheng [2007]. *ZK™Ajax Without JavaScript™Framework*. First Edition. Apress. ISBN 1590599012. (Cited on pages 30, 31 and 32.)
- Cole, Jason and Helen Foster [2008]. *Using Moodle*. Second Edition. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. (Cited on pages 10, 11, 12 and 13.)
- Cressman, Darryl and Norm Friesen [2005]. *The Politics of E-Learning Standardization*. http://learningspaces.org/n/papers/standards_ant.doc. (Cited on page 24.)
- Crockford, D. [2006]. *The application/json Media Type for JavaScript Object Notation (JSON)*. URL: <http://www.ietf.org/rfc/rfc4627.txt>. Accessed: 2011-02-18. (Cited on pages vii and 56.)
- Deibler, Nina Pasini [2008]. *SCORM Tutorial IITSEC 2008*. <http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/Files/SCORMTutoriaIIITSEC2008.pdf>. (Cited on page 18.)
- Dojo Foundation [2010a]. *D.O.H: Dojo Objective Harness*. <http://dojotoolkit.org/reference-guide/util/doh.html>. (Cited on page 55.)
- Dojo Foundation [2010b]. *Reference Guide*. <http://dojotoolkit.org/reference-guide/dojo/index.html#dojo-core>. (Cited on pages 53, 54 and 55.)
- Dojo Foundation [2010c]. *Writing Your Own Widget*. <http://dojotoolkit.org/reference-guide/quickstart/writingWidgets.html#quickstart-writingwidgets>. (Cited on page 55.)
- Downes, Stephen [2008]. *Introducing Edupunk*. <http://www.downes.ca/cgi-bin/page.cgi?post=44760>. (Cited on page 23.)
- ECMA [2009]. *Standard ECMA-262 ECMAScript Language Specification*. Fifth Edition. ECMA. <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>. (Cited on page 50.)
- Eklund, John, Margaret Kay, and Helen M. Lynch [2003]. *e-learning: Emerging Issues and Key Trends*. <http://pre2005.flexiblelearning.net.au/research/2003/elearning250903final.pdf>. (Cited on pages 3, 4 and 25.)
- Fallon, Carol, Jeanne M. Dams, and Sharon Brown [2002]. *E-Learning Standards: A Guide to Purchasing, Developing, and Deploying Standards-Conformant E-Learning: A Primer for Using the Standards as Decision Support Tools*. First Edition. St Lucie Press. ISBN 1574443453. (Cited on pages 3, 4, 5, 6, 7, 8, 9, 10, 15, 16, 17, 18, 20, 21, 22 and 25.)
- Fowler, Martin [2002]. *Patterns of Enterprise Application Architecture*. First Edition. Addison-Wesley Professional. ISBN 0321127420. (Cited on page 61.)
- Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides [1994]. *Design Patterns. Elements of Reusable Object-Oriented Software*. First Edition. Addison-Wesley Longman, Amsterdam, Holland. ISBN 0201633612. (Cited on pages 37 and 47.)
- Garrett, Jesse James [2005]. *Ajax: A New Approach to Web Applications*. <http://www.adaptivepath.com/ideas/essays/archives/000385.php>. (Cited on page 27.)

- Godwin-Jones, Robert [2004]. *Emerging Technologies: Learning Objects: Scorn or SCORM? Language Learning Technology*, 8, pages 7–12. ISSN 1094-3501. <http://llt.msu.edu/vol8num2/emerging/default.html>. (Cited on pages 23 and 24.)
- GuideTools Ltd. [2009]. *Industry Standards Compliancy Organisations*. <http://hosting.guidetools.co.nz/scripts/runisa.dll?GUIDE:GT:1026761241:mthd=PAGE&course=GUIDEELEARNINGENGINEWEBSITEEMAY03&pageid=TRAINERSINTRO10&template=tplWebSite>. (Cited on page 18.)
- Hesse, Friedrich W. [2009]. *Use and Acquisition of Externalized Knowledge*. In *Proceedings of the 4th European Conference on Technology Enhanced Learning: Learning in the Synergy of Multiple Disciplines*, pages 5–6. EC-TEL '09, Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-642-04635-3. doi:http://dx.doi.org/10.1007/978-3-642-04636-0_3. http://dx.doi.org/10.1007/978-3-642-04636-0_3. (Cited on page 3.)
- Hui, Kin-chuen, Zhigeng Pan, Ronald Chi kit Chung, Charlie C.L. Wang, Xiaogang Jin, Stefan Göbel, and Eric C.-L. Li [2007]. *Technologies for E-Learning and Digital Entertainment Second International Conference, Edutainment 2007, Hong Kong, China, June 11-13, 2007. Proceedings*, volume 1. Springer Berlin/Heidelberg. ISBN 978-3-540-73010-1. doi:10.1007/978-3-540-73011-8. (Cited on page 1.)
- IEEE [2002]. *Draft Standard for Learning Object Metadata*. http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf. (Cited on page 8.)
- IMS [2002]. *IMS Question Test Interoperability QTILite Specification Final Specification Version 1.2*. http://www.imsglobal.org/question/qtiv1p2/imsqti_litev1p2.html. (Cited on page 22.)
- IMS [2003]. *IMS Content Packaging Information Model Version 1.1.3 Final Specification*. http://www.imsglobal.org/content/packaging/cpv1p1p3/ims_cp_infv1p1p3.html. (Cited on page 19.)
- IMS [2006]. *IMS Question and Test Interoperability Overview Version 2.1 Public Draft (revision 2) Specification*. http://www.imsglobal.org/question/qtiv2p1pd2/imsqti_oviewv2p1pd2.html. (Cited on pages 20 and 21.)
- IMS [2011]. *About IMS Global Learning Consortium*. <http://www.imsglobal.org/background.html>. (Cited on page 4.)
- ISO [2002]. *ISO/IEC 16262:2002 - Information technology - ECMAScript language specification*. http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm. (Cited on page 50.)
- ISO [2008]. *ISO/IEC 9075-14:2008 - Information technology – Database languages – SQL – Part 14: XML-Related Specifications (SQL/XML)*. http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=45499. (Cited on page 50.)
- ISO [2010]. *English country names and code elements*. http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm. (Cited on pages 35, 36, 45, 59 and 60.)
- IsoDynamic [2001]. *E-Learning*. http://www.isodynamic.com/web/pdf/IsoDynamic_elearning_white_paper.pdf. (Cited on page 5.)
- Jesukiewicz, Paul [2009a]. *SCORM 2004 4th Edition Content Aggregation Model (CAM) Version 1.1. Advanced Distributed Learning*. (Cited on pages 18, 19 and 20.)

- Jesukiewicz, Paul [2009b]. *SCORM 2004 4th Edition Run-Time Environment (RTE) Version 1.1*. Advanced Distributed Learning. (Cited on page 19.)
- Jesukiewicz, Paul [2009c]. *SCORM 2004 4th Edition Sequencing and Navigation (SN) Version 1.1*. Advanced Distributed Learning. (Cited on page 19.)
- Jones, Edward R. [2002]. *Implications of SCORM and Emerging E-learning Standards On Engineering Education*. In *Proceedings of the 2002 ASEE Gulf-Southwest Annual Conference*. American Society for Engineering Education. <http://www.aseegsw.org/Proceedings/IB5.pdf>. (Cited on pages 4, 5, 8, 16, 18 and 19.)
- jQuery Project and the jQuery UI Team [2011]. *jQuery UI - Documentation: UI/Getting_Started*. http://jqueryui.com/docs/Getting_Started. (Cited on page 53.)
- JSPWiki [2009]. *JSP Wiki Plugins*. <http://www.jspwiki.org/wiki/JSPWikiPlugins>. (Cited on page 75.)
- Jurkiewicz, Jared and Stephanie L. Walter [2008]. *Unit testing Web 2.0 applications using the Dojo Objective Harness*. <http://www.ibm.com/developerworks/web/library/wa-aj-doh/>. (Cited on page 55.)
- Kanendran, T. A., J. Savarimuthu, and B. V. Durga Kumar [2005]. *Issues in E-Learning Standards*. *Sunway Academic Journal*, 2, pages 55–65. http://www.scribd.com/document_collections/2342529. (Cited on pages 4, 15, 17 and 18.)
- Krasner, Glenn E. and Stephen T. Pope [1988]. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80*. *J. Object Oriented Program.*, 1, pages 26–49. ISSN 0896-8438. <http://portal.acm.org/citation.cfm?id=50757.50759>. (Cited on page 47.)
- Leacock, Claudia, Martin Chodorow, Michael Gamon, and Joel Tetreault [2010]. *Automated Grammatical Error Detection for Language Learners*. *Synthesis Lectures on Human Language Technologies*, 3(1), pages 1–134. doi:10.2200/S00275ED1V01Y201006HLT009. <http://www.morganclaypool.com/doi/abs/10.2200/S00275ED1V01Y201006HLT009>. (Cited on page 5.)
- Louis, Catherine Saint [2006]. *Pop Them in, and They're Ready to Push You*. <http://www.nytimes.com/2006/02/23/fashion/thursdaystyles/23Fitness.html>. (Cited on page 65.)
- Marshall, Stephen [2004]. *E-learning standards: Open enablers of learning or compliance strait jackets?* In *Beyond the Comfort Zone: Proceedings of the 21st ASCILITE Conference*, pages 596–605. ASCILITE. ISBN 0-9751702-3-6. <http://www.ascilite.org.au/conferences/perth04/procs/pdf/marshall.pdf>. (Cited on pages 23 and 24.)
- Montandon, Corinne [2004]. *Standardisierung im e-Learning - Eine empirische Untersuchung an Schweizer Hochschulen*. <http://www.iwi.unibe.ch/content/publikationen/arbeitsberichte/2004/e6050/e6133/e7162/e7164/e7170/AB161.pdf>. (Cited on page 19.)
- Moodle [2010a]. *About Moodle*. http://docs.moodle.org/en/About_Moodle. (Cited on page 11.)
- Moodle [2010b]. *Five Key Principles*. http://docs.moodle.org/en/five_key_principles. (Cited on page 11.)
- Moodle [2011]. *Modules and plugins*. <http://moodle.org/mod/data/view.php?id=6009>. (Cited on page 13.)

- Mozilla Developer Network [2010a]. *Firefox 3.6 for developers*. https://developer.mozilla.org/en/firefox_3.6_for_developers#JavaScript. (Cited on page 50.)
- Mozilla Developer Network [2010b]. *JavaScript Guide*. <https://developer.mozilla.org/en/JavaScript/Guide>. (Cited on page 50.)
- Muhr, Rudolf [2010]. *Hilfe zu den Wörterbüchern der Wörterwelt*. http://www-oedt.kfunigraz.ac.at/woerterwelt/content.php?page=hilfe_wb. (Cited on page 42.)
- Muhr, Rudolf and Mirna Kadric [2005]. *Wörterwelt*. First Edition. Weber, Eisenstadt, Burgenland. ISBN 3852533635. (Cited on page 25.)
- Naidu, Som [2006]. *E-Learning A Guidebook of Principles, Procedures and Practices*. Second Edition. Commonwealth Educational Media Centre for Asia, New Delhi, India. ISBN 8188770043. (Cited on pages 4, 5, 6, 7, 8, 9, 15, 17, 25 and 73.)
- Nichani, Maish [2001]. *LCMS = LMS + CMS [RLOs]*. http://www.elearningpost.com/articles/archives/lcms_lms_cms_rlos/. (Cited on pages 10 and 11.)
- Oaho, Alfred V., Jeffrey D. Ullman, and Mihalis Yannakakis [2009]. *Personal Services: Debating the Wisdom of Personalisation*. In Marc Spaniol, Ralf Klamma, Qing Li and Rynson W.H. Lau (Editors), *Lecture Notes in Computer Science*, pages 1–11. Number 1 in Advances in Web Based Learning, ICWL, Springer, Aachen, Germany. (Cited on page 25.)
- Piolat, Annie, Jean yves Roussey, and Olivier Thunin [1997]. *Effects of screen presentation on text reading and revising*. *International Journal of Human-Computer Studies*, 47, pages 565–589. (Cited on page 40.)
- Portsch, Christoph [2010]. *Wiki-based Assessment System for the Creation of Interactive Language Exercises*. www.sprichwort-plattform.org/attach/Ergebnisse/MA_Portsch_01_2010.pdf. (Cited on page 75.)
- Potix Corp. [2009a]. *ZK Mobile 0.8.10*. <http://www.zkoss.org/release/zkmob-rn-0.8.10.dsp>. (Cited on page 32.)
- Potix Corp. [2009b]. *ZK Mobile Docs*. http://docs.zkoss.org/wiki/ZK_Mobile_Docs. (Cited on page 30.)
- Potix Corp. [2009c]. *ZK Studio New Features*. http://docs.zkoss.org/wiki/ZK_Studio_New_Features. (Cited on page 32.)
- Potix Corp. [2010a]. *Top 10 Reasons*. <http://www.zkoss.org/WhyZK/top10.dsp>. (Cited on page 33.)
- Potix Corp. [2010b]. *ZK 5.0.5 release notes*. <http://www.zkoss.org/release/rn-5.0.5.dsp>. (Cited on page 32.)
- Potix Corp. [2010c]. *ZK Developer's Reference/Architecture Overview*. http://books.zkoss.org/wiki/ZK_Developer%27s_Reference/Architecture_Overview. (Cited on pages 30 and 33.)
- Potix Corp. [2010d]. *ZK Developer's Reference/MVC*. http://books.zkoss.org/wiki/ZK_Developer%27s_Reference/MVC. (Cited on page 47.)
- Potix Corp. [2010e]. *ZK Developer's Reference/UI Composing/Component-based UI*. http://books.zkoss.org/wiki/ZK_Developer's_Reference/UI_Composing/Component-based_UI. (Cited on pages 31 and 34.)

- Potix Corp. [2010f]. *ZK Getting Started/Tutorial*. http://books.zkoss.org/wiki/ZK_Getting_Started/Tutorial. (Cited on page 31.)
- Potix Corp. [2010g]. *ZK Homepage*. <http://www.zkoss.org>. (Cited on page 30.)
- Potix Corp. [2010h]. *ZK Installation Guide/Setting up Servers*. http://books.zkoss.org/wiki/ZK_Installation_Guide/Setting_up_Servers. (Cited on page 33.)
- Potix Corp. [2011]. *Direct RIA*. <http://www.zkoss.org/DirectRIA/>. (Cited on page 32.)
- Qiu, Mei Kang, Kang Zhang, and Maolin Huang [2004]. *An Empirical Study of Web Interface Design on Small Display Devices*. In *Web Intelligence, 2004. WI 2004. Proceedings. IEEE/WIC/ACM International Conference on*, volume 1, pages 29–35. ISBN 0769521002. doi:10.1109/WI.2004.10041. (Cited on page 30.)
- Russell, Matthew A. [2008]. *Dojo: The Definitive Guide*. First Edition. O'Reilly Media Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. ISBN 9780596516482. (Cited on pages 52, 53, 54 and 55.)
- Rustici, Mike [2009a]. *SCORM Versions – An eLearning Standards Roadmap*. <http://scorm.com/scorm-explained/business-of-scorm/scorm-versions/>. (Cited on pages 4, 18 and 23.)
- Rustici, Mike [2009b]. *Technical SCORM*. <http://scorm.com/scorm-explained/technical-scorm/>. (Cited on page 19.)
- Schmaranz, Klaus [2004]. *Entwurf und Entwicklung großer Systeme, Vorlesungsunterlagen SS 2004*. (Cited on pages 37 and 46.)
- Schroeder, Ulrik [2009]. *Web-Based Learning – Yes We Can!* In Spaniol, Marc, Qing Li, Ralf Klamma, and Rynson Lau (Editors), *Advances in Web Based Learning – ICWL 2009, Lecture Notes in Computer Science*, volume 5686, pages 25–33. Springer Berlin / Heidelberg. http://dx.doi.org/10.1007/978-3-642-03426-8_3. 10.1007/978-3-642-03426-8_3. (Cited on page 4.)
- Schwarz, E., I. P. Beldie, and S. Pastoor [1983]. *A comparison of paging and scrolling for changing screen contents by inexperienced users*. *Human factors*, 25(3), pages 279–282. (Cited on page 40.)
- Seidler, Kai Oswald [2011]. *XAMPP*. <http://www.apachefriends.org/en/xampp.html>. (Cited on page 57.)
- Seirand Institute [2008]. *An Introduction to ZK A Direct RIA Development Framework*. <http://www.zkoss.org/support/training/webinar/zkintro.dsp>. (Cited on page 30.)
- Sonwalkar, Nishikant [2002a]. *Demystifying Learning Technology Standards Part I: Development and Evolution*. <http://campustechnology.com/articles/2002/03/demystifying-learning-technology-standards-part-i-development-and-evolution.aspx>. (Cited on pages 4, 8, 15, 16, 17 and 18.)
- Sonwalkar, Nishikant [2002b]. *Demystifying Learning Technology Standards, Part II: Acceptance and Implementation*. <http://campustechnology.com/articles/2002/04/demystifying-learning-technology-standards-part-ii-acceptance-and-implementation.aspx>. (Cited on page 18.)
- Stojanovic, Ljiljana, Steffen Staab, and Rudi Studer [2001]. *eLearning based on the Semantic Web*. In *In WebNet2001 - World Conference on the WWW and Internet*, pages 23–27. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.16.295&rep=rep1&type=pdf>. (Cited on page 6.)

- The jQuery Project [2011]. *jQuery: The Write Less, Do More, JavaScript Library*. <http://jquery.com/>. (Cited on page 53.)
- The PHP Group [2011a]. *Function Reference*. <http://www.php.net/manual/en/funcref.php>. (Cited on page 56.)
- The PHP Group [2011b]. *History of PHP*. <http://www.php.net/manual/en/history.php.php>. (Cited on page 56.)
- The PHP Group [2011c]. *What has changed in PHP 5.0.x*. <http://www.php.net/manual/en/migration5.changes.php>. (Cited on page 56.)
- Varlamis, Iraklis, Alex Koochang, and Ioannis Apostolakis [2006]. *The Present and Future of Standards for E-Learning Technologies*. *Interdisciplinary Journal of Knowledge and Learning Objects*, 2, pages 59–76. <http://ijkl.o.org/Volume2/v2p059-076Varlamis.pdf>. (Cited on pages 6, 7, 15, 16 and 18.)
- Vossen, Gottfried and Peter Westerkamp [2008]. *Why service-orientation could make e-learning standards obsolete*. *International Journal of Technology Enhanced Learning*, 1, pages 85–97. ISSN 1753-5263. <http://inderscience.metapress.com/link.asp?id=q43n17857h48380h>. (Cited on pages 17 and 23.)
- W3C [1999]. *HTML 4.01 Specification*. <http://www.w3.org/TR/html4/>. (Cited on page 4.)
- W3C [2001]. *Introduction to CSS3 - W3C Working Draft*. <http://www.w3.org/TR/css3-roadmap/>. (Cited on page 50.)
- W3C [2002]. *Cascading Style Sheets, level 2 revision 1 CSS 2.1 Specification*. <http://www.w3.org/TR/2002/WD-CSS21-20020802/>. (Cited on page 50.)
- W3C [2005]. *Document Object Model (DOM)*. <http://www.w3.org/DOM/>. (Cited on page 31.)
- Wenz, Christian [2007]. *JavaScript und AJAX: Das umfassende Handbuch*. Seventh Edition. Galileo Computing, Bonn, Germany. ISBN 3898428591. (Cited on page 52.)
- Yergeau, F. [2003]. *UTF-8, a transformation format of ISO 10646*. URL: <http://www.ietf.org/rfc/rfc3629.txt>. Accessed: 2011-02-16. (Cited on page 50.)
- Young, Jeff [2008]. *Frustrated With Corporate Course-Management Systems, Some Professors Go 'Edu-punk'*. <http://chronicle.com/blogs/wiredcampus/frustrated-with-corporate-course-management-systems-some-professors-go-edupunk/3977>. (Cited on page 23.)
- Yu, Hua and Jianbo Fan [2009]. *Design and Implementation of the Framework for Adaptive e-Learning System*. In Wang, Fu, Joseph Fong, Liming Zhang, and Victor Lee (Editors), *Hybrid Learning and Education, Lecture Notes in Computer Science*, volume 5685, pages 140–149. Springer Berlin / Heidelberg. http://dx.doi.org/10.1007/978-3-642-03697-2_14. 10.1007/978-3-642-03697-2_14. (Cited on pages 3, 5 and 6.)
- Zend Technologies Ltd. [2010]. *Programmer's Guide - Zend Dojo*. <http://framework.zend.com/manual/en/zend.dojo.html>. (Cited on page 29.)

Glossary

ADL	Advanced Distributed Learning
AGR	AICC Guidelines and Recommendations
AICC	Aviation Industry CBT Committee
ARIA	Accessible Rich Internet Application
ARIADNE	Alliance of Remote Instructional Authoring Distribution Networks For Europe
ASI	Assessment Test, Section, and Item Information Model
CAM	Content Aggregation Model
CBT	Computer-Based Training
CMI	Computer Managed Instruction
DOD	Department of Defense
DOH	Dojo Objective Harness
DOM	Document Object Model
ECMA	Ecma International
IEEE	Institute of Electrical and Electronics Engineers
IMS	IMS Global Learning Consortium
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LMS	Learning Management System
LCMS	Learning Content Management System
LO	(Digital) Learning Object
LOM	Learning Object Metadata
LP	Learning Package
LTSC	Learning Technology Standards Committee
PIF	Package Interchange File
PEAR	PHP Extension and Application Repository
PECL	PEAR Extended Code Language
QTI	IMS Question and Test Interoperability Specification
ORM	Object Relational Mapping
RAID	Reusable Adjustable Interoperable and Durable
RIA	Rich Internet Application
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SN	Sequencing and Navigation
WBT	Web-based CBT
WYSIWYG	What You See Is What You Get
XAMPP	Cross-platform Apache MySQL PHP Perl
XUL	XML User Interface Language
ZK	ZK OpenSource Ajax Framework
ZK AU	ZK Auto-Update Engine
ZUL	ZK User Interface Language
ZUML	ZK User Interface Markup Language