VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Import adaptivní učebnice do systému Barborka
Import of Adaptive Textbook into System Barborka

2013 Lukáš Cholasta

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Cholasta**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Import adaptivní učebnice do systému Barborka
Import of Adaptive Textbook into System Barborka

Zásady pro vypracování:

Adaptivní učebnice je rozdělena na kapitoly, rámce a vrstvy, které jsou studentovi k dispozici v několika různých výukových stylech. Každá vrstva může být vytvořena až ve třech úrovních podrobnosti výkladu (hloubka) a čtyřech smyslových variantách (vizuální, auditivní, verbální a kinestetické). Výuku pomocí těchto učebnic a jejich tvorbu realizuje systém Barborka. Autoři učebnic ovšem často dávají přednost tvorbě učebnice v textovém editoru (především Microsoft Word) před tvorbou učebnice v systému Barborka. Do textového editoru jsou zadávány kromě vlastních výukových textů i metadata, která udávají ke každé části textu jeho příslušnost ke kapitole, rámci, vrstvě, výukovému stylu a další informace. Tento speciálně naformátovaný text je pak potřeba přepsat do systému Barborka, což se v současné době musí provádět ručně. Tuto činnost je ovšem možné automatizovat realizací programu, který by převedl naformátovaný text učebnice do datových struktur systému Barborka.

Cíle práce:
1. Seznamte se s problematikou adaptivní výuky.
2. Prostudujte strukturu dat adaptivní učebnice a její formátování v textovém editoru.
3. Seznamte se s možnostmi importu obsahu dokumentů MS Word.
4. Realizujte program, který bude možné používat na PHP serveru k automatickému převodu učebnice zadané formátovaným textem do systému Barborka a napište programátorskou a uživatelskou příručku k tomuto programu.

Seznam doporučené odborné literatury:

HOLUB, Libor. Automatizované řízení adaptivní výuky v e-learningu podle stylů učení studenta. Ostrava, 2011. 94 s. Dizertační práce. Vysoká škola báňská - Technická univerzita Ostrava.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Ondřej Takács**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Neubuzi dne 7.5.2013

Luboš Chloupek
.....
podpis autora

Rád bych poděkoval svému vedoucímu práce Ing. Ondřeji Takácsovi za cenné rady a připomínky, které mi během zpracování mé bakalářské práce poskytoval.

Abstrakt

Tato bakalářská práce se zabývá importem adaptivní učebnice do systému Barborka. Má za úkol zjednodušit a zrychlit vkládání dokumentu do e-learningového systému, do kterého se dokumenty vkládají pomocí webového formuláře, což je velmi zdlouhavá a pracná záležitost. V úvodu práce je popsáno, co to adaptivní výuka je, jak se v historii vyvíjela až do současné podoby systému Barborka. Dále je v práci popsána struktura dat adaptivní učebnice, tedy jak vypadá databáze s vkládanými daty. V práci jsou popsány, jaké se naskytly možnosti importu adaptivní učebnice do systému, jejich výhody a nevýhody a nakonec zvolený způsob řešení načítání dokumentu. Pro lepší pochopení tohoto načítacího algoritmu je k dispozici vývojový diagram. V další části je popsána příručka jak pro uživatele, tak pro programátora ve které jsou popisy některých použitých metod a funkcí. V závěru jsou zhodnoceny dosažené cíle práce a také návrhy dalších možností, jak by se tato práce dala rozšířit.

Klíčová slova

aplikace, buňka, databáze, dokument, funkce, klíč, lekce, metoda, předmět, objekt, odpověď, otázka, rámec, rozhraní, řádek, tabulka, učebnice, varianta, vrstva

Abstract

This bachelor thesis deals with import adaptive textbook in the Barborka system. It's task is to simplify and speed up loading a document into the e-learning system in which documents are inserted via a web form which is very time-consuming and laborious affair. The introduction describes what adaptive teaching them how ever developed to the present form of Barborka. The study also describes an adaptive data structure textbooks, that is how it looks with the inline data. The paper describes what he got an opportunity to import adaptive textbook system, their advantages and disadvantages, and ultimately selected method of document retrieval solutions. For a better understanding of a retrieval algorithm is a flowchart. The next section describes the manual for both users and programmers which are descriptions of some of the methods and functions. In conclusion, achievements are assessed work and suggestions for other ways would this work gave expand.

Key words

application, cell, database, document, function, key, lesson, method, subject, object, answer, question, framework, row, table, textbook, variant, layer

Seznam použitých symbolů a zkratek

API - Application Programming Interface

FTP – File Transfer Protocol

HTML – HyperText Markup Language

ID – Identification

IDE - Integrated Development Environment

JDK – Java Development Kit

LMS - Learning Management System

PHP - Hypertext Preprocessor

SDK – Software Development Kit

SQL - Structured Query Language

XML - Extensible Markup Language

OBSAH

1 Úvod **1**
2 Adaptivní výuka **2**
2.1 Historie 3
2.2 Technika a metodika 3
2.3 Systém Barborka 3
2.3.1 Autorská část LMS Barborka 4
2.3.2 Základní prvky pro vytváření studijních opor 4
3 Struktura dat adaptivní učebnice **7**
3.1 Tabulky 7
3.2 Číselník 7
3.3 Vazební tabulky 8
3.4 Systémové tabulky 8
3.5 Zjednodušený ER - Diagram databáze 8
3.6 Popis atributů tabulek databáze 8
4 Možnosti importu obsahu dokumentů Microsoft Word **10**
4.1 Převod dokumentu pomocí programu Antiword 10
4.2 Přímé načítání obsahu dokumentu pomocí Open Office API 11
4.3 Spouštění programu ve Webovém prostředí 11
4.3.1 Java Applet 11
4.3.2 Spouštění kódu z příkazové řádky 12
4.4 Open Office API 12
4.4.1 Software 12
5 Uživatelské rozhraní **13**
6 Uživatelská příručka **14**
6.1 Je-li aplikace nahraná na serveru 14
6.1.1 Potřebný software 14
6.2 Nahrání aplikace na server 14
6.2.1 Potřebný software 14
6.2.2 Instalace software 14
6.3 Importovaný dokument 15
6.4 Ovládání programu 16
6.5 Informační panel 16
7 Vývojový diagram **17**
7.1 Popis vývojového diagramu 20
8 Programátorská příručka **21**
8.1 Nástroje pro vývoj 21
8.2 Použité funkce Open Office API 22
8.3 Popis třídy ParseTable2 23
9 Závěr **26**
Seznam obrázků **27**
Seznam tabulek **28**
Seznam použité literatury **29**
Internetové zdroje **30**
Přílohy **31**

1 Úvod

V současné době jsou možnosti vkládání adaptivní učebnice do systému Barborka velmi zdlouhavé a nepraktické. Autor nebo uživatel, který se chystá vložit takovou učebnici do databáze systému Barborka, je nucen zdlouhavě vyplňovat webové formuláře a složitě vkládat a zařazovat jeho části do různých oddělení databáze. Tato možnost může být zejména nepříjemná, pokud je text učebnice obsáhlý a existuje v ní mnoho různých částí, které musí uživatel přesně rozlišit a vložit do správné části databáze tak, aby správně fungovaly a měli požadovaný efekt, jaký si uživatel představoval.

Aplikace této bakalářská práce by měla vytvořit snadnou cestu ke vkládání takovýchto dokumentů do systému bez toho, aby uživatel musel hodnoty jednu po druhé vkládat do databáze přes webový formulář. Uživateli bude stačit, když si pouze připraví dokument, který bude splňovat všechny dohodnuté náležitosti. Dokument má podobu jedné až několika tabulek, které obsahují logicky uspořádané části adaptivní učebnice. Pak bude jen stačit vložit jej do systému a tato aplikace by se již o zbytek měla postarat.

2 Adaptivní výuka

Adaptivní učení je vzdělávací metoda, která využívá počítače jako interaktivních výukových zařízení. Přizpůsobuje počítačové prezentace studijních materiálů podle vzdělávacích potřeb studentů, jako odpovědi na jejich na otázky a zadané úkoly. Tento model pochází od radikálního behavioristy z období kolem roku 1950 a z nerealizovaných příslibů stroje Teaching Machine B. F. Skinnera.[i2] Teaching Machine bylo mechanické zařízení, jehož účelem bylo poskytnout učební plán programované výuky. Motivací je umožnit elektronickou výukou, začlenit hodnotu interaktivity poskytované studentovi, skutečným lidským učitelem nebo vychovatelem. Tato technologie zahrnuje aspekty odvozené z různých oborů, včetně počítačové vědy, vzdělávání a psychologie.[i3]

Adaptivním učením bylo částečně učiněno poznání, že učení na míru nelze dosáhnout ve velkém měřítku s využitím tradičních, neadaptivních přístupů. Adaptivní učební systémy usilují o transformaci studenta před pasivním receptorem informací spolupracovníka do vzdělávacího procesu.[4] Primárními adaptivními systémy učení jsou aplikace z oblasti vzdělávání, ale i další populární aplikace jako jsou obchodní školení. Byly navrženy jako stolní počítačové aplikace a webové aplikace.[i3]

Adaptivní učení je rovněž známo jako adaptivní vzdělávací hypermedia, počítačové vzdělávání, adaptivní výuka, doučování inteligentními systémy a počítačové bázové pedagogické prostředky.[i3]

Výukové opory pro adaptovanou výuku

Aby bylo možno sestavovat výuku na míru osobním charakteristikám studentů, musí mít systém možnost manipulace s výukovými oporami. Úkolem je naučit všechny studenty stejným výsledným znalostem a dovednostem, ale každého případně s jiným přístupem, jiným způsobem nebo postupem. To znamená, že bude potřeba výukové materiály zpracovat buď v mnoha různých variantách pro „typické“ studenty, ovšem s rizikem, že až přijde jiný typ studenta, budou se varianty dále a dále rozmnožovat.

Jinou možností je materiály podrobně rozčlenit a výuku z těchto částí vhodně sestavovat. Uvědomme si, že fakta předkládaná k výuce jsou stále stejná, jen podrobnost jejich komentování, podrobnost průběžných pedagogických pokynů, pořadí jejich výkladu a dalších průvodních částí výkladu je rozdílná. Pokusíme se tedy navrhnout takové rozdělení výukové látky na části, aby z částí byly sestavitelné nejrůznější varianty výuky.

Podmínkou samozřejmě je znalost důležitých charakteristik studenta, takových, které mají vliv na jeho proces výuky, na jeho učební styl.

Hlavním úkolem této příručky je popsat podrobnou strukturu – rozčlenění výukových materiálů tak, aby se s nimi dalo libovolně manipulovat. Ostatní kapitoly jsou doplňující, stručně vysvětlují souvislosti - důvody strukturalizace výuky se strany studenta i proces řízení výuky automatickým učitelem.

Inteligentní řízení adaptované výuky

I když budou výukové opory vhodně strukturovány, i když budeme o aktuálním studentovi znát jeho učební charakteristiky, nestačí to k optimální výuce. Je nutné vědět, jak má být který student výukou prováděn, aby spotřebovaný čas i kvalita výsledných znalostí byla nejlepší.

V praktické „živé“ výuce by to byl úkol pro emaptického, zkušeného učitele, který by volil vždy nejvhodnější individuální přístup ke každému studentovi. Předpokládáme, že nejde o frontální výuku ve třídě, ale o soukromou individuální výuku.

V e-learningovém prostředí je nutné tuto pedagogicko-psychologickou zkušenost zabudovat do řídicího systému adaptivního LMS. Nazveme tuto funkci systému virtuálním učitelem. Protože současné LMS takovéto funkce obvykle nemaji, musí se ve spolupráci s programátory do LMS dobudovat.

V následujícím odstavci si popíšeme teoreticky činnost virtuálního učitele a jeho okolí podrobněji.

2.1 Historie

Adaptivní učení nebo-li inteligentní výuka má svůj původ v umělé inteligenci pohybu a začala získávat popularitu v roce 1970. V té době, to bylo obecně přijímáno, že počítače by nakonec mohly dosáhnout lidské schopnost adaptability. V učení adaptivním, je základním předpokladem, že nástroj nebo systém bude schopen přizpůsobit studentovým/ uživatelovým vyučovacím metodám, což má za následek lepší a efektivnější vzdělávací zkušenosti pro uživatele. Již v 70. letech byla hlavní překážkou cena a velikost počítačů, které skýtalo toto uplatnění nepraktickým. Další překážkou v přijetí prvních inteligentních systémů bylo, že uživatelské rozhraní nepřispívalo k procesu učení.[i3]

To se nestalo dokud „AutoTutor“, který byl vyvinut Institutem Intelligent System na přelomu století, adaptivní učební systém dostal hlas. To byl významný krok u adaptivních systémů učení, protože bylo přidáno další médium v komunikaci s koncovým uživatelem. Podle zakladatele a vedoucího projektu AutoTutor – p. Graessera "Může výpočetní prostředí podporovat sociální vztahy, které mohou zlepšit učení." Také v některých aplikacích je zvukový obsah nutností, například v aplikacích pro jazyky. V současné době počet nových adaptivních systémů vzdělávání společnosti neustále roste stejně jako učeben s počítačovým vybavením a i další odvětví nacházejí využití pro aplikace adaptivního učení, pro profesní rozvoj.[i3]

2.2 Technika a metodika

Adaptivní learningové systémy byly tradičně rozděleny do samostatných složek nebo-li modelů. Zatímco různé modelové skupiny byly prezentovány, většina systémů zahrnují některé nebo všechny z těchto modelů (někdy s různými názvy): [i3]

- Expert model - model s informací, které mají být vyučovány
- Student model - model, který sleduje a učí se o studentovi
- Instruktážní model - model, který skutečně vyjadřuje informace
- Vzdělávací prostředí - uživatelské rozhraní pro interakci se systémem

2.3 Systém Barborka

LMS Barborka je e-learningový systém, tedy programový systém pro podporu všech částí výuky: přípravu výukových opor, vlastní výuku, testování znalostí, vedení potřebných evidencí.[3]

Historie Barborky se datuje od roku 1982, kdy vznikl návrh struktury databáze i první implementace základních funkcí autorských a studentských. První verze byla napsána v jazyce Fortran pro počítače SMEP, druhá v relační databázi Redap s pascalovskými moduly pro lokální síť PC, třetí verze v jazyce Delphi s databází formátu DB. Teprve současná internetová verze Barborky však plně umožňuje využít navíc multimediální prvky, komunikační možnosti internetu a tím i plnohodnotnou e-learningovou výuku a její řízení.[3]

Vývoj současné verze začal v roce 2001 na FEI VŠB-TU Ostrava spoluprací 2 diplomantů oboru Inženýrská informatika. Oba diplomanti po absolvování dále na dalším vývoji systému pracují na katedře informatiky FEI s podporou transformačních a rozvojových programů. Na vývoji spolupracují i další diplomanti a doktorandi katedry.[3]

2.3.1 Autorská část LMS Barborka

LMS Barborka se začala na VŠB–TU Ostrava používat začátkem roku 2003. Pro pilotní testování autorské části byla oslovena řada autorů z celé univerzity. Nabídli jsme autorům plnou podporu a pomoc při počáteční tvorbě studijních opor, protože jsme předpokládali určitou nedůvěru ze strany autorů, jelikož Barborka ve své internetové podobě byla novým, teprve se rodícím LMS systémem. [2]

2.3.2 Základní prvky pro vytváření studijních opor

Podle známých pravidel pro vytváření studijních opor je vhodné členit text do malých dávek, odstavců, obsahujících 1–2 nové základní informace pro studenta. Druhým důvodem pro rozčlenění textu je možnost vícenásobného využití jednotlivých komponent v různých výukových oporách. Následující struktura autorské databáze Barborky vychází z těchto požadavků a umožňuje autorům tvořit výukové opory podobně jako stavět různé celky ze stavebnice základních prvků.[2]

Struktura autorské části databáze Barborky je následující:

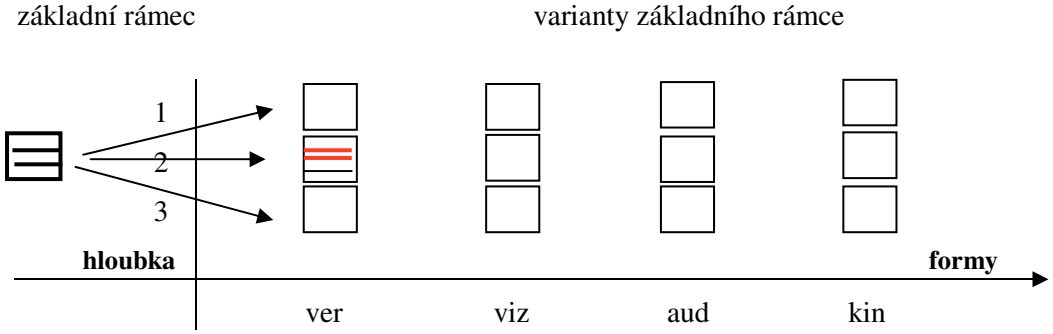
P**ŘEDMĚT** – kompletní předmět může být tvořen jak lekcemi, tak kapitolami. Vhodnou kombinací těchto dvou prvků lze docílit dostatečné interaktivity, vhodné pro distanční vzdělávání.[2]

LEKCE – je speciální výuková struktura určená pro programovanou výuku. Je tvořena nelineární strukturou výkladových rámců a otázek. Vhodným uspořádáním rámců a otázek může autor dosáhnout optimální formy dynamické interaktivní výuky. Lekce se dá zobrazit grafem, v němž rámce a otázky tvoří uzly, možné přechody mezi nimi jsou znázorněny hranami. Studentovi jsou předkládány výkladové rámce, z nichž studuje, a průběžně mu jsou pokládány otázky na probrané téma. Podle reakce studenta se buď zobrazí další do vysvětlující rámce nebo příklady. Jestliže student odpoví na otázku správně, není dalším výkladem zdržován a je mu zobrazen následující výkladový rámec. Vhodným návrhem lekce může autor dosáhnout vysokého stupně adaptability výuky.[2]

RÁMEC je elementární část lekce, obsahující jednotkovou výukovou informaci; na této úrovni se analyzují jeho varianty a vrstvy. Tzv. základní rámec definuje jeho obsahovou náplň, varianty rámce se liší jen formou nebo hloubkou výkladu, ne obsahem.[6]

VARIANTY rámce jsou jiné způsoby výkladu a ověřování téže látky.
Dle úvahy v předcházejícím odstavci navrhujeme až 4 varianty dle preferovaného smyslového vnímání studenta (nazývané dále též 4 smyslové formy variant) a až 3 varianty z hlediska hloubky výkladu. Celkem tedy může být až 4 x 3 = 12 variant ve dvou dimenzích, formě a hloubce, viz obrázek 2.

Není nutné využít vždy všechny varianty. Je na autorovi opory, aby pokryl smysluplné varianty nebo některé nevyužil, pokud to není vhodné nebo potřebné.[6]



Obr. 1 Variantní rámce dle smyslové formy

- **Varianty rámce dle smyslové formy**

Z hlediska formy dělíme varianty do již uvedených čtyř typů (sloupec variant ve výše uvedeném schématu):

- **Verbální** – varianta obsahuje převážně text,
- **Vizuální –** varianta obsahuje mnoho obrázků, grafů, animací apod.,
- **Auditivní** – varianta obsahuje hodně mluveného slova, zvuků, videopřednášek apod.,
- **Kinestetická** – varianta obsahuje hodně interaktivních výukových programů apod.

Málokdy varianta patří čistě jedné formě. Většinou jde o kombinaci forem a pak autor určí procentuální poměr jednotlivých forem, udávající použité procento každé formy. Podle převažující hodnoty bude varianta zařazena na příslušné místo ve výše zobrazené „matici variant“.[6]

- **Varianty rámce dle hloubky**

Hloubka výkladu udává míru podrobnosti výkladu, specifikuje detailnost předkládaných výukových informací. Prozatím definujeme 3 úrovně hloubky.[6]

Základní úrovní je hloubka 2. Zde je nejčastěji používaný výklad z hlediska podrobnosti. Jeho rozsah a obsah určuje autor. Varianta může obsahovat i teoretické otázky nebo úlohy k řešení. Pomocí nich si systém ověřuje, zda student látku pochopil. Pokud jsou odpovědi správné, systém nabízí další informaci (další část rámce nebo další rámec) na stejné úrovni hloubky.[6]

Pokud student neodpovídá správně, nabídne mu systém v hloubce 3 podrobnější výklad nejprve s jednoduššími a postupně se složitějšími příklady. Také otázek může být více po menších celcích.[6]

Naopak studentovi výbornému, rychle chápajícímu, může systém nabídnout v hloubce 1 rozšiřující informace, souvislosti a vazby na jiné oblasti apod.[6]

KAPITOLA – je tvořena lineární posloupností jednotlivých výkladových rámců, zakončena je obvykle testem složeným z řady otázek na probranou látku. Kapitola může být libovolně dlouhá o libovolném počtu rámců. Bývá koncipována stejně jako kapitola v klasických tištěných skriptech nebo učebnicích. Je vhodná také k tisku.[2]

VRSTVA – Varianty lišící se jen formou a hloubkou výkladu nestačí pokrýt všechny potřebné rozdílnosti ve výkladovém stylu. Výklad musí reagovat i na další rozdílné osobní vlastnosti studentů. Analýzou těchto studentských vlastností jsme došli k výsledku, že se výklad liší také pořadím dílčích částí výkladu a průběžného testování, případně organizačních informací.[6]

Provádět adaptaci výkladového stylu rámce nám umožní rozdělení rámce na dílčí části - na vrstvy. **Vrstvou** rámce nazýváme část rámce homogenní z hlediska fází vyučovacího procesu (výklad teorie, vysvětlování, upevňování, prověřování vědomostí, motivace, řízení výuky).

Typy vrstev:

- Výkladové – skupina vrstev obsahující vlastní výklad probírané látky. Jde o tyto vrstvy:

T Teoretická – obsahující teorii: definice, pojmy, pravidla, algoritmy atd. Z hlediska výuky se jedná o nejdůležitější typ vrstvy.

S Sémantická – vysvětlující zaváděné pojmy, formálně popsanou teorii, obsahuje doplňující informace k teoretické vrstvě, vysvětluje souvislosti plynoucí z teorie atd.

F Fixační – pomocí opakování, jiných formulací a alternativních pojmů, zasazením do širšího kontextu usnadnit lepší zapamatování teorie.

R Řešené příklady – obsahuje příklady na využití teorie, řešené „školní" příklady. Jsou s studentovi vzorem k řešení jemu předložených úkolů.

P Praktická – obsahuje řešení příkladů z praxe, které využívají teoretické znalosti.

- Testovací – skupina vrstev pro průběžné testování získaných znalostí a za pomocí úloh k řešení tyto teoretické znalosti zafixovat, získat praktické dovednosti. Každá otázka nebo úloha obsahuje nejen formulaci zadání, ale i jednoznačně rozpoznatelný výsledek.[6] Jde o vrstvy:

Otázky - teoretické otázky z probrané látky. Otázky mohou sloužit jen jako kontrola studentovi nebo je využije adaptivní algoritmus k řízení dalšího výkladu.[6]

U Úlohy – „školní" úlohy k řešení.

X Praktické úlohy – úkoly z praxe.

- Ostatní

C Cíle – formulované cíle lekce nebo rámce, obdobně jako v distančních oporách.

M Motivační – motivující informace o předmětu, lekci nebo rámci, které by nemotivovanému studentovi zdůvodnily přínos studia.

N Navigační – informace pedagogické, organizační, jakýsi průvodce lekcí nebo probíranou látkou, doporučený postup při studiu apod.

L Literatura – doporučená literatura

Informace o formě a hloubce výkladu a typu vrstvy je nutno evidovat v metadatech. Pomocí metadat pak systém může vybírat a řídit správné pořadí výuky.[6]

Na základě konkrétních vlastností studenta je možno změnou pořadí typu vrstev měnit výkladový styl rámce. Při tomto typu adaptace neztrácí rámec svou obecnou výkladovou hodnotu. Řízení výkladu se provádí výběrem smyslové formy a potom volbou pořadí a hloubky vrstev. Tím dostáváme univerzální možnost výukovou oporu libovolně adaptovat.[6]

Multivrstvy

Někdy může autor rozložit i jednu vrstvu do více částí stejného typu, ty pak tvoří tzv. multivrstvu. Například v rámci popisuje 3 nové pojmy, které spolu souvisejí a proto nejsou rozděleny do 3 rámců. Pak může sestavit vrstvy T1, T2, T3, S1, S2, S3, … Zvláště u testovacích vrstev se může vyskytovat více otázek nebo úloh uvnitř rámce. Ty také tvoří multivrstvu otázek nebo úloh.[6]

Proto se u vrstvy zadává také pořadí vrstvy v multivrstvě; toto pořadí nebude systém měnit, aby zachoval případnou návaznost pojmů. Může ale měnit – stejně jako u vrstev – pořadí použitých typů vrstev podle typu studenta například na T1, S1, T2, S2, … nebo S1, S2, S3, … , T1, T2, T3 apod.[6]

KOMPONENTY jsou části výukové opory. Dělí se na texty (včetně obrázků), otázky, multimédia. Každá komponenta může být využita vícekrát v různých variantách nebo vrstvách, každá varianta nebo vrstva může být vyskládána z více komponent. Je definována svým id, typem komponenty a vlastním obsahem.[6]

Metadata výukové opory

Již jsme se zmínili, že mimo vlastní výukové části - komponenty (texty, otázky, multimédia apod.) musí systém evidovat i informace o nich: o rozdělení opory předmětu do kapitol či lekcí, do rámců, o typech variant rámců a typech vrstev. Těmto informacím o výukových komponentách říkáme metadata, tj. data o datech. Teprve pomocí těchto metadat může systém vybírat a sestavovat správně varianty i pořadí výkladu a testování pro konkrétního studenta.

Předmět se dělí na lekce (vyučovací hodiny), lekce na rámce (jednotkové informace). Rámec má několik variant smyslových a hloubkových. Každá varianta může mít více vrstev. Vrstvy testovací mohou mít více otázek a úkolů, každá otázka nebo úkol může mít více očekávaných odpovědí.[6]

3 Struktura dat adaptivní učebnice

Databázové tabulky systému Barborka lze rozdělit do čtyř kategorií:

- **Tabulky**
- **Číselníky**
- **Vazební tabulky**
- **Systémové tabulky**

Popis tabulek databáze

3.1 Tabulky

- **au_predmet – Seznam předmětů v databázi**
- **au_lekce – Seznam lekcí v databázi**
- **au_ramec – Seznam rámců v databázi**
- **au_varianta – Seznam variant v databázi**
- **au_kapitola – Seznam kapitol v databázi**
- **au_tvrstva – Seznam testovacích vrstev v databázi**
- **au_vrstva – Seznam vrstev v databázi**
- **au_odpoved – Seznam odpovědí v databázi**
- **au_kompon – Seznam komponent v databázi**

3.2 Číselník

- **au_ckomp – Seznam typů komponent**
- **au_codpo – Seznam typů odpovědí**
- **au_cotaz – Seznam typů otázek**
- **au_cvrstva – Seznam typů vrstev**

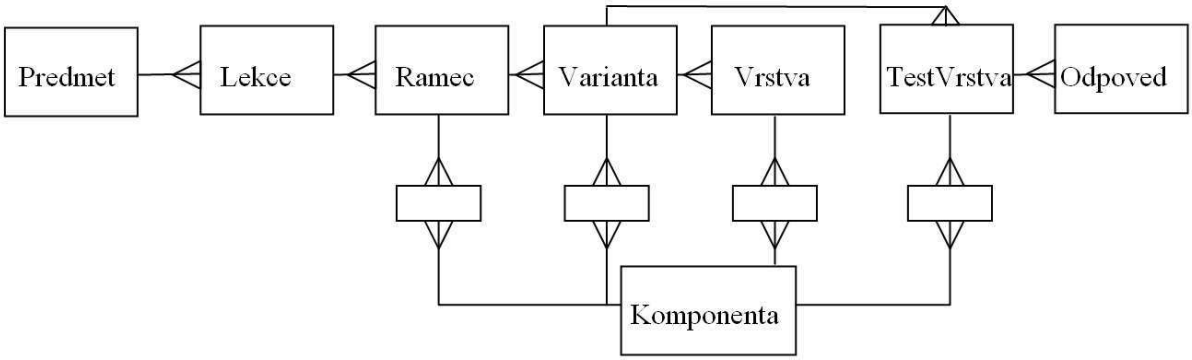
3.3 Vazební tabulky

- **au_xautor – Seznam vazeb předmětů a jejich autorů**
- **au_xram – Seznam vazeb rámců s jedinými komponentami**
- **au_xvar – Seznam vazeb variant s jedinými komponentami**
- **au_xvrs – Seznam vazeb vrstev s komponentami**
- **au_xtvr – Seznam vazeb testovacích vrstev s komponentami**

3.4 Systémové tabulky

- **sy_soubor – Seznam nahraných souborů**
- **sy_uziv – Seznam uživatelů systému**

3.5 Zjednodušený ER - Diagram databáze



Obr. 2 Zjednodušený ER – Diagram databáze

Podrobnější ER-Diagram je uveden v příloze (Obr. 1 ER – Diagram databáze)

3.6 Popis atributů tabulek databáze

Kompletní popis všech tabulek a atributů databáze je uveden v příloze

Všechny tabulky a číselníky databáze, které budou popsány obsahují tři atributy, které mají ve všech tabulkách stejný význam. Proto tyto atributy nebudu uvádět u každé tabulky zvlášť, ale popíšu je pouze na začátku.
Jsou to:

Název: dat_zme
Datový typ: timestamp
Popis: Obsahuje datum aktualizace záznamu, tedy datum a čas, ve který byl tento záznam naposled změněn.

Název: dat_zap
Datový typ: timestamp
Popis: Obsahuje datum a čas zápisu záznamu nebo-li přesné datum a čas kdy byl do databáze záznam vložen.

Název: kdo_zme
Datový typ: char(10)
Popis: Obsahuje ID uživatele, který provedl poslední aktualizaci záznamu.

Tabulky

Tabulka au_predmet

Tabulka obsahuje záznamy všech předmětů v databázi.

Atributy

Název: id_pred
Datový typ: char (10)
Popis: Jednoznačný identifikátor předmětu. Pro tento atribut je v databázi vytvořen trigger, který zajišťuje jeho automatické generování při vkládání záznamu.

Název: nazev_p
Datový typ: varchar (130)
Popis: Název předmětu.

Název: zkrat_p
Datový typ: varchar (20)
Popis: Zkratka předmětu.

Název: uroven_p
Datový typ: int (2)
Popis: Úroveň předmětu.

Tabulka au_lekce

Tabulka obsahuje záznamy všech lekcí v databázi.

Atributy

Název: id_lekce
Datový typ: char (10)
Popis: Jednoznačný identifikátor lekce. Pro tento atribut je v databázi vytvořen trigger, který zajišťuje jeho automatické generování při vkládání záznamu.

Název: nazev_l
Datový typ: varchar (130)
Popis: Název lekce.

Název: id_pred
Datový typ: varchar (20)
Popis: Cizí klíč, identifikátor navázaného předmětu.

Název: id_kapi
Datový typ: varchar (20)
Popis: Cizí klíč, identifikátor nadřazené kapitoly

Název: uroven_l
Datový typ: int (2)
Popis: Úroveň lekce.

Název: poradi_l
Datový typ: int (3)
Popis: Pořadí lekce v předmětu.

Tabulka au_ramec

Tabulka obsahuje záznamy všech rámců v databázi.

Atributy

Název: id_ramec
Datový typ: char (10)
Popis: Jednoznačný identifikátor rámce. Pro tento atribut je v databázi vytvořen trigger, který zajišťuje jeho automatické generování při vkládání záznamu.

Název: nazev_r
Datový typ: varchar (130)
Popis: Název rámce.

Název: id_lekce
Datový typ: varchar (20)
Popis: Cizí klíč, identifikátor navázané lekce.

4 Možnosti importu obsahu dokumentů Microsoft Word

Mým úkolem bylo najít vhodný způsob, jak by šla moje aplikace spustit na webovém PHP serveru a současně najít vhodný program, který by dokázal přečíst obsah dokumentu.

Jako kód, který by se dal spustit na straně serveru, se nabízel jazyk PHP s dalším využitím Java Appletu a JavaScriptu. Hlavním problémem však zůstávalo, jak převést obsah dokumentu do nějaké čitelné podoby pro tyto programovací jazyky.

První myšlenkou jak zpracovat obsah dokumentu bylo načíst jej ve formě textu, PostScriptu nebo XML. Pokud by byl převod do těchto textových formátů bez chyb a dobře čitelný, byla by to dobrá cesta k jeho načítání.

4.1 Převod dokumentu pomocí programu Antiword

Antiword je svobodný software ke čtení Microsoft Word dokumentů a je k dispozici pro většinu počítačových platforem. Programem Antiword lze převést dokumenty z Microsoft Word verze 2, 6, 7, 97, 2000, 2002 a 2003 na prostý text, PostScript, PDF a XML.

Tato možnost se jevila jako dobré řešení pro převod dokumentu. Ale pro nevhodný převod tabulek a problémy se spuštěním tohoto programu v operačním systému Windows, byla tato možnost zavrhnuta. Navíc v červenci roku 2007 byl ukončen vývoj a aktualizace tohoto programu.

4.2 Přímé načítání obsahu dokumentu pomocí Open Office API

Další možností, která se nabízela, bylo načítat přímo obsah dokumentu tak jak je strukturován v kódu a vkládat jej do databáze. Tím by odpadlo hledání dalšího programu pro převod dokumentu do jiné textové podoby.

Pro tuto možnost se jako dobrá volba jevilo API, které nabízel program Open Office. Programování s tímto API šlo realizovat ve více jazycích (C++, Java, Python). Díky dobré podpoře jazyka Java a mé oblibě jsem si ho nakonec zvolil pro programování mé bakalářské práce.

4.3 Spouštění programu ve Webovém prostředí

Protože má aplikace měla fungovat ve webovém prostředí, aby uživatelé mohli pohodlně za pomocí webového prohlížeče importovat dokumenty, hledal se vhodný způsob, kterým by program šel ve webovém prostředí spustit.

4.3.1 Java Applet

Applet je program v jazyku Java, který pro svůj běh vyžaduje Java-kompatibilní prohlížeč. Nespouští se přímo (jako aplikace), nýbrž otevřením HTML dokumentu, kde je na něj umístěn odkaz pomocí speciální značky „<APPLET>".

Vlastnosti Java Appletu:

- Applet nemůže nahrávat knihovny ani definovat nativní metody.
- Applet nemůže navazovat síťové spojení na jiný než domovský server.
- Applet nemůže zapisovat do souborů na straně klienta (prohlížeče).
- Applet nemůže spouštět programy na domovském serveru.
- Applet nemůže číst některé systémové proměnné.
- Applet může přehrávat zvuky
- Applet může požádat browser o zobrazení libovolné WWW stránky
- Applet může volat veřejné metody appletů umístěných na téže WWW stránce

Moje aplikace potřebovala pracovat s knihovnami open office. To byl hlavní důvod, proč jsem se pro Java Applet nerozhodl. Neumožňuje totiž přímo nahrávat knihovny, tudíž by musely být zahrnuty v aplikaci, což by způsobilo její dlouhé načítání. U Java Appletů nastávají v poslední době také problémy s bezpečností.

4.3.2 Spouštění kódu z příkazové řádky

Nakonec se jako nejvhodnější způsob jevila možnost spouštění aplikace z příkazové řádky za pomocí PHP kódu, který bude zabudován v HTML stránce. Využívá se funkce exec( ), kterou PHP nabízí. Tato funkce umožňuje text, který je jí dán na vstupu, spouštět přímo v příkazové řádce. Tímto textem je na mysli cesta ke zdroji na serveru, kde bude umístěna moje aplikace.

4.4 Open Office API

OpenOffice.org nabízí jazykově nezávislé API (aplikační programovací rozhraní), které umožňuje naprogramovat funkce v různých programovacích jazycích (např. C + +, Java, Python, CLI, StarBasic, JavaScript, OLE). To umožňuje používat OpenOffice.org jako poskytovatele služeb v jiných aplikacích, rozšíření o nové funkce nebo jednoduše upravovat a řídit OpenOffice.org.

Popularita standardizovaného (OASIS a ISO / IEC 26300), Open Document Format pro kancelářské aplikace roste. To také zvyšuje popularitu OpenOffice.org obecně. Firemní uživatelé často vyžadují integraci kancelářských funkcí do stávajících procesů a aplikací. Také často vyžadují dodatečné funkce nebo speciální vlastní nastavení stávajících funkcí. A to je jeden z hlavních cílů projektu API. Poskytnout možné přizpůsobení nebo kontrolu nad prací, které se vejde do vašeho stávajícího prostředí a bude naplňovat vaše zvláštní požadavky.

Pro programování mé aplikace bakalářské práce jsem zvolil jazyk Java, protože pro něj autoři Open Office API poskytovali dostatek podpory, rad a návodů, jak pomocí něj s API pracovat.

4.4.1 Software

Následující softwarové komponenty jsou potřebné pro práci s OpenOffice.org API v programovacím jazyku Java za použití vývojového prostředí **NetBeans**.

JDK

Java Development Kit (JDK) je produktem Oracle Corporation, který obsahuje soubor základních nástrojů pro vývoj aplikací pro platformu Java.

Java aplikace pro OpenOffice.org 2.0 vyžadují Java Development Kit 1.3.1 nebo novější. Pokud máme 64-bitový operační systém, musíme si stáhnout 32-bitovou verzi JDK pro použití s OpenOffice.org. Chceme-li získat všechny funkce, je zapotřebí alespoň Java v. 1.4.1_01. Doporučené je však používat vždy nejnovější oficiální verzi Javy, z důvodu důležitých oprav chyb.

Java IDE

Jako integrované vývojové prostředí pro práci v jazyku Java jsem zvolil prostředí NetBeans od společnosti Sun Microsystems. Mohl jsem použít i jiné IDE, ale NetBeans / Sun ONE Studio nabízí nejlepší integraci. Integrace OpenOffice.org s IDE jako je NetBeans úspěšně pokračuje a stále se zlepšuje.

OpenOffice.org API plugin

OpenOffice.org API plugin pro NetBeans zjednodušuje vývoj a rozšíření OpenOffice.org. Můžeme použít OpenOffice.org API pro program OpenOffice.org, a automatizovat dálkové nebo procesních úkoly a rozšířit OpenOffice.org o úplně nové funkce. OpenOffice.org API plugin pro NetBeans zjednodušuje přístup a používání API v nových projektech a zjednodušuje vytváření kompletních OpenOffice.org rozšiřovacích balíčků (srovnatelných s NetBeans Plugin moduly).

OpenOffice.org

OpenOffice je open-source kancelářský software pro zpracování textu, tabulek, prezentací, grafiky, databází a další. Je k dispozici v mnoha jazycích a funguje na všech běžných počítačích. Ukládá všechna data do mezinárodního otevřeného standardního formátu, a umí číst a zapisovat soubory z dalších běžných balíků kancelářského software. Je možné ho stáhnout a používat zcela zdarma k jakémukoli účelu. Je také potřebný k programování v OpenOffice API.

OpenOffice.org Software Development Kit

Vývojový set, který nám umožní, aby OpenOffice.org API plugin používaný ve vývojovém prostředí NetBeans spolupracoval s OpenOffice.org.

5 Uživatelské rozhraní

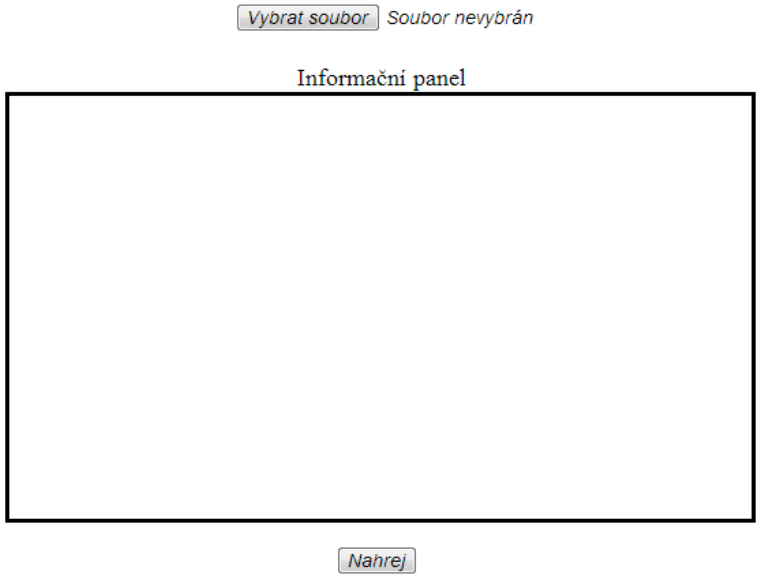
Uživatelské rozhraní je jednoduché, ale navrhnuto tak, aby splňovalo všechny požadavky uživatele. Tyto požadavky jsou možnost výběru souboru z lokálního umístění v počítači a výpisu zprávy o proběhnutí importu dokumentu, ať už kladné nebo záporné.

Uživatelské rozhraní obsahuje tyto prvky:

- **Tlačítko „Vybrat soubor“** – uživateli se po stisknutí tohoto tlačítka otevře okno, ve kterém vybere soubor ze svého počítače. Tento soubor musí být dokument aplikace Microsoft Word nebo Open Office Document. Vkládaný dokument musí být také správně naformátován, aby mohlo proběhnout korektní načtení jeho prvků do databáze.

- **Informační panel** - reprezentuje místo pro výpis textu o úspěchu či neúspěchu proběhnutí aplikace. Uživateli se zde objeví zprávy typu informativního, které informují o načtených částech dokumentu. Dále zprávy varovné, které neukončí běh programu, ale upozorní na určité nedostatky ve formátování dokumentu. Nakonec se zde mohou objevit zprávy kritické, které uživatele upozorní, že dokument nebyl korektně načten a bude potřeba jeho úprava uživatelem.

- **Tlačítko „Nahrej“** – Tímto tlačítkem uživatel spustí aplikaci, která začne rozebírat dokument a vkládat ho do systému.



Obr. 3 Uživatelské rozhraní

6 Uživatelská příručka

6.1 Je-li aplikace nahraná na serveru

Je-li aplikace nahraná na serveru, uživatel nemusí instalovat žádný software.

6.1.1 Potřebný software

K provozu aplikace uživateli postačí mít webový prohlížeč a připojit se na server kde se aplikace nachází.

6.2 Nahrání aplikace na server

Server, na kterém chce uživatel aplikaci provozovat, musí podporovat skriptování PHP a databázi MySQL s povoleným vkládáním Triggerů.

6.2.1 Potřebný software

Vše potřebné ke zprovoznění aplikace na serveru je obsaženo na přiloženém CD k této bakalářské práci.

6.2.2 Instalace software

Uživatel obsah složky, jménem „server“, obsažené na přiloženém CD, zkopíruje na server. Zkopírovat soubory může například pomocí FTP Clienta FileZilla, přiloženého taky na CD ve složce software. Dále musí soubor „barborka4.sql“ uživatel importovat do MySQL databáze na serveru.

Poté uživatel spustí soubor index.php a zobrazí se uživatelské rozhraní aplikace.

6.3 Importovaný dokument

Dokument, který bude uživatel vkládat do systému, musí být dokument společnosti Microsoft Word, tedy dokument s příponou „doc“. Tento dokument musí obsahovat strukturalizovaný formulář, který byl pro tvorbu e-learningových opor, navržen.

Rnazev = Název rámce					RPor = **n**		
	Varianta - hloubka:				VHloub = **1-3**		
	Varianta - forma:				VForm=ver,viz,aud,kin		
VrImpl	**Obsah vrstvy**					**VrTyp**	**VrPor**
	1. text vrstvy T (text, obrázek, …)					**T**	**1**
	2. text vrstvy T					**T**	**2**
	text vrstvy S					**S**	**1**
	…					**S**	**2**
	text vrstvy F					**F**	**1**
	Příklad: …					**R**	**1**
	Praktický příklad: …					**P**	**1**
	text vrstvy C					**C**	**1**
	text vrstvy L					**L**	**1**
	text vrstvy N					**N**	**1**
	text vrstvy M					**M**	**1**
	Název otázky 1	VrSk=**1**	VrPov=**P**	VrBod=**1**	VrVyh=**P**	**O**	**1**
	formulace otázky … variantní 1.skupiny						
					OBod	**OTyp**	**OSpr**	**OPor**
	text varianty 1				**0**	**Vnm**	**N**	
	…				**0**		**N**	
	Název úlohy 1	VrSk=**2**	VrPov=**P**	VrBod=**3**	VrVyh=**P**	**U**	**1**
	Formulace zadání úlohy… tvořená 2.skupiny						
					OBod	**OTyp**	**OSpr**	**OPor**
	Odpověd 1 – správná				3	Txx	**A**	**1**
OReak	Text netypické slovní reakce na odpověď 1						
	Odpověd 2 - částečně správná				0	Txx	**N**	**2**
OReak	Text netypické slovní reakce na odpověď 2, vysvětlení						
	Odpověd 3 – chybná, často se vyskytující				0	Txx	**N**	**3**
OReak	Text slovní reakce na odpověď 3, vysvětlení						

Tab. 1 Formulář pro tvorbu e-learningových opor

6.4 Ovládání programu

Pro import dokumentu musí být uživatel ve svém webovém prohlížeči na stránkách serveru s rozhraním pro import dokumentu.

Chce-li uživatel naimportovat dokument, klikne na tlačítko „Vybrat soubor“ a vybere dokument umístěný ve svém počítači.

Po kliknutí na tlačítko „Nahrej“, se provede import dokumentu do systému.

6.5 Informační panel

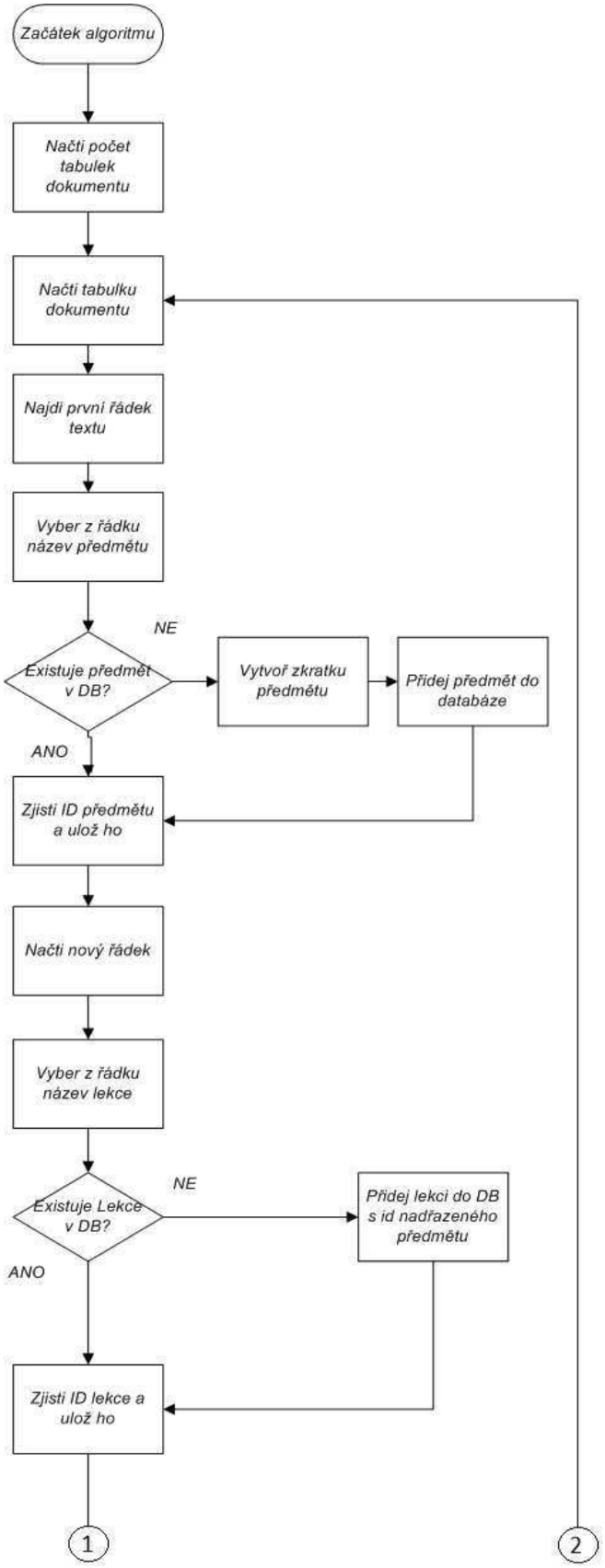
Informační panel slouží pro výpis informací aplikace. Tyto informace mohou být druhu informativního, varovného nebo chybového.

Informační hlášení slouží k informování uživatele. Jaké části byly průběžně nahrány do databáze a zda proběhl import úspěšně.

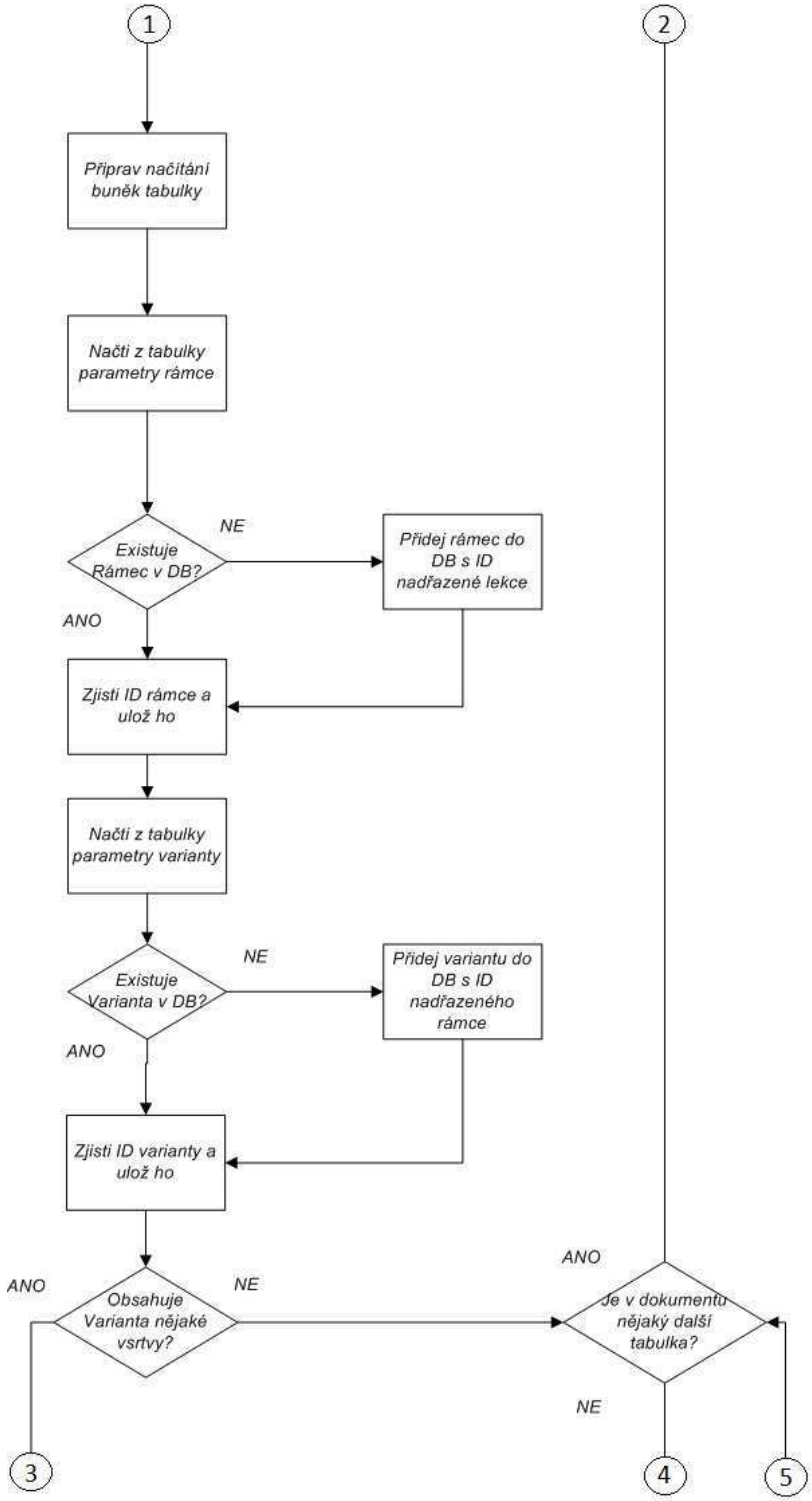
Varovné hlášení upozorní uživatele na některé nedostatky v dokumentu, které neznamenají znemožnění jeho nahrání, ale uživatel by je měl v dokumentu upravit, aby byly informace přesnější.

Chybové hlášení upozorní uživatele na takový nedostatek v dokumentu, že program nemůže dále pokračovat v jeho načítání. Chce-li uživatel tento dokument úspěšné nahrát do systému, bude muset tyto nedostatky opravit.

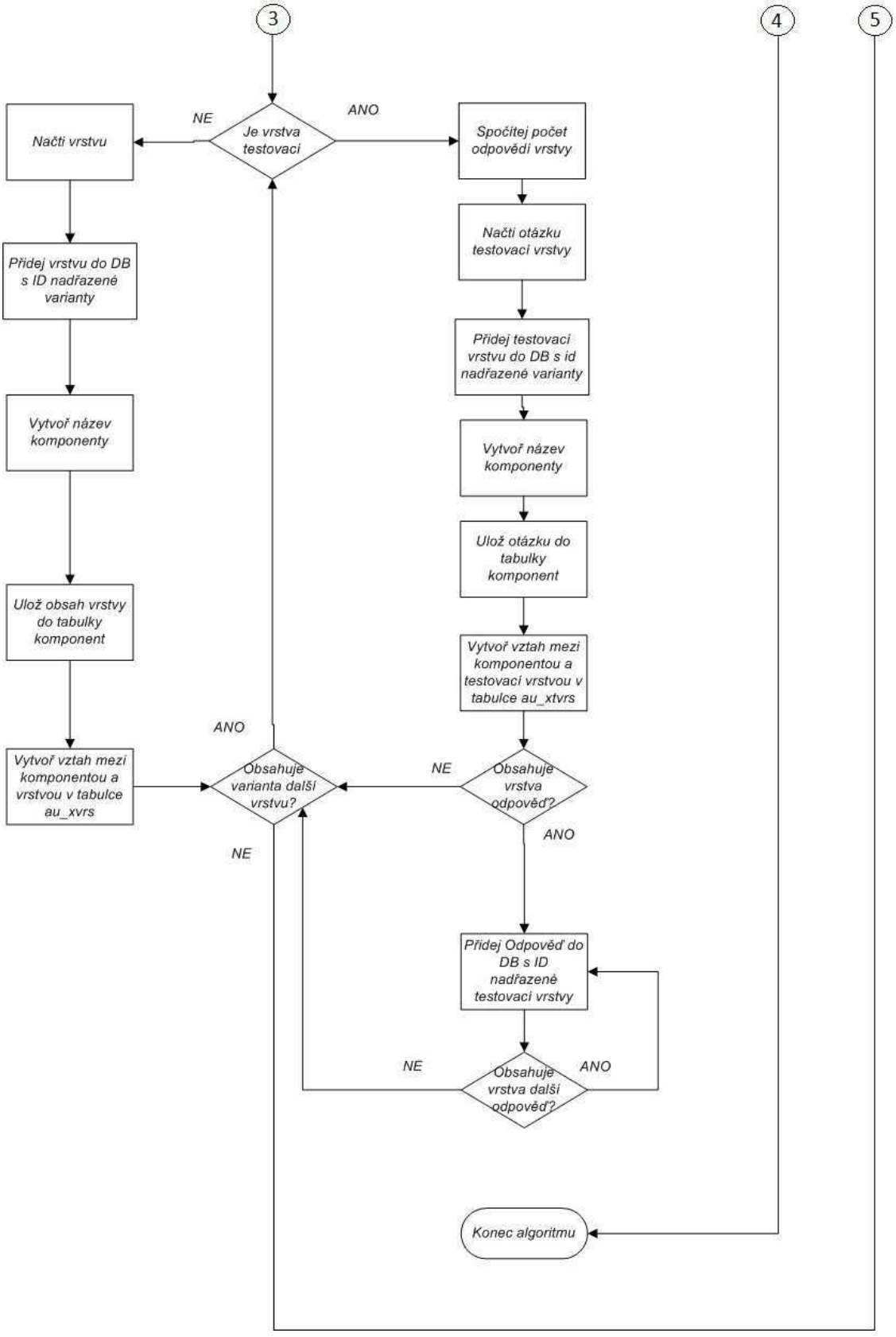
7 Vývojový diagram



Obr. 4 Vývojový diagram 1. část



Obr. 5 Vývojový diagram 2. část



Obr. 6 Vývojový diagram 3. část

7.1 Popis vývojového diagramu

1. Načti počet tabulek dokumentu - zjistí si kolik dokument obsahuje tabulek a načte si jejich počet.

2. Načti tabulku dokumentu – načte první tabulku dokumentu podle čísla.

3. Najdi první řádek textu – načte první řádek, který obsahuje nějaký text

4. Vyber z řádku název předmětu – vybere z načteného řádku název předmětu

5. **<u>Rozhodnutí</u>** - **Existuje předmět v DB?**

- NE

6. Vytvoř zkratku předmětu – vytvoří z názvu předmětu jeho zkratku

7. Přidej předmět do databáze – vloží předmět do databáze
- pokračuje se bodem 8.

- ANO

8. Zjisti ID předmětu a ulož ho – načte z databáze ID předmětu a uloží si ho pro pozdější použití

9. Načti nový řádek – načte další řádek obsahující text

10. Vyber z řádku název lekce – načte z řádku název lekce

11. <u>Rozhodnutí</u> – Existuje lekce v DB?

- NE

12. Přidej lekci do DB s id nadřazeného předmětu
- pokračuje se bodem 13

- ANO

13. Zjisti ID lekce a ulož ho – načte z databáze ID lekce a uloží si ho pro pozdější použití

14. Připrav načítání buněk tabulky – získám instanci pro načítání buněk tabulky

15. Načti z tabulky parametry rámce – načte z tabulky obsahy buněk, které se týkají rámce

16. <u>Rozhodnutí</u> – Existuje rámec v DB?
- NE

17. Přidej rámec do DB s ID nadřazené lekce
- pokračuje se bodem 18

- ANO
18. Zjisti ID rámce a ulož ho - načte z databáze ID rámce a uloží si ho pro pozdější použití

19. Načti z tabulky parametry varianty - načte z tabulky obsahy buněk, které se týkají varianty

20. <u>Rozhodnutí</u> – Existuje varianta v DB?

- NE

21. Přidej variantu do DB s ID nadřazeného rámce

pokračuje se bodem 22

- ANO

22. Zjisti ID varianty a ulož ho - načte z databáze ID varianty a uloží si ho pro pozdější použití

23. <u>Rozhodnutí</u> – Obsahuje varianta nějaké další vrstvy?

ANO – varianta obsahuje další vrstvy, přesouvá se tedy na další rozhodnutí (24)
NE – pokud varianta již neobsahuje další vrstvy, varianta je tedy už prázdná, na dalším rozhodnutí otestujeme, jestli obsahuje dokument nějaké další tabulky. Pokud ano algoritmu se vrací k bodu (2.) a začne načítat další tabulku. Pokud dokument už žádné tabulky neobsahuje, algoritmus končí.

24. <u>Rozhodnutí</u> – Je vrstva testovací?
NE – načte se klasická vrstva, vytvoří se pro ni komponenta a ve vazební tabulce au_xvrs se propojí,
ANO- vrstva je testovací, než uložíme otázku vrstvy, musíme si spočítat počet odpovědí, protože to parametr otázky vyžaduje. Pak se testovací vrstva uloží a následně se uloží i komponenta, která obsahuje text otázky. Nakonec se tato komponenta s testovací vrstvou propojí v tabulce au_xtvrs. následuje další rozhodnutí:

25. <u>Rozhodnutí</u> – Obsahuje vrstva odpověď?
NE – algoritmus směřuje na rozhodnutí 23.
ANO - odpověď se načte a přidá do databáze, poté se algoritmus dostane do smyčky, kde ukládá postupně všechny odpovědí u testovací vrstvy. Po uložení všech odpovědí, pokračuje algoritmus bodem 23

Tyto 3 Rozhodnutí se opakují jsou-li k dispozici vrstvy. Pokud varianta už další vrstvy neobsahuje, algoritmus se pokusí načíst další tabulku dokumentu a algoritmus by opět pracoval od začátku. Pokud však žádná další tabulka v dokumentu není, algoritmus končí.

8 Programátorská příručka

8.1 Nástroje pro vývoj

Nástroje pro vývoj této aplikace jsou na přiloženém CD ve složce s názvem „nástroje pro vývoj“. Postup přípravy nástrojů pro vývoj:

1. Instalace vývojového prostředí NetBeans.
2. Instalace JDK(Java Development Kit).
2. Instalace Open Office kancelářského balíku.
3. Instalace Open Office SDK.
4. Soubor ze složky „knihovny“ importovat do projektu v NetBeans, jedná se o knihovny OpenOffice API.
5. Založit nový projekt v NetBeans, jako typ projektu vybrat „Apache OpenOffice“.
6. Aby bylo možné pracovat s databází, je potřeba do projektu, jako knihovnu, naimportovat jar soubor z archivu „mysql-connector-java-5.1.24.zip“, tedy soubor „mysql-connector-java-5.1.24-bin.jar“.

Nyní je projekt připraven pro vývoj s OpenOffice API.

8.2 Použité funkce Open Office API

Open Office API obsahuje funkce, ze kterých jsem využil především ty pro načítání tabulek a práci s jejich buňkami.

XComponent xWriterComponent

- instance objektu XComponent vytvoří objekt do kterého se uloží importovaný dokument

XTextDocument xTextDocument = (XTextDocument)UnoRuntime.queryInterface(

XTextDocument.class, xWriterComponent)

- instance objektu XTextDocument nám umožní vytvořit z již načteného dokumentu, dokument, který je již připraven k práci s aplikací. Jako vstupní parametr je vkládána instance objektu XComponent: xWriterComponent.

XText xText = xTextDocument.getText()
- do instance objektu XText takto načítáme obsah celého dokumentu. S tímto textem už teď můžeme pracovat**.**

XTextCursor xCursor = xText.createTextCursor()
- instance objektu XTextCursor nám umožní pohybovat se v textu pomocí souřadnice, která je na prvním znaku textu 0 a na dalších znacích se inkrementuje

XTextTablesSupplier xTablesSupplier = (XTextTablesSupplier)UnoRuntime.queryInterface(
XTextTablesSupplier.class, xTextDocument)
- vytváříme instanci rozhraní XTextTablesSupplier, jehož metody nám umožní manipulovat s tabulkami. Jako vstupní parametr je tu instance vloženého dokumentu xTextDocument

XNameAccess xNamedTables = xTablesSupplier.getTextTables()
- do instance objektu XNameAccess si načteme kolekci všech tabulek v dokumentu

XIndexAccess xIndexedTables = (XIndexAccess)
UnoRuntime.queryInterface(XIndexAccess.class, xNamedTables)
- touto instancí získáme přístup k metodám, které nám umožní manipulovat s tabulkami tak, že každá bude pod číslem, začínajícím od nuly

Object table = xIndexedTables.getByIndex(i);
- do objektu table ukládáme tabulku z dokumentu danou indexem i

XCellRange xCellRange = (XCellRange)UnoRuntime.queryInterface(
XCellRange.class, table)
- pomocí instance xCellRange získáme kontrolu nad buňkama tabulky, která je uložena v parametru „table“.

8.3 Popis třídy ParseTable2

Aplikace obsahuje třídy: IdClass, Database a ParseTable2. Kompletní popisy tříd a metod jsou obsaženy v příloze.

Metody:

XComponent newDocComponent()
- z importovaného dokumentu vytvoří komponentu pro jeho nahrání
- vrací komponentu s připraveným dokumentem

void NactiPredmet (XTextCursor xCursor, Database db)
- načte z tabulky dokumentu předmět
- parametr xCursor – instance objektu XTextCursor, která umožňuje pohyb v textu pomocí kurzoru
- parametr db – instance třídy Database, která pracuje s databází

void NactiLekci(String retezec, int pKonec, Database db)
- načte z tabulky dokumentu lekci
- parametr retezec – obsahuje načtený text před tabulkou
- parametr pKonec – obsahuje index, na kterém skončilo načítání názvu předmětu
- parametr db – instance třídy Database, která pracuje s databází

void NactiRamec(Database db, XCellRange xCellRange, IdClass idClass)
- načte z tabulky dokumentu rámec
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky

void NactiVariantu(Database db, XCellRange xCellRange)
- načte z tabulky dokumentu variantu
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky

int NactiVrstvy(Database db, XCellRange xCellRange, int radek, IdClass idClass)
- načte z tabulky dokumentu klasické netestovací vrstvy
- vrátí číslo řádku, na kterém skončila
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky
- parametr radek – obsahuje aktuální číslo řádku tabulky
- parametr idClass – instance třídy IdClass, která obsahuje pomocné proměnné

int NactiOtazky(Database db, IdClass idClass, XCellRange xCellRange, int radek)
- načte z tabulky dokumentu otázku v testovací vrstvě
- vrátí číslo řádku, na kterém skončila
- parametr db – instance třídy Database, která pracuje s databází
- parametr idClass – instance třídy IdClass, která obsahuje pomocné proměnné
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky
- parametr radek – obsahuje aktuální číslo řádku tabulky

int SpocitejOdpovedi(int radek, XCellRange xCellRange)
- spočítá počet odpovědí u otázky
- vrátí číslo řádku, na kterém skončila
- parametr radek – obsahuje aktuální číslo řádku tabulky

- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky

void NactiOdpovedi(Database db, XCellRange xCellRange, int radek, int pocet_odpovedi, IdClass idClass)
- načte z otázky všechny odpovědi
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky
- parametr pocet_odpovedi – obsahuje počet odpovědí otázky
- parametr radek – obsahuje aktuální číslo řádku tabulky
- parametr idClass – instance třídy IdClass, která obsahuje pomocné proměnné

int NactiTestovaciVrstvu(int radek, Database db, XCellRange xCellRange, IdClass idClass)
- načte vrstvu, která obsahuje odpovědi s reakcemi „OReak“
- vrátí číslo řádku, na kterém skončila
- parametr radek – obsahuje aktuální číslo řádku tabulky
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky
- parametr idClass – instance třídy IdClass, která obsahuje pomocné proměnné

int ZjistiVrstvu(Database db, XCellRange xCellRange, int radek, IdClass idClass)
- zjišťuje, o jaký druh vrstvy se jedná
- vrací číslo 0 – vrstva je klasická, 1- vrstva je testovací, 2- další vrstva neexistuje
- parametr db – instance třídy Database, která pracuje s databází
- parametr xCellRange – instance, která umožňuje načítání obsahu buněk dané tabulky
- parametr radek – obsahuje aktuální číslo řádku tabulky
- parametr idClass – instance třídy IdClass, která obsahuje pomocné proměnné

String VytvorNazevTvrstvy(String nazevRamce, String typ_o, int cisloTVrstvy)
- vytvoří název testovací vrstvy
- vrátí vytvořený název testovací vrstvy
- parametr nazevRamce – obsahuje Název nadřazeného rámce
- parametr typ_o – obsahuje typ testovací vrstvy
- parametr cisloTVrstvy – obsahuje pořadové číslo testovací vrstvy v rámci

String VytvorNazevKomponenty(String nazevRamce, String typ_vr, int cislo)
- vytvoří název komponenty
- vrátí vytvořený název komponenty
- parametr nazevRamce – obsahuje Název nadřazeného rámce
- parametr typ_vr – obsahuje typ vrstvy, ze které je komponenta
- parametr cislo – obsahuje pořadové číslo komponenty v rámci

int NactiZBunkyCislo(String obsahBunky)
- načte číslo z buňky, které se nachází za znakem „=“
- vrátí načtené číslo
-parametr obshaBunky – obsahuje řetězec, ze kterého chceme načíst číslo

void UlozKomponentu(Database db, int cisloKomponenty, String obsah_k, String nazevRamce, String typ_vr, int typVrstvy)
- uloží komponentu, jejíž parametry dostane na vstupu
- parametr db – instance třídy Database, která pracuje s databází
- parametr cisloKomponenty – obsahuje pořadové číslo komponenty v rámci

- parametr obsah_k – vlastní text komponenty
- parametr nazevRamce – obsahuje Název nadřazeného rámce
- parametr typ_vr – typ vrstvy ve které je ukládaná komponenta
- parametr typVrstvy – obsahuje číslo, které označuje typ vrstvy: 0 – klasická vrstvy, 1 – testovací vrstva

void Vypis(String text)
- vypíše text a uloží jej do textového souboru na nový řádek
- parametr text – text, který se uloží do souboru pro výpis a vypíše se

void main(String[] args)
- startovní metoda, volá postupně všechny ostatní metody

9 Závěr

Předkládaná bakalářská práce si kladla za cíl zjednodušit a zpříjemnit uživatelům vkládání dokumentů typu Microsoft Word do systému Barborka. Vytyčený cíl této práce se podařilo splnit. Vyskytly se však některé věci, které se budou muset i nadále vyplňovat přes formuláře systému Barborka. Jde zejména o obrázky, videa, zvukové záznamy, které významně přispívají k výuce a k věrohodnosti adaptivní učebnice. Mnou zvolené programovací rozhraní neumožňovalo takovéto komponenty načíst a tudíž ani vložit je do databáze systému. Nalezením jiného programovacího rozhraní, které by umožňovalo takovýto požadavek splnit, se mně bohužel nepodařilo. Zde bych tedy i hledal do budoucnosti možnost, jak by šla má bakalářská práce rozšířit. I tak se mi povedlo uživatelům velmi zjednodušit vložení a začlenění takovéto učebnice do databáze. Uživatel už nebude muset vkládat do systému přes webový formulář jednotlivé části učebnice, ale postačí, když si dokument správně připraví a nahraje ho tak jako celek.

Seznam obrázků

Obr. 1 Variantní rámce dle smyslové formy 5
Obr. 2 Zjednodušený ER – Diagram databáze 8
Obr. 3 Uživatelské rozhraní 14
Obr. 4 Vývojový diagram 1. část 17
Obr. 5 Vývojový diagram 2. část 18
Obr. 6 Vývojový diagram 3. část 19

Seznam tabulek

Tab. 1 Formulář pro tvorbu e-learningových opor 15

Seznam použité literatury

[1] *HOLUB, Libor. Automatizované řízení adaptivní výuky v e-learningu podle stylů učení studenta. Ostrava, 2011. 94 s. Dizertační práce. Vysoká škola báňská – Technická univerzita Ostrava.*

[2] *Distanční vzdělávání v České republice: současnost a budoucnost : "uplatnění distanční formy pro vzdělávání dospělých" : III. národní konference, sborník příspěvků, Brno, 30.6.-2.7.2004. Vyd. 1. Praha: Centrum pro studium vysokého školství, 2004, 342 s. ISBN 80-863-0202-4. Dostupné z: http://www.csvs.cz/konference/NCDiV2004_sbornik*

[3] *HOLUB, Libor, Jana ŠARMANOVÁ a Radoslav FASUGA. VŠB – TECHNICKÁ UNIVERZITA OSTRAVA. Student - uživatelská příručka Barborka. Dostupné z: http://k101.unob.cz/~hajkova/1html/10psp1/4soubory/1student.pdf*

[4] *PARAMYTHIS, Alexandros a Susanne LOIDL-REISINGER. JOHANNES KEPLER UNIVERSITY, Linz, Austria. Adaptive Learning Environments and e-Learning Standards. 2nd European Conference on e-Learning (ECEL 2003), 2004. Dostupné z: http://web.archive.org/web/20100331014042/http://www.ejel.org/volume-2/vol2-issue1/issue1-art11-paramythis.pdf*

[5] *KOSTOLÁNYOVÁ, Kateřina. Teorie adaptivního e-learningu. Vyd. 1. Ostrava: Ostravská univerzita v Ostravě, 2012. 118 s. ISBN 978-80-7464-014-8. Ostravská univerzita v Ostravě.*

[6] *ŠARMANOVÁ, Jana. VŠB – TECHNICKÁ UNIVERZITA OSTRAVA. Metodika tvorby adaptivních e-learningových učebnic. Ostrava, 2011.*

Internetové zdroje

[i1] *Barborka Learnig Management System [online]. 2010 [cit. 2010-05-06]. Dostupný z WWW: <http://barborka.vsb.cz/>.*

[i2] *B. F. Skinner: Teaching machine. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-02]. Dostupné z: http://en.wikipedia.org/wiki/Bf_skinner#Teaching_machine*

[i3] *Adaptive_learning. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2013-05-02]. Dostupné z:* http://en.wikipedia.org/wiki/Adaptive_learning

Přílohy

CD obsahuje složky:

1. server – obsahuje soubory pro nahrání na server
2. software – ostatní software
3. nástroje pro vývoj – software a nástroje pro vývoj v OpenOffice API
4. aplikace – vlastní aplikace v NetBeans
5. test aplikace – nástroje a vše potřebné pro okamžité otestování aplikace
6. Příloha_BP_CHO0031.pdf – dokument, příloha k bakalářské práci