VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

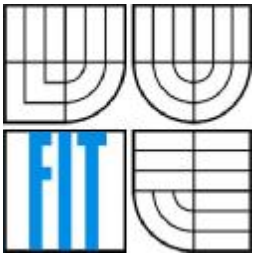
SYSTÉM PRO PODPORU E-LEARNINGU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE MICHAL DRAHOŠ
AUTHOR

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PODPORU E-LEARNINGU

SYSTEM FOR E-LEARNING SUPPORT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — MICHAL DRAHOŠ
AUTHOR

VEDOUCÍ PRÁCE — RNDr. JITKA KRESLÍKOVÁ, CSc.
SUPERVISOR

BRNO 2007

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Drahoš Michal**
Obor: Výpočetní technika a informatika
Téma: **Systém pro podporu e-learningu**
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se s jazykem pro specifikaci omezení objektů systému (OCL, UML 2.0) a s prostředím CASE Rational Rose, který UML podporuje.
2. Prostudujte problematiku e-learningu a tvorby elektronických dokumentů, vhodných pro samostatné studium. Proveďte revizi dokumentů pro výuku OCL vytvořených v rámci ročníkového projektu tak, aby kompletně pokrývaly celou problematiku.
3. Navažte na ročníkový projekt a specifikujte požadavky na programový systém pro podporu e-learningu. Systém musí také umožňovat prověřování znalostí a sledování pokroku v pochopení prezentované látky. Systém navrhněte.
4. Zvolte vhodné implementační prostředí, volbu zdůvodněte. Vytvořte funkční prototyp navrženého systému.
5. Funkčnost systému ověřte na studijních materiálech pro výuku jazyka specifikace omezení objektů systému.
6. Diskutujte možnosti dalšího rozvoje aplikace a zhodnoťte použitelnost ve výuce.

Literatura:

- Horton, W.: Design Web-Based Training: How to Teach Anyone Anything Anywhere Anytime, Wiley; 1 edition, ISBN: 047135614X
- Horton, W.: E-learning Tools and Technologies. John Wiley & Sons; 1st edition (January 10, 2003), ISBN: 0471444588.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 - 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kreslíková Jitka, RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Michal Drahoš**
Id studenta: 22041
Bytem: Mlýnská 260, 741 01 Nový Jičín
Narozen: 18. 04. 1982, Bílovec
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Systém pro podporu e-learningu
Vedoucí/školitel VŠKP: Kreslíková Jitka, RNDr., CSc.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nahývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tento projekt se zabývá tvorbou systému pro podporu e-learningu. Popisuje problematiku e-learningu (elektronického vzdělávání), vysvětluje, co tento pojem znamená, a popisuje jeho výhody i nevýhody. Úkolem bylo vytvořit aplikaci vhodnou k elektronickému vzdělávání, která by objasňovala specifikaci omezení objektů systému. To znamená vytvořit aplikaci, ve které by byla vysvětlena problematika OCL a to v takovém formátu, jenž by odpovídal struktuře a formě aplikace užívající se k elektronickému vzdělávání. Seznámil jsem se s jazyky OCL i UML 2.0. Prostudoval prostředí CASE Rational Rose a také se obeznámil s problematikou elektronického vzdělávání. Těchto poznatků jsem využil při vytváření e-learningové aplikace.

Klíčová slova

Elektronické vzdělávání, elektronické dokumenty, OCL, UML 2.0, CASE Rational Rose, diagram, web-based training, www stránky, HTML, CSS, skriptovací jazyk PHP.

Abstract

This project is dealing with creating of system for e-learning support. It describes problematics of e-learning, explains meaning of this term and describes its advantages and disadvantages. The objectives were to create an application suitable for electronic learning, which could illustrate specifications of restrictions of the system objects. This means to create an application, where is explained the problem of OCL in that kind of format, which reacts the structure and the form of application used for e-learning. I meet the program languages of OCL and UML 2.0. I read over the CASE Rational Rose environment and meet the e-learning problem, too. These are the knowledges, which I used while I was creating my e-learning application.

Keywords

E-learning, electronic documents, OCL, UML 2.0, CASE Rational Rose, diagrams, web-based training, www sites, HTML, CSS, PHP language.

Citace

Michal Drahoš: Systém pro podporu e-learningu, diplomová práce, Brno, FIT VUT v Brně, 2007

Systém pro podporu e-learningu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením RNDr. Kreslíkové Jitky, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Drahoš
1.5.2007

Poděkování

Děkuji vedoucí semestrální práce RNDr. Kreslíkové Jitce, CSc. za poskytnuté rady a doporučenou literaturu.

Obsah

Obsah ... 1
1 Úvod ... 3
2 Jazyk pro specifikaci omezení objektů systému ... 4
2.1 OCL, UML 2.0 ... 4
2.1.1 UML 2.0 ... 4
2.1.2 OCL ... 16
2.2 CASE Rational Rose ... 21
2.2.1 Prostředí CASE Rational Rose ... 21
3 E-learning a tvorba elektronických dokumentů ... 25
3.1 E-learning ... 25
3.1.1 Co je to vlasně e-learning ... 25
3.1.2 Definice e-learningu ... 26
3.1.3 Historie e-learningu ... 26
3.1.4 E-learning: co, proč, jak ... 28
3.1.5 Distribuce ... 29
3.2 Obecná struktura elektronických dokumentů ... 30
3.3 Revize dokumentů pro výuku OCL ... 32
4 Požadavky na programový systém pro podporu e-learningu ... 33
4.1 Funkční požadavky ... 33
4.2 Jazykové požadavky ... 33
4.2.1 Program Socrates / Erasmus ... 33
4.3 Požadavky na prověřování znalostí ... 33
5 Prostředky pro vývoj www aplikací ... 34
5.1 WWW, URL a http ... 34
5.1.1 Co se skrývá pod zkratkou WWW. ... 34
5.1.2 URL a http ... 35
5.2 HTML ... 37
5.2.1 Vývoj a historie ... 37
5.2.2 Popis jazyka ... 38
5.3 CSS ... 39
5.3.1 Výhody a nevýhody používání CSS ... 39
5.4 PHP ... 41
5.4.1 Co to je PHP ... 41
5.4.2 Jak PHP pracuje a jaké máme verze ... 41

5.5 Databáze 42
5.5.1 MySQL 43
5.6 Editory a prohlížeče 43
5.6.1 Editory 44
5.6.2 Prohlížeče 44
5.7 Výběr a zhodnocení prostředků pro vývoj www aplikace 44
6 Programový systém pro podporu e-learningu 46
6.1 Specifikace požadavků a analýza 46
6.1.1 Obecné požadavky 46
6.1.2 Funkční požadavky 46
6.2 Návrh 47
6.2.1 Ročníkový návrh aplikace 48
6.3 Implementace 52
6.3.1 Puštění aplikace a výběr jazykové verze 52
6.3.2 Modul přihlašování 53
6.3.3 Grafické rozhraní 57
6.3.4 Menu 63
6.3.5 Zpracování studijních materiálů 64
6.3.6 Jazykové verze 64
6.3.7 Testová část 65
6.3.8 Testování a funkčnost 67
6.3.9 Aktualizace obsahu 67
6.3.10 Instalace 67
7 Závěr 69
Literatura 70
Seznam příloh 72

1 Úvod

Tato práce se zabývá problematikou elektronického vzdělávání a jazykem pro specifikaci omezení objektů systému. Úkolem, dle zadání, bylo prostudovat tyto problematiky a navrhnout a vytvořit systém, který by obsahoval specifikaci OCL UML 2.0. Systém by měl mít vlastnosti e-learningové aplikace, takže by měl být přehledný a měl by umožňovat prověřování znalostí a sledování pokroku v pochopení prezentované látky.

Součástí diplomové práce bylo navázat na ročníkový projekt a provést revizi dokumentů pro výuku OCL vytvořených v rámci ročníkového projektu tak, aby kompletně pokrývaly celou problematiku. Tato problematika je velmi rozsáhlá a zpracování materiálu obsahující specifikaci OCL je velmi náročné, což jsem zjistil již při tvorbě ročníkového projektu.

V teoretické části této práce se věnuji nejprve jazyku pro specifikaci omezení objektů systému. To je popsáno v kapitole 2. Kapitola 3 pojednává o problematice e-learningu a tvorbě elektronický dokumentů. Součástí této kapitoly je část, ve které se zabývám návazností práce na semestrální projekt.

Kapitoly 4 a 5 jsou věnovány specifikaci požadavků na daný systém a popisem vývojových prostředků nutných pro vývoj systému.

Implementaci systému je určena kapitola kapitola 6. Tato kapitola je věnována konkrétnímu návrhu systému, podle kterého jsem vytvořil funkční prototyp umožňující studium specifikace omezení objektů systému. Popisuji zde jednotlivé části projektu, způsoby jejich implementace. Také se zde zabývám testováním funkčnosti vytvořeného prototypu.

Závěrem se zamýšlím nad použitelností obsahu diplomové práce ve výuce a možností dalšího rozvoje aplikace.

2 Jazyk pro specifikaci omezení objektů systému

V této kapitole se seznámíme s jazykem pro specifikaci omezení objektů systému (OCL, UML 2.0) a s prostředím CASE Rational Rose, který UML podporuje. V další části si povíme něco o tom, co je to e-learning, co tento pojem představuje, a popíšeme si, jak by měla e-learningová aplikace vypadat.

2.1 OCL, UML 2.0

Cílem této podkapitoly je obeznámit čtenáře s jazykem OCL (UML 2.0), což je jazyk pro specifikaci omezení objektů systému.

2.1.1 UML 2.0

V této kapitole se budu věnovat UML 2.0. Seznámíme se tímto jazykem, povíme si něco o historii a vývoji. Také si ukážeme, kde a jakým způsobem se používá.

2.1.1.1 Co je to UML

Jazyk UML (Unified Modeling Languague, unifikovaný modelovací jazyk) je univerzální jazyk pro vizuální modelování systému. Je to standardní průmyslový jazyk pro specifikaci, vizualizaci, vytváření a dokumentaci artefaktů softwarových systémů. Přestože je nejčastěji spojován s modelováním objektově orientovaných softwarových systémů, má mnohem širší využití, což vyplývá z jeho zabudovaných rozšiřovacích mechanismů.

Byl navržen, aby spojil nejlepší existující postupy softwarového inženýrství a modelovacích technik. Explicitně je navržen tak, aby jej mohly implementovat všechny nástroje CASE (computer-aided software engineering). Diagramy vytvořené v jazyku UML jsou srozumitelné pro lidi, ale navíc je mohou snadno interpretovat i CASE programy.

Definice

UML = unified modeling language (tj. *unifikovaný modelovací jazyk*).

Unifikovaný : unifikuje Booch, OMT a Objectory modelovací jazyky

Modelovací : UML je jazyk pro specifikaci, visualizaci, konstrukci a dokumentaci artefaktů SW systémů.

UML je však využitelný i pro business modelování i pro modelování ne-SW systémů. V UML lze modelovat jakýkoliv typ aplikace běžící na jakémkoliv typu a kombinaci HW, OS, programovacím jazyku a sítě. Lze modelovat distribuované aplikace. Pro UML je nejpřirozenější

modelování pro OO jazyky a prostředí (jako jsou např. C++, Java, C#), ale lze modelovat i pro neobjektové jazyky (např. Fortran, VB, COBOL). UML profiles nám pomůže v modelování transakčních, real-time a fault-tolerant systémů.

Language (jazyk) :UML není programovací jazyk, ale je to jazyk, neboť má :

- syntaxi (grafickou), tj. pravidla, dle kterých jsou elementy jazyka sestavovány do výrazů
- sémantiku, tj. pravidla, dle kterých je syntaktickým výrazům přiřazen význam

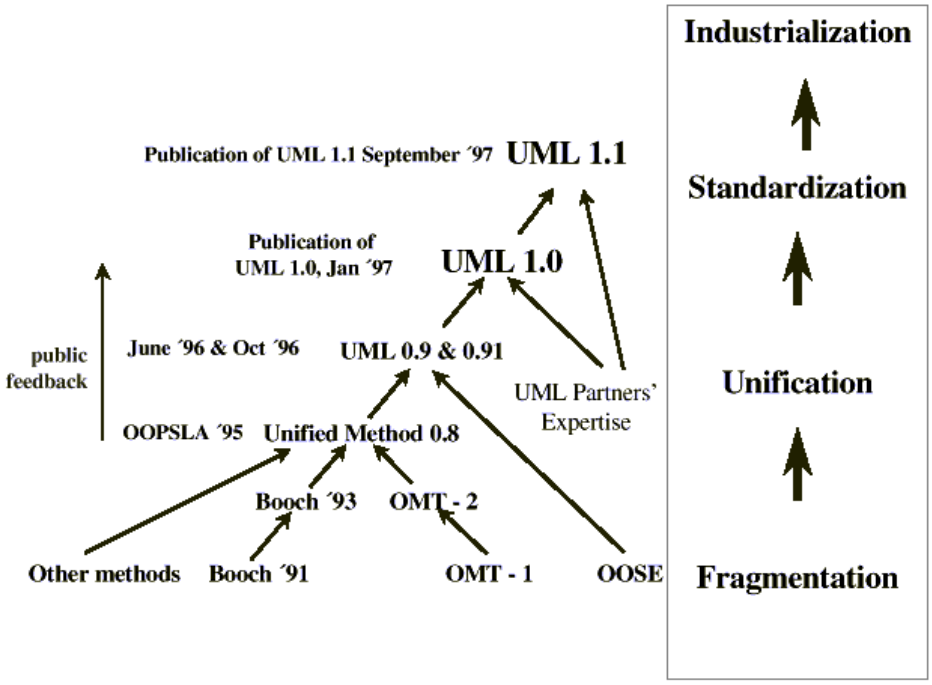
UML samo v sobě obsahuje mechanismy pro jeho rozšíření (dle potřeb konkrétního projektu nebo konkrétního vývojáře) [16].

Další vlastnosti UML :

- otevřený standard
- podporuje celý vývojový cyklus
- podporuje různě aplikační oblasti
- založeno na zkušenostech a potřebách komunity uživatelů
- podporováno celou řadou nástrojů
- přispěvatelé : Booch (Booch metoda), Rumbaugh (OMT), Jacobson (OOSE), Shlaer-Mellor (object lifecyles), Odell (clasification), Wirfs-Brock (Responsibilities), Embley (Singleton classes and high level view), HP Fusion (operation descriptions and message numbering), Gamma a kolektiv (frameworks and patterns), Harel (statecharts), Meyer (before and after conditions)

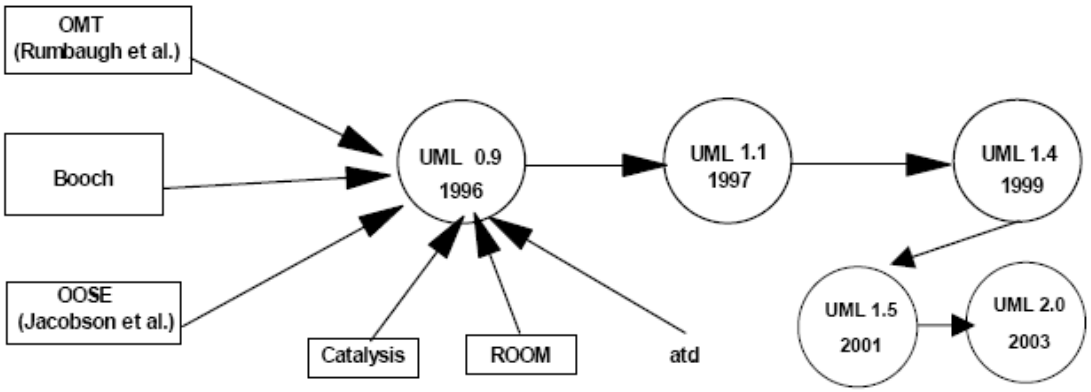
2.1.1.2 Historie a standardy

Jazyk UML vytvořili metodici Grady Booch, Ivar Jacobson a Jim Rumbaugh ve společnosti Rational Software. Jejich cílem bylo sjednotit různé stávající systémy do jediného nejlepšího modelovacího jazyka. První verze UML 0.8 byla vytvořena v říjnu 1995. Koncem roku 1995 se k firmě Rational připojil také Ivar Jacobson a začalo připojování jeho OOSE metodologie. Výsledkem práce těchto velikánů v oblasti OOAD byla verse UML 0.9 koncem roku 1996. Během roku 1996 začalo také strategické spojení, jednak s Object Management Group (OMG), ale také s mnohými komerčními partnery. V roce 1996 navrhlo sdružení OMG (Object Management Group) specifikaci RFP (Request for Proposal) pro objektově orientovaný jazyk pro vizuální modelování, v němž jako standard navrhlo jazyk UML. Jednalo se o verzi UML 1.0. V lednu roku 1997 byl jazyk UML přijat jako standard a koncem roku 1997 byla přijata rozšířená verze s označením 1.1. Vývoj jazyka UML až po verzi 1.1 je zobrazen na obrázku 1. [13]. Jazyk UML se stal standardem v softwarovém průmyslu a dále se vyvíjí. V roce 1999 byl zaveden standard OML 1.3 a v roce 2001 byla uvedena verze UML 1.4. Specifikace UML 2.0 je převzata ze semestrálního projektu.



Obr. 1. Vývoj UML s časovou osou do verze UML 1.1

Jazyk UML se stal standardem v softwarovém průmyslu a dále se vyvíjí. V roce 1999 byl zaveden standard OML 1.3 a v roce 2001 byla uvedena verze UML 1.4. Následovala verze UML 1.5, která byla uvedena v roce 2001. Tato diplomová práce se věnuje specifikaci UML 2.0, která byla uvedena v roce 2003. Viz obr.2. [15].



Obr. 2. Vývoj UML do verze UML 2.0

Účastníci UML

V současné době se vývoje UML oficiálně účastní mnoho špičkových světových firem. Jedná se o následující firmy Hewlett-Packard, IBM, I-Logix, ICON Computing, IntelliCorp, MCI Systemhouse, Microsoft, ObjectTime, Oracle, Platinum Technology, Ptech, Rational Software, Reich Technologies a Taskon, Softeam, Sterling Software, Unisys.

Vývoj UML též podporuje celá řada dalších veličin v oblasti OOAD, například Peter Coad, Mary Loomis, Ed Yourdon a celá řada dalších [13].

2.1.1.3 SOUČASNOST A BUDOUCNOST UML

UML je otevřený, všem přístupný standard. K jeho používání se hlásí jak mnoho teoretiků, tak i běžných uživatelů, organizací a mnoho tvůrců CASE nástrojů. UML staví na obdobné sémantice a zápisu jako Booch, OMT a další vůdčí metodiky. UML je „unifikovaný“ jazyk zejména v tom, že

- efektivně odstraňuje rozdíly (mnohdy bezdůvodné) mezi modelovacími jazyky předcházejících metod,
- unifikuje přístupy k rozličným typům systémů (ekonomické vers. softwareové), vývojovým fázím (analýza požadavků, návrh, implementace) a interním konceptům.

Standardizace, industrializace a evoluce UML

Mnoho organizací již přijalo UML za svůj standard. Přestože UML definuje jazyk precizně, není to překážkou dalších vylepšení modelovacích technik, aniž by tato rozšíření musela znamenat redefinici jádra UML.

UML dovolilo integrovat řadů různorodých myšlenek, čímž přispělo i k rozšíření OO. Mezi jiným je důležitý i vývoj založený na komponentech. Přesto, že se široce uplatňuje, neznamená náhradu OO a znamená jen drobné rozdíly v sémantice komponent a tříd.

UML 1.4.2 je přejato i v soustavě norem ISO/IEC č. 19501 a doporučeních ITU, ITU-T Recommendations Z.100 (SDL) a Z.109 (SDL UML profile) [15].

2.1.1.4 K čemu slouží UML jazyk?

Jazyk UML byl navržen hlavně pro následující účely:

- modelovaní aplikačních procesů s případy použití
- modelování tříd a objektů
- modelování komponent
- modelování rozdělení a nasazení

2.1.1.5 Základní součásti UML

Standard ve verzi 2.0 se skládá ze čtyř částí:

- **UML 2.0 SuperStructure** – popis UML z hlediska uživatele (analytik/programátor). Tato část popisuje jednotlivé diagramy.
- **UML 2.0 Infrastructure** – metamodel stojící v pozadí za UML, specifikovaný pomocí Meta-Object Facility (MOF).
- **UML 2.0 Object Constraint Language (OCL)** – jazyk pro specifikaci vstupních a výstupních podmínek, invariantů v jednotlivých diagramech.

- **UML 2.0 Diagram Interchange** – popis XML struktur pro výměnu konkrétních modelů mezi jednotlivými modelovacími nástroji.

Vedle vlastního standardu existují UML profily – přizpůsobení UML pro jednotlivé oblasti:

- UML Profile for CORBA®
- UML Profile for CORBA Component Model (CCM)
- UML Profile for Enterprise Application Integration (EAI)
- UML Profile for Enterprise Distributed Object Computing (EDOC)
- UML Profile for QoS and Fault Tolerance
- UML Profile for Schedulability, Performance, and Time
- UML Testing Profile

Z UML též začínají vznikat různé dialekty – modelovací jazyky pro určité oblasti, které přebírají část UML, kterou modifikují a doplní o prvky specifické pro konkrétní oblast. Příkladem může být jazyk Systems Modeling Language (SysML), určený pro specifikaci, analýzu, návrh, verifikaci a validaci různých systémů.

Též většina metodik pro analýzu a návrh systémů upřednostňuje části z UML a doplňuje je o další prvky [22].

2.1.1.6 Způsoby použití UML

Existuje několik způsobů použití, které to jsou popisuje tato kapitola.

Kreslení konceptu

Při tomto použití je UML podpůrným nástrojem pro komunikaci mezi vývojáři a pro zaznamenání myšlenek a návrhů. Do diagramů se kreslí pouze věci podstatné pro grafické vyjádření návrhu, části návrhu před tím, než se začne programovat.

Důležitá je srozumitelnost, rychlost nakreslení a snadnost změny či navržení alternativ řešení.

Kreslení detailních návrhů

Cílem je zaznamenat kompletní návrh, nebo kompletní realizaci. Při kreslení návrhu by měl analytik obsáhnout všechny prvky tak, aby programátor byl schopen vytvořit program bez velkého přemýšlení nad věcnou oblastí (pro programátora by neměla vzniknout potřeba konzultace s uživatelem). Při kreslení detailních návrhů se obvykle používají specializované programy (CASE), které jsou schopny sdílet informace mezi jednotlivými modely a kontrolovat konzistenci návrhu. Při dokumentaci programu se často používá nástroje pro generování diagramů z vlastního kódu aplikace.

UML jako programovací jazyk

Při tomto použití vývojář nakreslí UML diagramy, ze kterých se vygeneruje přímo spustitelný kód. Toto vyžaduje specializované nástroje a velmi přesné vyjadřování v UML diagramech. V této

souvislosti se velmi často používá pojem Model Driven Architecture (MDA), což je další standard skupiny OMG, který se snaží standardizovat použití UML jako programovací jazyk.

Metamodel

Tento pohled používají autoři UML a autoři CASE nástrojů - nedívají se na UML jako na diagramy, pro ně je základem UML metamodel (diagramy jsou pouze grafickou reprezentací metamodelu). Při tomto přístupu se často používá pojem model místo pojmu diagram, např. místo diagramu tříd se používá pojem model tříd. Metamodel se popisuje pomocí Meta-Object-Facility (MOF) - abstraktního jazyka pro specifikaci, vytváření a správu metamodelů (další standard OMG). Pro výměnu metamodelů se používá XMI - na XML založený standard (součást standardu UML) [22].

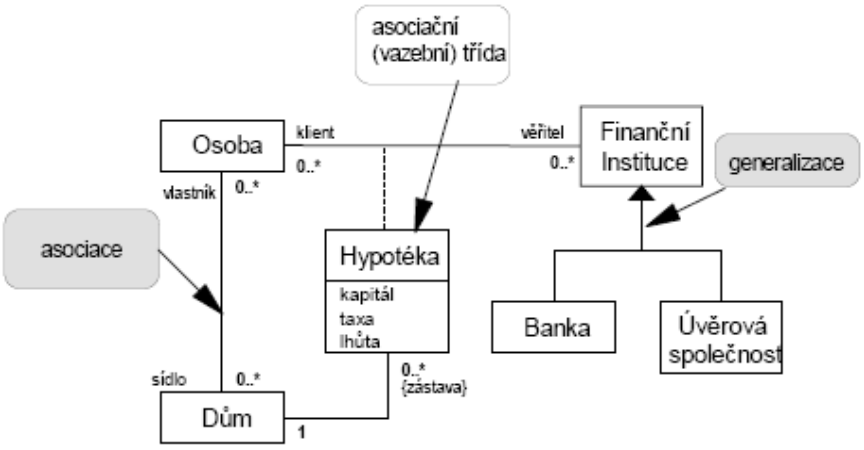
2.1.1.7 Základní modelovací koncepty UML

Tato část popisuje základní modelovací koncepty, včetně diagramů.

Diagram tříd – statická struktura

Diagramy tříd patří bezesporu k nejčastěji používaným nástrojům UML a tvoří základ všech prostředků OOAD. Tyto diagramy jsou většinou vytvářeny již ve fázi analýzy často jako pojmové modely, jejichž úkolem je formálněji definovat jednotlivé termíny používané ve studované problémové oblasti. S postupným přibližováním k fázi implementace jsou diagramy tříd poměrně zásadně přehodnocovány a v ideálním případě zpracovávány až do podoby grafického modelu ekvivalentního zdrojovému kódu. Diagramy tříd zobrazují *statickou stránku* systému, především vztahy mezi třídami. Viz obr. 3. [15].

Při tvorbě těchto diagramů se doporučuje dodržovat nepsané pravidlo 7 + 2, které říká, že v jednom diagramu je vhodné zobrazit 7, maximálně 9 tříd. Při respektování tohoto doporučení se většinou výraznou měrou zpřehlední výsledné modely, nicméně na druhou stranu vyvstává problém vhodného rozdělení systému na dílčí oblasti (v terminologii UML jsou jimi tzv. packages).



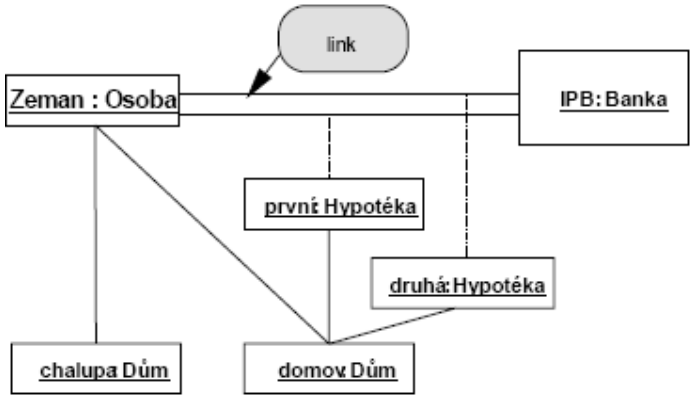
Obr. 3. Diagram tříd

Diagram popisuje entity v systému a jejich základní (obecné) vztahy. Jednotlivé třídy mohou být blíže charakterizovány svými atributy a operacemi (metodami objektů), jež provádějí, či připouštějí. UML zde obsahuje speciální konstrukce pro vyjádření dvou základních druhů hierarchických abstrakcí, které jsou zde vyjadřovány jako zvláštní druhy asociací: generalizace (viz dva druhy Finanční Instituce v příkladu) i agregace (případně "kompozice", jako "silnější agregace"). Nejčastější chybou při tvorbě diagramů tříd (a celkově při objektovém modelování) je zaměňování pojmů objekt a třída, čímž může dojít k zavlečení poměrně závažných nekonzistencí do vytvářeného modelu.

Diagram instancí objektů (zvláštní typ Diagramu tříd)

Popisuje instance objektů a jejich vztahů (linky) v konkrétním případě. Tento diagram je při analýze použitelný jako doplněk k diagramu tříd, zejména pro vysvětlení detailních charakteristik modelů, které nejsou dostatečně popsatelné na úrovni tříd a asociací.

Jsou zde vidět konkrétní instance objektů Osoba, Dům, Finanční Instituce (resp. Banka, v případě těchto instancí) a Hypotéka, jakož i instance asociací mezi nimi (tzv. linky). Tento diagram je zobrazen na obrázku 4. [15].



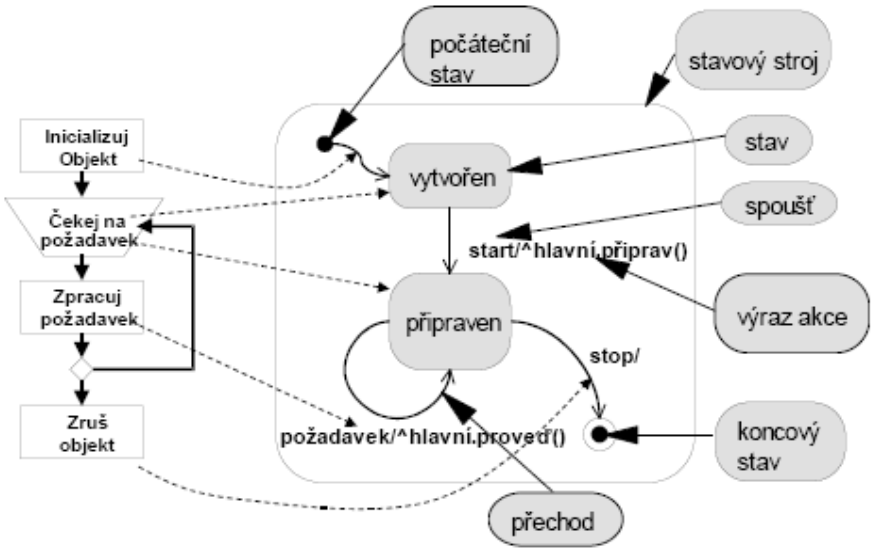
Obr. 4. Diagram instancí

Diagram stavového stroje

Diagram stavů a přechodů, jak je někdy tento prostředek nazýván, slouží pro modelování životního cyklu části systému, který svým rozsahem odpovídá jednomu objektu. Je obdobou známého diagramu stavů a přechodů (State Transition Diagram – STD) a stejně jako on je též alternativním způsobem popisu algoritmu. Přechod mezi jednotlivými stavy bývá vyvolán podnětem z vnějšího okolí, nejčastěji ve formě zprávy zaslané příslušnému objektu nebo jinou externí událostí. Validní stav objektu je definován přípustnými hodnotami jeho atributů, přechod mezi stavy je pak vlastně vyjádřením změny hodnot těchto atributů.

Životní cyklus objektu nějak začíná a zpravidla i nějak končí (v případě, že se nejedná o objekt perzistentní, který je například uchováván v objektové databázi). Z implementačního hlediska je start

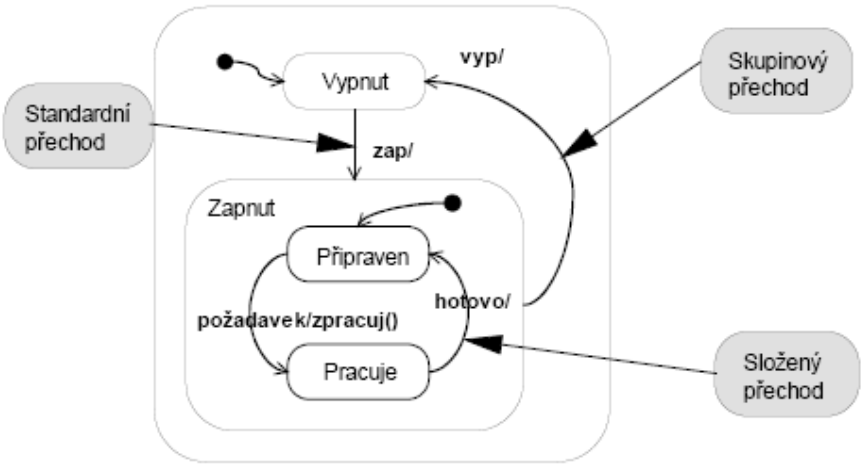
stav ekvivalentní alokaci paměti, resp. Zavolání konstruktoru třídy, naopak stop stav je realizován vykonáním destruktoru, případně prostředky automatické správy paměti (garbage collection). Jak vypadá diagram stavového stroje můžeme vidět na obrázku 5. [15].



Obr. 5. Diagram stavového stroje

Hierarchické stavy a přechody

Hierarchické stavy a přechody umožňují postupné čištění modelu a přehlednější nazírání na složité chování objektu. Jak stavy, tak přechody mezi nimi mohou být na různých hierarchických úrovních. Příklad vyobrazený na obrázku 6. ilustruje jednotlivé možnosti: standardní přechod, přechod mezi skupinovým a normálním stavem, přechod mezi pod-stavy (složený přechod) [15].



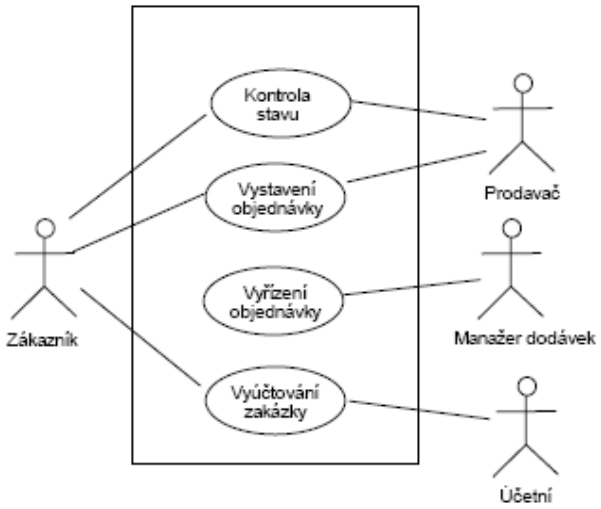
Obr. 6. Diagram hierarchický stavů a přechodů

Diagram užití (Use Case Diagram)

Vymezení hranic systému je v UML podpořeno dynamickým pohledem modelovaným prostřednictvím *diagramů užití*. Diagram užití ilustruje použití části systému pro určitý případ (resp. použití určité funkčnosti systému).

Tento diagram (model) zobrazuje základní vztah systému k jeho okolí, tzv. aktérům. Aktéry jsou nejčastěji samotní uživatelé systému, ale může se jednat i o další HW s SW prostředky, jiné IS, instituce, které se systémem komunikují atd. Každý případ užití pak představuje jeden z možných způsobů použití systému, jednu z možných cest komunikace aktér - systém. Konkrétním případem užití může být například použití systému za účelem zjištění aktuálního stavu skladových zásob, zpracování faktury atp. V rámci tvorby těchto diagramů modelujeme vztah systému a jeho okolí, nikoliv vzájemnou interakci, tj. způsob, jakými jsou jednotlivé případy užití zajištěny interními funkcemi systému. Největší úskalí při používání tohoto prostředku spočívá v odhadnutí "správné úrovně abstrakce".

Na obrázku 7. je vyobrazen příklad znázorňující jednotlivé prvky funkčnosti systému (Kontrola stavu, Vyúčtování zakázky, Vystavení objednávky, Vyřízení objednávky) a jejich užití jednotlivými aktéry (Zákazník, Účetní, Manažer dodávek, Prodavač). Širší notace tohoto diagramu umožňuje též popisovat různé druhy vztahů mezi jednotlivými případy užití (např. “obsahuje”, “rozšiřuje”, “je zobecněním”) [15].



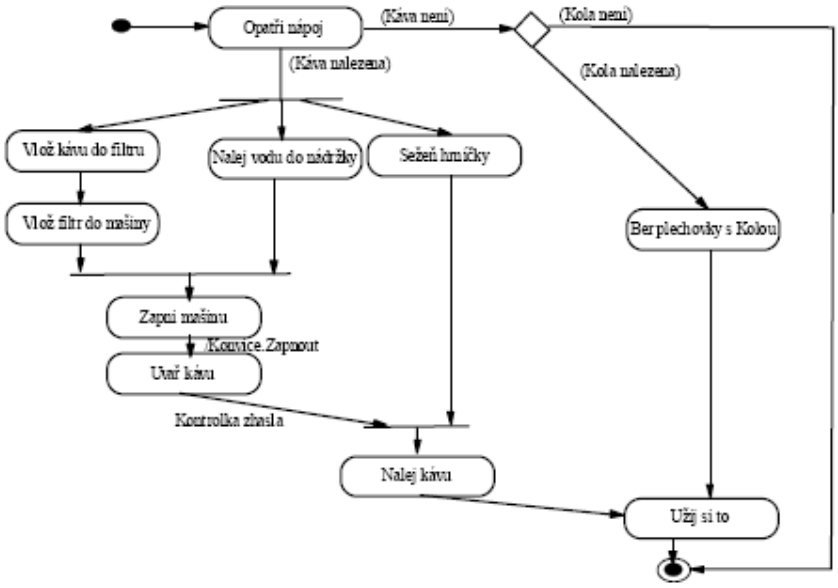
Obr. 7. Diagram užití (Use Case Diagram)

Diagram činností (Activity Diagram)

Jako jistou obdobu stavových diagramů můžeme chápat i diagramy činností (aktivit), kde jednotlivými stavy rozumíme aktivity a přechod mezi aktivitami je vyvolán dokončením aktivity stávající. Diagram činnosti se zpravidla vztahuje k jednomu případu užití, případně k jedné metodě

objektu. Pomocí diagramu činností modelujeme tentokrát dynamický tok řízený nikoliv vnějšími událostmi, ale interními podněty.

Diagramy činností lze použít i pro základní schematickou tvorbu GUI, což může pomoci odhalit případná nedorozumění ohledně logické výstavby (návaznosti dialogů) GUI. Diagram aktivit je zobrazen na následujícím obrázku (viz obr. 8.) [15].



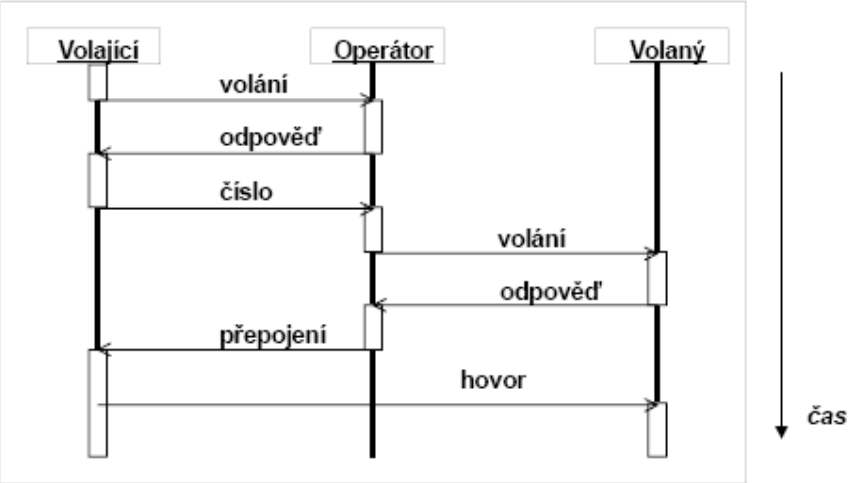
Obr. 8. Diagram činností (Activity Diagram)

Jak je vidět z příkladu, je zde v centru zájmu samotný proces, nikoliv zúčastněné objekty, či aktéři. Pro přesnější specifikaci vztahu k ostatním modelům, popisujícím objekty, je tedy zpravidla nutné tento diagram doplnit ještě dalšími modely chování (diagramy posloupností a diagramy spolupráce – viz níže).

Diagram posloupností

Popisuje přípustné interakce mezi jednotlivými objekty z hlediska jejich společného procesu (činnosti). Diagramy posloupností (sekvenční diagramy) se vytvářejí většinou přímo z diagramů případů užití. K jednomu případu užití může existovat několik sekvenčních diagramů, které modelují interakci objektů v rámci komunikace aktéra se systémem.

Sekvenční diagram obsahuje dvě dimenze. V horizontální rovině se zobrazují jednotlivé objekty, zatímco vertikální rovina představuje tok času. Zprávy posílané mezi jednotlivými identifikovanými objekty mohou být různého druhu. Záleží-li na jejich bližším odlišení, lze klasifikovat zprávy asynchronní, vnořené, zprávy představující návratové hodnoty apod.



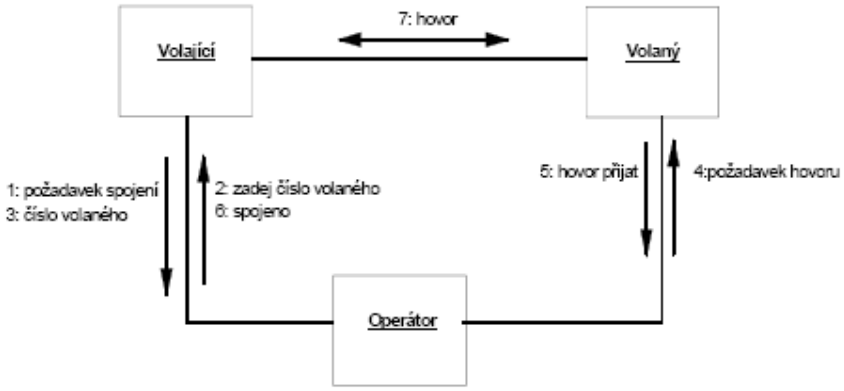
Obr. 9. Diagram posloupností

Jak je vidět na obrázku 9., diagram ilustruje volání s pomocí operátora, kdy jednotliví aktéři (Volající, Operátor, Volaný) si vzájemně předávají aktivitu. Každé předání aktivity by mělo odpovídat vztahu mezi zúčastněnými objekty, jakož i mezi jejich příslušnými operacemi. Souhrn zde znázorněných aktivit jednoho objektu by měl odpovídat jeho životnímu cyklu (stavovému stroji).

Diagram spolupráce

Podobně, jako objektový diagram, i diagram spolupráce může být smysluplné vypracovat na úrovni specifikací (kdy popisujeme role klasifikátorů a vztahů), nebo instancí (kdy popisujeme objekty, linky a stimuly). Následující příklad diagramu spolupráce ilustruje též, co má tento diagram společného s diagramem posloupností a v čem se liší (viz předchozí příklad). Narozdíl od diagramu posloupností je zde ve středu zájmu spíše strukturální korespondence objektů v jejich společném procesu, nežli hledisko časové. Viz obr. 10.

Chceme-li v jednom diagramu znázornit jak strukturu objektů, tak jejich dynamické chování, použijeme s výhodou diagramů spolupráce, které jsou vedle sekvenčních diagramů dalším prostředkem, jehož těžiště tkví v modelování dynamiky. Na rozdíl od sekvenčních diagramů je však znatelně obtížnější vysledovat návaznost jednotlivých posílaných zpráv zajišťujících samotnou funkcionalitu systému. Zatímco v sekvenčních diagramech je tato návaznost zřejmá z vertikálního uspořádání celého diagramu, v diagramech spolupráce je následnost zobrazena pořadovým číslem, kterým jsou zprávy uvozeny [15].

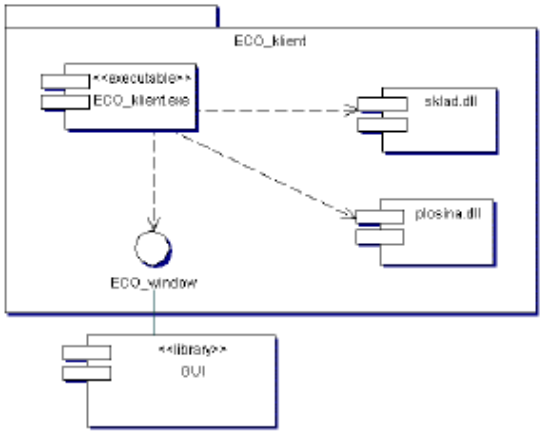


Obr. 10. Diagram spolupráce

Diagram komponent

Poskytuje fyzický pohled na systém. Smyslem je ukázat závislosti komponent navrhovaného systému na ostatních součástech (např. závislost SW na jiných modulech, knihovnách apod.). Diagram se vytváří na různých úrovních granularity. Toto je zobrazeno na obrázku 11.

Komponentou ve smyslu UML je i určitá množina tříd, které spolu realizují požadovanou funkčnost. Diagramy tříd je tak možné "rozsekat" na abstraktnější celky, s kterými můžeme následně pracovat jako se základními stavebními kameny [15].

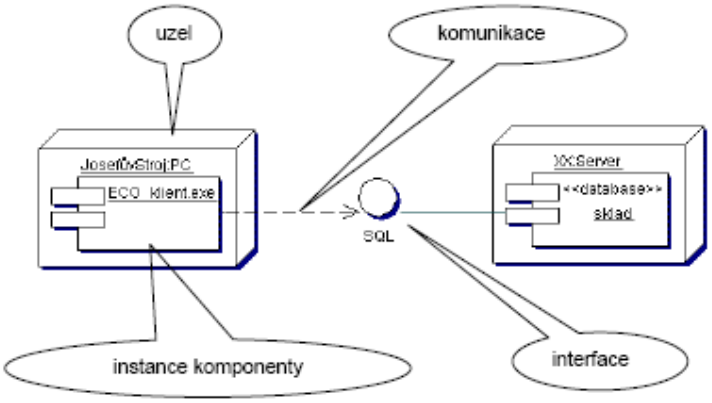


Obr. 11. Diagram komponent [18]

Diagram zavedení (nasazení)

Diagram znázorňuje fyzické začlenění systému do příslušného hardwarového prostředí. Je určen pro implementační fáze, především zobrazuje vztahy mezi částmi systému tak, jak vypadají v době samotného vykonávání. Diagramy nasazení zobrazují rozložení jednotlivých SW komponent na HW zdrojích a jejich spolupráci, rozmístění HW a SW prostředků v lokalitách, topologie používaných sítí, druhy a využití komunikačních prostředků atp. Notace vychází z diagramu komponent, ale přidává představu o příslušném uzlu: uzel reprezentuje fyzický nebo virtuální stroj. Názvotvorná konvence

(jako u sekvenčních diagramů): [instance name] : [instance type] (např. "w3reporting.myco.com : Application Server"). Diagram zavedení je zobrazen na obrázku 12.



Obr. 12. Diagram zavedení (nasazení) [18]

2.1.2 OCL

Tato kapitola popisuje jazyk OCL, jeho historii, vývoj a případy použití. Také obsahuje ukázky OCL výrazů a diagram tříd, ke kterému se vztahují.

2.1.2.1 Co je to OCL

Object Constraint Language (OCL) je formální jazyk užívaný pro popis výrazů na UML modelech [21]. Tyto výrazy typicky specifikují neměnné podmínky které musí platit pro modelované systémy nebo dotazy nad objekty popsané v modelu. Jazyk byl navržen společností IBM a slouží k popisu omezení (*constraints*), podmínek (*pre-/post-conditions*), apod. v různých částech objektově orientovaných modelů popsaných v UML [5]. Všimněte si, že když OCL výrazy jsou ohodnocené, nemají vedlejší efekty; to jest jejich vyhodnocení nemůže změnit stav odpovídajícího provádění systému.

OCL výrazy mohou být použity pro specifikovaní operace/akce, jejichž vykonáním změníme stav systému.

Tvůrci UML modelů mohou užívat OCL pro upřesnění aplikačně - specifického omezení v jejich modelech. Také mohou užívat OCL pro zpřesnění dotazů v UML modelu, které jsou úplně nezávislé na programovacím jazyku.

2.1.2.2 Historie OCL

V roce 1996 vyhlásila organizace Object Management Group (OMG, http://www.omg.org), aby libovolné společnosti navrhly přístupy k objektové analýze a designu. V lednu 1997 IBM spolu se

společností ObjecTime na tuto výzvu zareagovaly a součástí jejich návrhu byla právě specifikace jazyka Object Constraint Language.

Během roku 1997 spolupracovaly IBM a ObjecTime spolu s ostatními partnery kolem UML a výsledkem byl návrh UML verze 1.1.

Primární příspěvek IBM k UML 1.1 je tedy OCL. OCL byl vyvinut Josem Warmerem, jako jazyk pro interní business modelování v IBM, a vycházel z metodiky Syntropy, jejímiž autory byli Steve Cook a John Daniels. Object Constraint Language je v rámci UML využíván k tomu, aby pomohl formalizovat sémantiku samotného jazyka a poskytl uživatelům možnost přesnějšího vyjádření omezení modelu [22].

2.1.2.3 Současnost (OCL a UML 2.0)

Donedávna bylo OCL součástí specifikace UML 1.5. Bohužel OCL narozdíl od UML není založeno na žádném konkrétním metamodelu (model, popisující model). Proto je integrace OCL s UML založena spíše na neformálním přístupu a vychází z přirozeného jazyka. Podle Klasse Objecten: The OCL Center je kvůli neexistenci metamodelu také zamlžená hranice mezi sémantikou a notací jazyka. To způsobuje problémy s přesnou interpretací výrazů jazyka. Zároveň je znemožněna přenositelnost *constraintů* mezi různými prostředími pomocí XMI (XML Model Interchange - jednotný standard pro přenos modelů na bázi XML). Bez metamodelu je také obtížné zajistit konzistenci mezi různými implementacemi OCL.

V rámci návrhu jazyka UML 2.0 jsou tedy zároveň navrženy změny v jazyce OCL, který ponese stejné označení verze. Jedná se zejména o tyto změny:

- K OCL by měl být definován přesný metamodel.
- Měly by být přidány další koncepty, které zajistí lepší práci s komponentami (prvek UML) a umožní definice behaviorálních *omezení* (omezení, týkající se chování).
- Měly by být definovány rozšiřující mechanismy, aby bylo možné jazyk OCL používat v různých doménách (např. mezi rozšiřující mechanismy UML patří stereotypy, přidané hodnoty, apod. Jde o to, zajistit obdobné možnosti i pro OCL).
- Pozice OCL v rámci UML by měla být lépe definována a měla by být posunuta směrem k obecnému výrazovému prostředku v UML. OCL by tedy neměl sloužit pouze pro definici omezení, jako je tomu dnes, ale jeho použití by mělo být univerzálnější.

2.1.2.4 Proč používat OCL

UML diagram, jako například diagram tříd (class diagram), není typicky dost popisný na to, aby poskytl všechny důležité aspekty specifikace. Je zde, kromě jiného, potřeba popsat další omezení o objektech v modelu. *Omezení* je čistě výraz, který má po vyhodnocení vzhledem ke stavu systému nějakou hodnotu. *Omezením* se rozumí pravidlo (zakazující, přikazující nebo vymezující), které se týká jedné nebo více hodnot nebo části objektově orientovaného modelu nebo systému. Stav daného

objektu nebo modelu je tedy *omezením* spíše sledován, nikoliv vytvářen. To je důležité východisko, protože znamená, že v objektovém modelu lze *omezení* libovolně měnit, aniž bychom narušili funkčnost modelu. Taková omezení jsou často popsána v přirozeném jazyce. Praxe ukázala, že to bude mít vždy za následek mnohoznačnosti. Za účelem psát jednoznačná omezení, byly vyvinuté tzv. formální jazyky. Nevýhoda tradičních formálních jazyků je, že jsou použitelné pro osoby matematickým vzděláním, ale obtížně použitelné pro průměrného obchodníka nebo systémového modeláře.

OCL byl vyvinut, aby vyplnil tuto mezeru. Je to formální jazyk, který je snadný pro čtení a zápis. Byl vyvinut jako obchodní modelový jazyk uvnitř pojišťovacího oddělení IBM a má své kořeny v Syntropy metodě.

Dále si zde povíme o několika konkrétních rysech OCL. Kompletní popis OCL je zahrnut ve vytvořené e-learningové aplikaci.

OCL je ryzí specifikační jazyk, proto OCL výraz je zaručeně bez vedlejšího efektu. Při vyhodnocování OCL výrazu je vracena hodnota. Ta v modelu nemůže cokoliv změnit. Toto znamená, že se stav systému nikdy nezmění kvůli vyhodnocení OCL výrazu, třebaže OCL výraz může být užívaný pro upřesnění změny stavu (např. v následné podmínce).

OCL není programovací jazyk, proto není možné psát programovou logiku nebo řízení toku v OCL.Nemůžeme vyvolávat procesy nebo aktivovat nedotazující se operace uvnitř OCL. Protože OCL je v první řadě modelový jazyk OCL, výrazy nejsou samozřejmě přímo proveditelné.

OCL je typový jazyk, takže každý OCL výraz má datový typ. Aby byl správně vytvořen, musí se OCL výraz přizpůsobit typu shodného s pravidly jazyka. Například nemůžeme srovnávat celé číslo (Integer) s řetězcem (String). Každý klasifikátor definovaný uvnitř UML modelu reprezentuje různý OCL datový typ. Navíc, OCL zahrnuje soubor doplňkových předdeklarovaných datových typů [21].

OCL je specifikační jazyk, pro který platí, že všechny implementační problémy jsou mimo rozsah a nemohou být vyjádřený v OCL. Vyhodnocení OCL výrazu je okamžité. Toto znamená, že stavy objektů v modelu se nemohou změnit během vyhodnocení.

Příklad, k čemu slouží OCL:

V objektovém modelu máme dvě třídy, majitel účtu a účet samotný. Jakým způsobem namodelujeme v diagramech UML fakt, že majitel nesmí při vybírání z bankomatu překročit svůj denní limit, případně celkový zůstatek na účtu ? Anebo příklad programátorský. Jak v UML definovat podmínku, že musí být při operaci pop v zásobníku uložen alespoň jeden prvek (jinými slovy že nelze vyvolat metodu pop nad prázdným zásobníkem ...)? Při hlubším zamyšlení zjistíme, že se vždy jedná o obdobné případy, pro které ovšem dříve UML nenabízelo vhodné modelovací prostředky. Ze všech myslitelných řešení bylo většinou nejlepší použít v modelech všudypřítomnou textovou poznámku, která to dokázala alespoň trochu ošetřit [15].

2.1.2.5 Případy použití OCL

OCL může být použit pro několik různých účelů:

- jako dotazovací jazyk
- pro specifikování invariant na třídách a typů v modelů tříd
- pro upřesnění datového typu invarianta pro stereotypy
- pro popisování předchozích a následných podmínek na operacích a metodách
- pro popisování Guards
- pro upřesnění cíle (nastavení) zpráv a akcí
- pro upřesnění omezeních operací
- pro zpřesnění odvození pravidel pro atributy každého výrazu v UML modelu

Každé *omezení* je spojeno s jedním prvkem modelu. Tento prvek se nazývá kontextem *omezení*. V OCL existuje speciální klíčové slovo ***self***, které referuje na objekt, který je aktuálním kontextem. Pokud je *omezení* typu *invariant*, pak je kontextem třída. Je-li *omezení* typu podmínka před spuštěním, pak je kontextem operace třídy. Je-li *omezením* hlídaný přechod (*guard*), pak je kontextem ten stav, z něhož přechod vychází. Ve všech uvedených případech referuje klíčové slovo ***self*** na instanci třídy, pro níž je omezení vyhodnocováno. Jde tedy o třídu, která definuje danou operaci, nebo o třídu, pro níž je vytvářen daný stavový diagram.

Ukázka OCL výrazu

Pro názornost uvádím, jak vypadá OCL výraz společně s diagramem tříd, ke kterému se vztahuje. Nejjednodušším příkladem je omezení typu *invariant* nad atributem. Následující zápis předpokládá, že v modelu existuje třída ZAKAZNIK, která má atribut VEK, a nastavuje pro tento atribut omezení:

context ZAKAZNIK **inv**: VEK >= 18

Omezení typu *invariant* lze také definovat nad asociací. Předpokládejme, že v modelu existuje navíc třída OBCHODNI_ZASTUPCE, s níž má třída ZAKAZNIK nastavenou asociaci. Role třídy OBCHODNI_ZASTUPCE v této asociaci je nazvaná *obchodnik* a má multiplicitu 1. Dále předpokládejme, že třída OBCHODNI_ZASTUPCE obsahuje atribut *uroven_znalosti*, který chceme v rámci asociace nějak omezit. Omezení bude zapsán takto:

context ZAKAZNIK **inv**: obchodnik.uroven_znalosti >= 5

Trochu složitější je definice omezení nad kolekcí prvků. Do situace, kdy potřebujeme definovat takové omezení, se dostáváme tehdy, má-li asociace vyšší multiplicitu než 1. Pak se omezení může týkat všech asociovaných instancí, nebo kolekce jako celku (např. omezovat její velikost). Předpokládejme asociaci mezi třídami OBCHODNI_ZASTUPCE a ZAKAZNIK, jejíž druhá role má

název *zakaznici* a multiplicitu 1..* na straně u třídy ZAKAZNIK. Omezení můžeme napsat například takto:

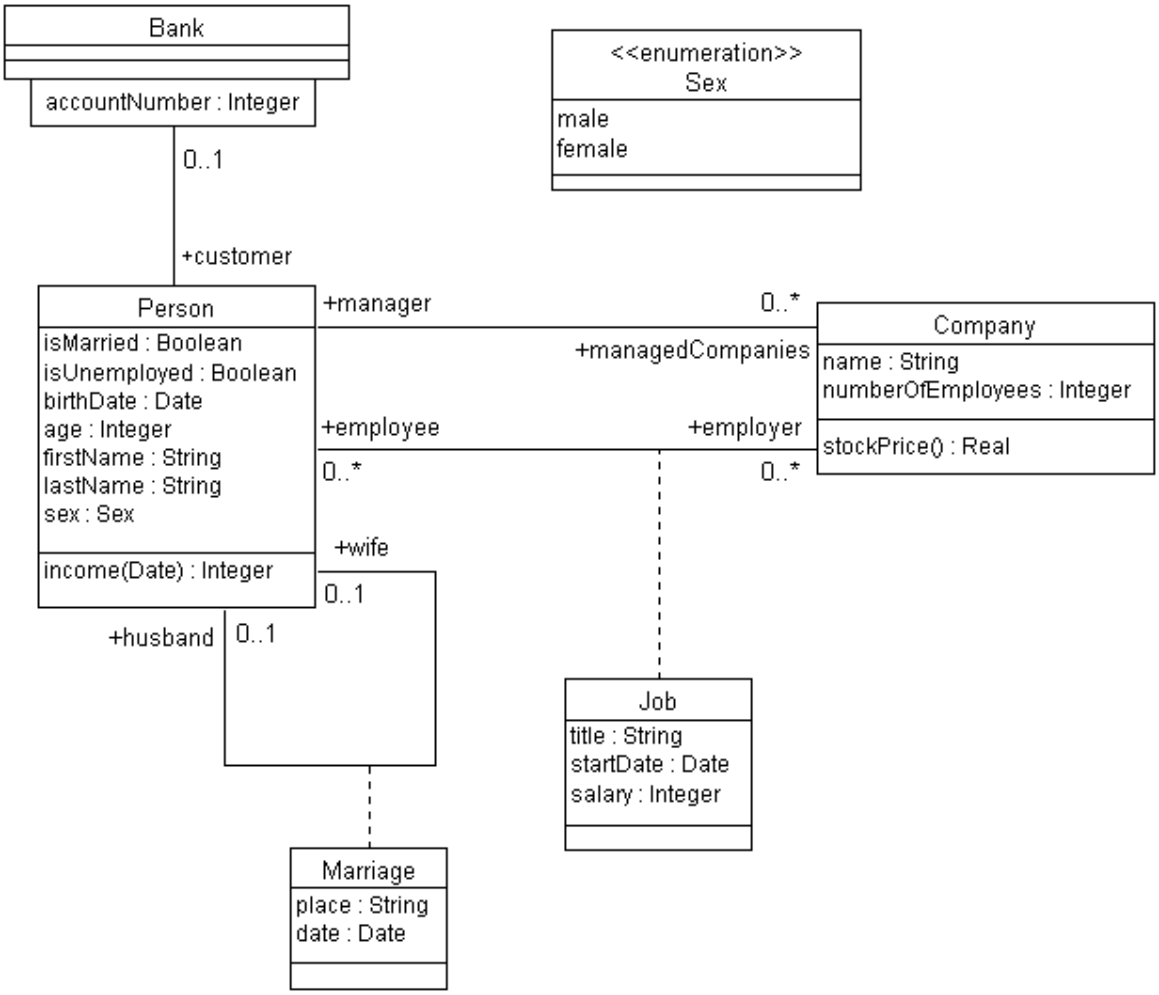
context OBCHODNI_ZASTUPCE **inv**: zakaznici->size() <= 100
and zakaznici->forAll(z: ZAKAZNIK | z.VEK >= 40)

V podmínkách před a po spuštění mohou být využity parametry (argumenty) operací. Navíc je zde definováno speciální klíčové slovo *result*, které odkazuje na návratovou hodnotu operace a může být použito pouze v podmínkách po spuštění. V příkladu se ke třídě OBCHODNI_ZASTUPCE přidává operace prodat:

context OBCHODNI_ZASTUPCE::prodat(predmet: TPredmet): Real
pre: self.seznamProdavanehoZbozi->includes(pedmet)
post: not self.seznamProdavanehoZbozi->includes(predmet) **and** result = predmet.cena

Jak již bylo řečeno, samotné vyhodnocení *omezení* nemění žádným způsobem proměnné systému. Omezení říká: "toto by mělo být tak a tak". Pokud pro určitý objekt nenabývá omezení hodnoty *pravda*, jinými slovy, je ve stavu *broken* (*zlomené omezení*), tak jediné, co z toho můžeme vyvodit, je to, že sledovaný objekt není v korektním stavu, neodpovídá specifikaci. To, jestli se jedná o fatální chybu, nebo jen o nevýznamnou odchylku, neřeší OCL a nelze to pomocí tohoto jazyka vyjádřit [5].

Následuje diagram tříd (viz obr. 13.), ke kterému se předchozí OCL výrazy vztahují.



Obr. 13. Diagram tříd pro OCL výrazy

2.2 CASE Rational Rose

Tato kapitole se věnuje prostředí CASE Rational Rose a popisuje k čemu slouží. Také se věnuje jeho vlastnostem a obsahuje ukázky diagramů vymodelovaných tímto nástrojem.

2.2.1 Prostředí CASE Rational Rose

Rational Rose je objektově orientovaný Unified Modeling Language (UML) softwarový návrhový nástroj určený k vizuálnímu modelování a budování komponent enterprise- level software aplikace. Stejným způsobem jako divadelní ředitel nastíní hru, softwarový návrhář užívá Rational Rose, aby vizuálně vytvořil (model) konstrukce pro aplikaci zhruba koncipující třídy s actory (hůlková čísla), užívající case elementy (ovály), objekty (obdélníky) a zprávy/vztahy (šipky) v časovém diagramu používáním táhni a pusť symbolů. Rational Rose dokumentuje diagram tak, jak je zkonstruovaný a

potom vyrábí kód podle volby návrháře ať už C++, Visual Basic, Javy, Oracle8, CORBA nebo Data Definition Language.

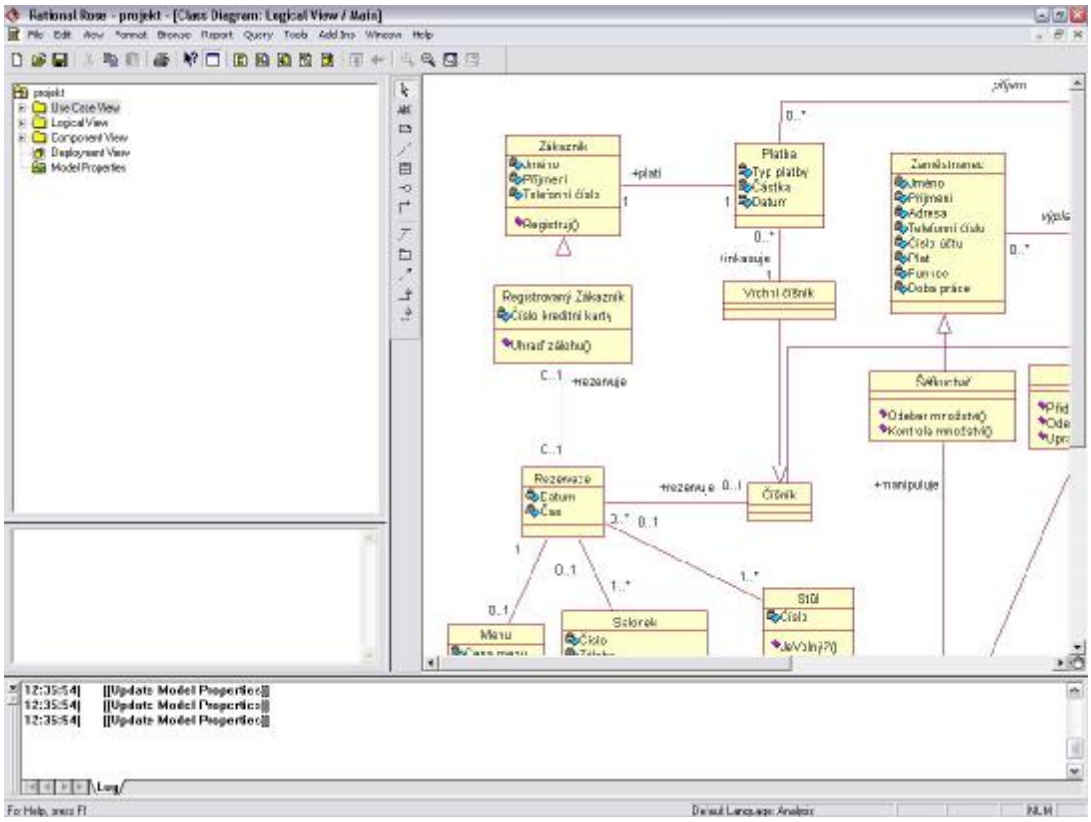
2.2.1.1 Vlastnosti

Rational Rose Enterpise je nástroj pro vizuální modelování. Mezi jeho výhody patří [7]:

- Bezkonkurenční podpora UML přímo od jeho tvůrců, společnosti Rational Software a jejích odporníků, Jamese Jacobsona, Jima Rumbaugha a Grady Booche.
- Podpora business proces modelování, modelování architektury systému a datového modelování.
- Podpora relačních databází (Oracle, MS SQL Server, DB2, Sybase).
- Objektově relační mapování pro sledování přechodu mezi objektovým modelem a návrhem databáze a naopak.
- Společné prostřední a společná notace pro všechny modelovací aktivity a všechny členy týmu, včetně business analytiků, architektů systému a návrhářů databází.
- Round-trip engineering umožňuje snadno dokumentovat stávající systémy a při vývoji rychle předcházet mezi návrhem systému a jeho fyzickou implementací v kódu a naopak.
- Podpora běžně používaných vývojových prostředí (C++, Visual C++, Visual Java, Centura, PowerBuilder, Delphi, Smalltalk atd.).
- Round-trip engineering umožňuje databází snadno dokumentovat existující databáze, migrovat schémata mezi různými databázemi a průběžně synchronizovat model s implementací databáze a naopak.
- Podpora modelování internetových aplikací.

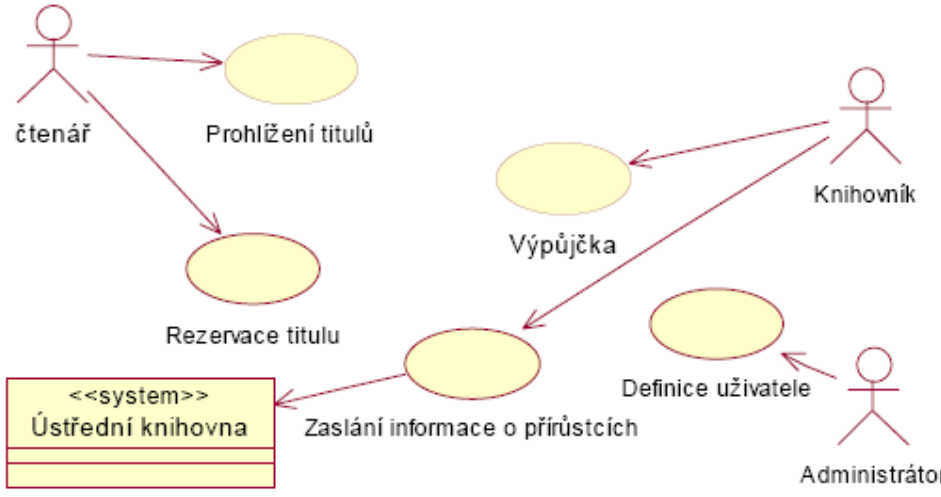
2.2.1.2 Ukázka prostředí CASE Rational Rose a diagramů v něm modelovaných

Nyní si ukážeme jak vypadá prostředí programu CASE Rational. Obrázek 14. vyobrazuje prostředí programu CASE Rational Rose. Zobrazuje ukázku diagramu tříd vymodelovaného v tomto prostředí.

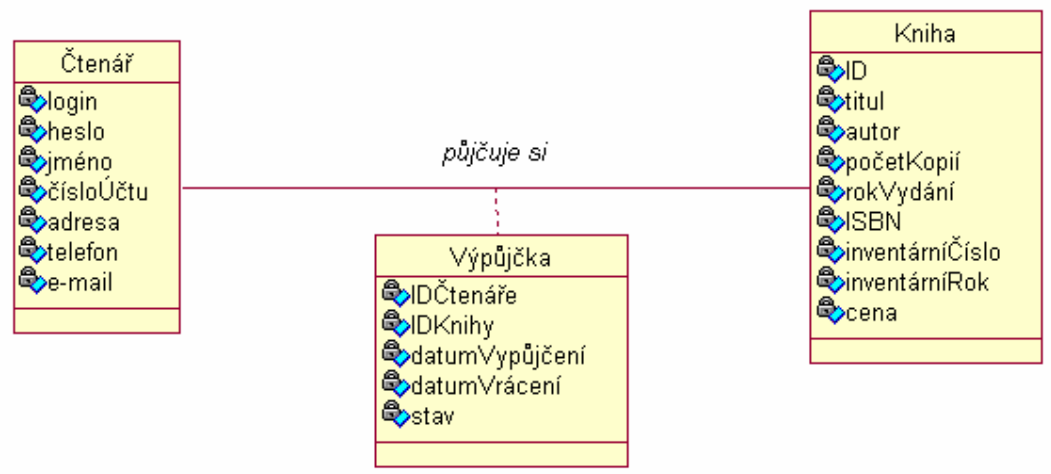


Obr. 14. Prostředí CASE Rational Rose

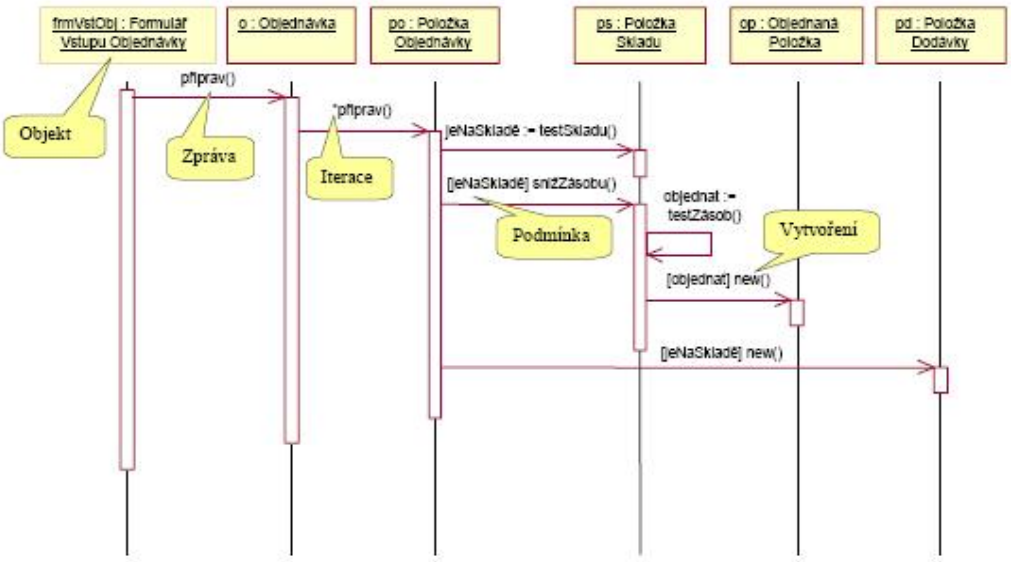
Další obrázky (viz. obr. 15.,16.,17.) ukazují, jak budou vypadat jednotlivé typy diagramů vymodelované v prostředí CASE Rational Rose. Tyto diagramy se nevztahují k problematice OCL ani UML, jsou zde uvedeny pro názornost a popis možností nástroje CASE Rational Rose.



Obr.15. Diagram použití (Use case diagram) modelovaný v Prostředí CASE Rational Rose



Obr. 16. Diagram tříd (Class diagram) v prostředí CASE Rational Rose - Příklad asociace na diagramu tříd



Obr. 17. Diagram sekvence modelovaný v prostředí CASE Rational Rose (Sequence diagram)

V aplikaci popisující specifikaci omezení objektů systému si vystačíme s diagramem tříd.

3 E-learning a tvorba elektronických dokumentů

E-learning nebo taktéž elektronické vzdělávání je v podstatě výuka podporovaná počítačem. Je to proces sdílení informací a poskytování vzdělávacích a výcvikových aktivit v elektronické podobě. E-learning můžeme definovat jako výučování prováděné na osobním počítači (PC) prostřednictvím internetu, intranetu nebo pomocí CD/DVD ROM mechanik a médií.

3.1 E-learning

V této kapitole se na e-learning podíváme trochu podrobněji. Vysvětlíme si přesněji, co to je, jak a proč se používá.

3.1.1 Co je to vlasně e-learning

Budeme li hledat definice e-learningu v odborné literatuře či elektronických zdrojích, narazíme na silnou terminologickou nejednotnost. Ta je dána zejména faktem, že e-learning lze definovat různě s ohledem na danou edukační realitu/prostředí. Proto budeme v rámci definic e-learningu rozlišovat definice v *širším a užším slova smyslu* [4].

3.1.1.1 E-learning v širším slova smyslu

E-learning v širším slova smyslu je definován zejména jako aplikace nových multimediálních technologií a Internetu do vzdělávání za účelem zvýšení jeho kvality posílením přístupu ke zdrojům, službám, k výměně informací a ke spolupráci. Tato definice chápe e-learning jako jakékoli využívání informačních technologií multimediálního charakteru ke zlepšení kvality a efektivity vzdělávání. To znamená, že za e-learning můžeme v širším slova smyslu považovat například používání výukových CD-ROMů v rámci prezenční školní výuky.

E-learning můžeme chápat jako multimediální podporu vzdělávacího procesu za použití moderních informačních a komunikačních technologií (ICT), jejichž primárním úkolem je zvýšit kvalitu a dostupnost vzdělávání.

3.1.1.2 E-learning v užším slova smyslu

V užším slova smyslu je e-learning chápán zejména jako vzdělávání, které je podporované moderními technologiemi a které je realizováno prostřednictvím počítačových sítí – intranetu a zejména Internetu. Tato definice již popisuje e-learning, jak jej zná většina populace – jako vzdělávání po Internetu.

V této definici již narážíme na fakt, že e-learning umožňuje předávat vzdělávací obsahy (learning content) prostřednictvím digitalizované informace ke komukoli, kdo má přístup k počítačové síti. Tedy e-learning **je vzdělávání umožňující svobodný a neomezený přístup k informacím**.

E-learning si pro zjednodušení představme jako soustavu moderních nástrojů, postupů a procesů, s jejichž pomocí můžeme efektivně (ale přiměřeně) působit na co největší množství smyslů a umožnit realizovat kvalitní a funkční proces řízeného sebeučení (učení bez učitele). Na zrak působíme prostřednictvím přehledného distančního textu doplněného o obrazovou dokumentaci, fotografie, videoukázky, na sluch pak působíme pomocí hudebních ukázek, čteného slova. Na smysly lze působit také kombinovaně – s pomocí multimediálních ukázek, interaktivních animací či vizualizací. E-learning nám nabízí širokou škálu technologických možností, které mohou zefektivnit studium, v principu však také znehodnotit (viz dále).

Některá pracoviště chápou e-learning jako aktuální didaktický prvek pro distanční vzdělávání i pro využití v rámci prezenčního vzdělávání. Představuje multimediální podporu vzdělávacího procesu využívající informační a komunikační technologie pro dosažení vyšší kvality a efektivity vzdělávání (Národní centrum distančního vzdělávání, 2005). Tato definice je poněkud matoucí – e-learning není didaktický prvek.

Zastánci procesuálního pojetí e-learningu definují eLearning jako vzdělávací proces, využívající informační a komunikační technologie k tvorbě kursů, k distribuci studijního obsahu, komunikaci mezi studenty a pedagogy a k řízení studia. V praxi neexistuje jednotná definice e-learningu. Shrnující definici, která využívá společné rysy všech ostatních, naleznete níže.

3.1.2 Definice e-learningu

E-learning chápeme jako multimediální podporu vzdělávacího procesu s použitím moderních informačních a komunikačních technologií, které jsou zpravidla realizovány prostřednictvím počítačových sítí. Jeho základním úkolem je v čase i prostoru svobodný a neomezený přístup ke vzdělávání.

Je třeba říci, že **není důležité, jak je pojem e-learning obecně definován**, protože obecná definice nevystihuje konkrétní edukační realitu. Každá vzdělávací instituce přizpůsobuje e-learning svým vlastním potřebám, upravuje jej a vyvíjí.

3.1.3 Historie e-learningu

Historii e-learningu lze rozdělit do několika etap. Jaké tyto etapy jsou popisuje tato kapitola.

3.1.3.1 Počátky e-learningu

Počátky e-learningu hledejme v technologii. I když je pravda, že technologie nijak neovlivňuje základní filozofii a princip vzdělávání, jsou to právě technologie, které umožnily nástup e-learningu.

Pojem e-learning je „poměrně“ nový. V druhé polovině šedesátých let se začalo experimentovat se stroji na učení. Byly to sálové počítače, které se využívaly i k výuce. Začalo se jim říkat vyučovací automaty. První výukové programy byly textové a pracovaly na principu testu, otázky a odpovědi [6].

3.1.3.2 Multimediální éra

Většího rozšíření bylo dosaženo v letech 1984 – 1993, v takzvané multimediální éře. V roce 1984 byla založena firma CBT Systems (CBT = computer-based training – vzdělávání založené na počítači). Založil ji v Kalifornii irský podnikatel Bill McCabe. Jeho myšlenka školit počítačové specialisty pomocí CBT byla velmi radikální. Velké počítačové firmy v té době poskytovaly školení zdarma spolu se softwarem, který spojovaly se svým hardwarem. Všechno se odehrávalo ve třídě. Nebyl žádný důvod k tomu, aby se za školení platilo. Právě když se zdálo, že se McCabe vrátí do Irska s nepořízenou, podařilo se mu najít firmu, která měla složitý software, žádný hardware a nedostatek lidí k tomu, aby uspokojila poptávku lidí, kteří se chtěli naučit software používat. Tato firma se stala prvním zákazníkem CBT Systems. Většina výukového softwaru byla zpracována v Irsku, kde byly v té době velmi nízké mzdy. Možnost školení bez nákladů na instruktory a místnosti lákala vedoucí pracovníky firem v počítačovém průmyslu, který podléhá periodickým ekonomickým výkyvům. Zanedlouho se CBT Systems stala předním poskytovatelem vzdělávání založeného na počítači. Obchodníci se softwarem potřebovali kvalifikované zákazníky v okamžiku, kdy dávali na trh nový produkt, a proto pracovníci CBT Systems měli možnost seznámit se s novými produkty dříve, než se objevily na trhu, a tím získávali jasnou konkurenční výhodu. Firma ovládla trh. Když se v polovině osmdesátých let stal CD-ROM novou vzdělávací pomůckou, převedla firma CBT Systems na toto médium všechny své vzdělávací programy a začala vydávat množství nových výukových materiálů. Koncem devadesátých let nabízela nejvíce titulů ze všech podniků na světě; 95 % z nich se zaměřovalo na informační technologii (IT). V této době se počítače začaly rozšiřovat do domácností a škol a ostatní institucí. Podniky měly o vzdělávání založené na CD velký zájem, protože CD byly ve srovnání se živými učiteli za babku. IT se najednou jevila jako nepostradatelná součást podnikání a zachování konkurenceschopnosti. Začaly se rozšiřovat počítače Macintosh, nebo osobní počítače s operačním systémem Windows 3.1 obsahující CD-ROM a také například program Powerpoint sloužící k prezentacím. Tyto technologie přispěly k širšímu používání e-learningu. Důležitou úlohu v této době hrály CD-ROM, které umožňovaly přenášení výukového programu kdykoliv a kamkoliv.

Na sklonku devadesátých let se začalo mluvit o tom, že kurzy založené na používání CD nesplňují očekávání, a jejich prodej prudce poklesl. Lidé se většinou nechtěli učit sami pomocí levné náhražky za instruktora ve třídě. Jakmile byli v koncích nebo udělali chybu, neměli na koho se

obrátit. Postrádali spolužáky, s nimiž by se poradili. Na seminářích, které navštěvovali dříve, je nikdo nevyrušoval tak, jako u psacího stolu v zaměstnání nebo doma [14].

3.1.3.3 První vlna e-learningu

Dalším důležitým milníkem je období 1994-1999, kdy se začal rozšiřovat internet. Toto období můžeme nazvat první vlnou e-learningu. V této době začal e-learning využívat Webu, emailu, mediálních přehrávačů. Začala se objevovat výuka pomocí emailu, intranet CBT(Computer-Based Training = výuka pomocí počítačů) s textem a jednoduchou grafikou a WBT(Web-Based Training = výuka pomocí počítače připojeného do sítě).

V roce 1998, kdy se na modelech CD projevily první trhliny, se stal novým prezidentem a CEO firmy CBT Systems Greg Priest. Ten pochopil, že Internet nahradí CD. Firma pod jeho vedením vytvořila projekt pro firmu UNISYS a pomohla jí založit UNISYS University, která dodávala obsah prostřednictvím Internetu, poskytovala však také personalizovaný vzdělávací portál, trasovací systémy, onlinové zpravodaje, diskusní skupiny a jiné věci představitelné v té době. Bylo to o osm měsíců dříve než firma IBM zavedla Learning on Demand (učení na požádání). Greg Priest předpokládal, že se nakonec všichni přikloní k této formě vzdělávání. Najal odborníka na marketing, který přizval ke spolupráci Jaye Crosse (autora článku) jako odborníka na e-learning, protože jeho jméno bylo v internetových vyhledávačích v souvislosti s e-learningem nejčastěji citováno. Zabýval se vzdělávacími programy pro počítače od konce sedmdesátých let a přes CD došel až k využití Internetu pro účely vzdělávání. Produkty, které navrhoval, přebíraly různé firmy, např. Bank of America, NASA, IBM, Atari aj. Začala se objevovat výuka pomocí emailu, intranet CBT(Computer-Based Training = výuka pomocí počítačů) s textem a jednoduchou grafikou a WBT(Web-Based Training = výuka pomocí počítače připojeného do sítě).

Začátkem roku 1999 měla firma CBT Systems kolem 250 zaměstnanců. Tehdy se vedení firmy rozhodlo změnit její název a s ním i image. V říjnu 1999 bylo oznámeno, že firma se bude nazývat SmartForce, The eLearning Company.

3.1.3.4 Druhá vlna e-learningu

Rozvíjením technologií jako jsou JAVA/IP síťové aplikace, dále také rozšíření Webových aplikací, streaming videa a zvuku umožnilo posunout hranici e-learningu ještě výše. V dnešní době můžeme spojit elektronickou výuku s výukou s lektorem v reálném čase a to vše prostřednictvím internetu. Toto období od roku 2000 bývá označováno jako druhá vlna e-learningu.

3.1.4 E-learning: co, proč, jak

E-learning je definovaný jako výuka zprostředkovaná počítačem pomocí CD-ROM, internetu nebo intranetu s následujícími vlastnostmi:

- Zahrnuje obsah týkající se učebních cílů.

- Používá instrukční metody jako příklady a cvičení, sloužící k pochopení látky.
- Používá mediální prvky jako slova a obrázky k dodání obsahu.
- Ke zvýšení organizačních vlastností staví nové znalosti a dovednosti provázaně s individuálními učebními cíli.

Nyní si popíšeme, co rozumíme pod pojmy co, proč a jak a co se k nim vztahuje.

Pod slovem *co* se skrývá, co obsahují kurzy e-learningu. Ty obsahují jak obsahovou část (to jsou informace) tak i instrukční metody (to jsou techniky) jak pomoci lidem se učit

Pod slovíčkem *jak*, je pospáno jak je e-learning poskytován. E-learning je prezentován pomocí počítače, a to pomocí slov jak v tištěné tak i v mluvené formě, také pomocí obrázků ať už jako fotografií, ilustrací, animací či videa.

Proč nám říká proč vlastně využíváme e-learningu. E-learningové jsou tvořeny tak, aby pomohly studentovi dosáhnout osobního učebního cíle nebo splnily svoji práci způsobem, který zvyšuje spodní hranici cílů organizace.

Když si rozebereme slovo e-learning, zjistíme, že „e“ značí to „jak“, což je způsob, jakým jsou kurzy prováděny, to znamená elektronickým způsobem. „learning“ zase značí „co“ a „proč“, to znamená, že se jedná o způsob učení [7].

3.1.5 Distribuce

E-learning je šířen pomocí CD-ROM, internetu, intranetu, nebo i pomocí dalších prostředků (telefon, atd.) nebo kombinováním těchto prostředků. V dnešní době je nejrozšířenější způsob poskytování e-learningu internet, intranet a také se používají přenosová média jako DVD-ROM.

Každý z těchto způsobů má nějaké výhody a nevýhody. Například pokud využíváme e-learning šířený pomocí internetu, je nutné mít možnost nejlépe trvalého připojení. Tato nutnost odpadá v případě využití CD-ROM či DVD-ROM. V těchto případech máme kompletní aplikaci na médiu a v počítači. Je zde ale problém týkající se aktualizací, kdy v případě změn je nutno si pořídit médium s aktualizací a aplikaci přeinstalovat. Kombinací těchto možností je, že je aplikace dodána na CD-ROM a z internetu jsou stahovány aktualizace. Tento přístup je výhodný v tom, že i přestože musíme mít k připojení k internetu, nemusí být tak rychlé a trvalé jako v případě, že kurz probíhá čistě na internetu. Výhodami WBT (Web-Based Training = výuka pomocí počítače připojeného do sítě) jsou snadná aktualizace, komunikace a distribuce. Jako hlavní výhoda CBT (Computer-Based Training = výuka pomocí počítačů) je, že nemusíme mít připojení k internetu, protože i v dnešní době ne každý připojení k internetu má. V dnešní době, kdy přenosová média mají kapacitu i 8,5 GB, je možné takto přenášet velké množství informací(video ve vysokém rozlišení), což přes internet není až tak možné.

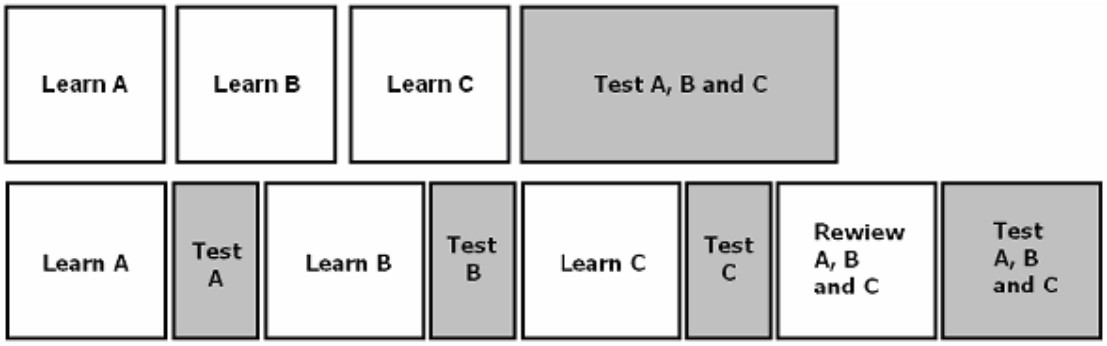
3.2 Obecná struktura elektronických dokumentů

Poté co jsme si vysvětlili základní pojmy, dostáváme se do části, ve které si popíšeme, jak by měla vypadat struktura elektronických dokumentů.

V této kapitole budu vycházet z knihy Williama Hortona: Designing Web-Based Training : How to Teach Anyone Anything Anywhere Anytime. V této knize je názorně popsáno jak vytvářet elektronické kurzy, jaká má být jejich struktura a co vše mají popisovat a obsahovat.. Tudíž mnou navrhnutá struktura elektronické aplikace se opírá o poznatky získané z této knihy.

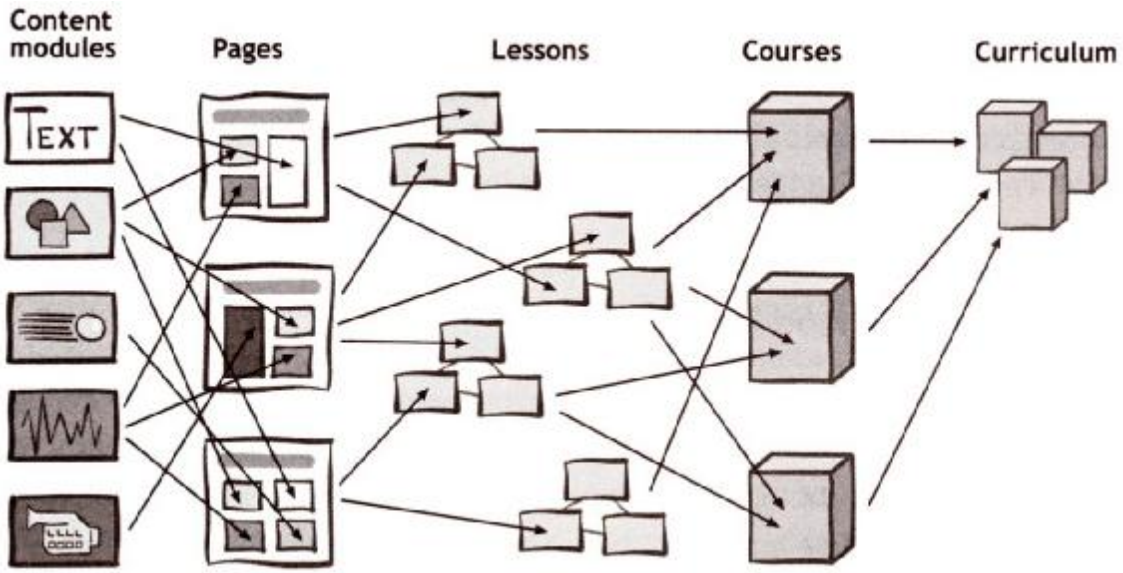
Struktura dokumentu odpovídá v tomto projektu struktuře dokumentu WBT (Web-Based Training) kurzu. Pojem WBT jsme si již vysvětlili v kapitole 3.1.

Elektronický kurz se skládá z několika lekcí. V každé lekci se nachází části, které se zabývají seznámením s problematikou a ukázkou příkladů, otestováním právě naučené látky a zobrazením výsledků testů např. v podobě grafu. Obtížnost látky by měla postupně narůstat s další lekcí. Informační a testové části lekce, by se měly střídat v kratších intervalech, aby si uživatel novou problematiku hned odzkoušel viz obr. 18. Obrázek je převzat z [7].



Obr. 18. Znázornění průběhu lekcí

Textová informace by měla být doplněna obrazem, videem a zvukem. Toto je výhoda elektronického kurzu oproti jiným výukovým prostředkům. To je ukázáno v následujícím obr. 19. (obrázek je převzat z [7]), který zobrazuje propojení mezi kurzy, lekcemi, stránkami lekcí a jejich obsahem [7]. Toto vše je zahrnuto v celkovém studijním plánu. Vzhled a ovládací prvky elektronického kurzu či dokumentu by měly být navrženy co nejpřehledněji.

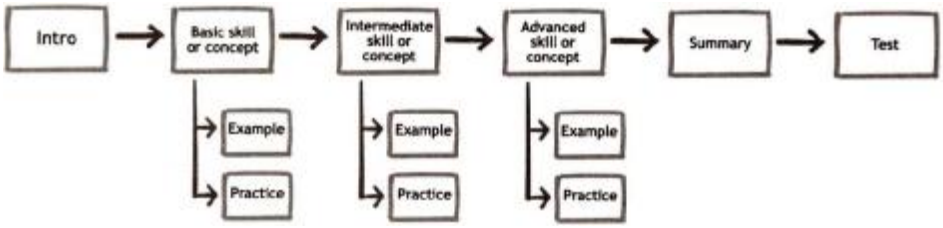


Obr. 19. Struktura e-learningového kurzu

V tomto projektu se nebudeme zabývat celým studijním plánem, ale pouze jedním kurzem, či spíše jednou lekcí.

Poznatky o tvorbě elektronických kurzů či dokumentů můžeme shrnout do několika bodů.

- Lekce se skládají z posloupností logicky spojených webových stránek, které společně vytvářejí bohaté učební zážitky.
- Klasická vyučovací struktura je nejběžnější, nejbezpečnější a nejméně účinná z běžných vyučovacích struktur. Viz obr. 20. Obrázek je převzat z [7].



Obr. 20. Klasická vyučovací struktura

Ostatní struktury učebních metod vyrobené na objednávku splňují potřeby žáků.

- Určité komponenty jsou běžné v mnoha typech lekcí. Ty zahrnují následující stránky: přivítání, související zdroje, úvod, shrnutí, událost přehrání nahraného záznamu, zpětná vazba lekcí, rozvětvení, procedury, a uvádění příkladů.
- Pří organizování lekcí a kurzů, analyzujte závislosti mezi pojmy a žákovou aktuální úrovní znalostí.
- Usilujte o kompletní samostatné moduly, které se mohou kombinovat pro učení komplexních předmětů.
- Namísto sekvenčních struktur uvažujte o vrstvených strukturách.

3.3 Revize dokumentů pro výuku OCL

V rámci pokračování diplomové práce bylo zapotřebí provést revizi dokumentů pro výuku OCL. Tato revize spočívala v podrobném překontrolování ročníkového projektu, zda obsahuje všechny údaje a zda poskytuje kompletní informace o jazyku OCL. Jelikož je tato problematika velmi rozsáhlá, kontrola trvala značnou dobu a musely být provedeny úpravy jelikož byly objeveny i nějaké nedostatky. Většina nedostatků byla odhalena a upravena během práce na semestrálním projektu, avšak až při kompletaci dat v rámci diplomové práce byly doplněny všechny důležité části. Po provedení revize jsou data kompletní a aplikace pokrývá kompletně celou problematiku specifikace omezení objektů systému.

4 Požadavky na programový systém pro podporu e-learningu

Úkolem bylo vytvořit systém pro podporu e-learnigu, který by umožňoval prostudování jazyku pro specifikaci omezení objektů systému (OCL). Systém musí také umožňovat prověřování znalostí a sledování pokroku v pochopení prezentované látky.

4.1 Funkční požadavky

Systém byl navržen tak, aby byl přehledný a nezatěžoval uživatele složitým ovládáním. Dalším kritériem pro vytvoření systému bylo, aby byl pokud možno univerzální a bylo ho možno spustit na jakémkoliv počítači kdekoliv na světě. Proto při vývoji bylo přihlédnuto i k těmto kritériím a systém byl vytvořen tak, aby tato kritéria splňoval (viz kapitola 5.7).

4.2 Jazykové požadavky

Jelikož se systém zabývá specifikací omezení objektů systému, což je velmi specifická látka byla jako hlavní jazyková verze zvolena angličtina, aby nedocházelo ke špatné interpretaci. Tím byla zajištěna i možnost využití tohoto systému studentům studujícím v anglickém jazyce. Tito studenti studují na VUT v rámci programu **ERASMUS.** Dále byla vytvořena i česká verze systému, která ovšem neobsahuje kompletní specifikaci.

4.2.1 Program Socrates / Erasmus

Podprogram ERASMUS je součástí programu Evropské unie SOCRATES. Vychází ze snahy Evropské unie o prohloubení spolupráce mezi školami v rámci členských zemí, od roku 1997 zemí přidružených k EU. Cílem je posílit spolupráci vysokých škol evropského regionu při realizaci vzdělávací politiky v jednotlivých zemích a postupně sbližovat vzdělávací systémy v zemích Evropy. V současné době probíhá druhá etapa tohoto programu, která zahrnuje období 2001 až 2006 [17].

4.3 Požadavky na prověřování znalostí

Aby systém umožňoval prověřování znalostí a sledování pokroku v pochopení prezentované látky byla do systému implementována testová část, která zajišťuje právě tento požadavek.

5 Prostředky pro vývoj www aplikací

V této kapitole si popíšeme jednotlivé prostředky sloužící k vývoji www aplikací. Seznámíme se ze základními pojmy jakou jsou www či http, také si povíme něco o webových serverech. Obeznámíme se s pojmy PHP, CSS, MySQL. Také si povíme jaké existují editory pro tvorbu html stránek a jaké jsou nejrozšířenější prohlížeče.

5.1 WWW, URL a http

V této kapitole si postupně popíšeme pojmy WWW, URL a http. Zjistíme, co se skrývá pod těmito zkratkami a k čemu se tyto technologie používají. U některých si povíme něco o vývoji a historii. V první podkapitole se budu věnovat WWW.

5.1.1 Co se skrývá pod zkratkou WWW.

World Wide Web (WWW, také pouze zkráceně web), ve volném překladu „Celosvětová pavučina“, je označení pro aplikace internetového protokolu HTTP. Je tím myšlena soustava propojených hypertextových dokumentů.

V češtině se slovo web často používá nejen pro označení celosvětové sítě dokumentů, ale také pro označení jednotlivé soustavy dokumentů dostupných na tomtéž webovém serveru nebo na téže internetové doméně nejnižšího stupně (internetové stránce).

Dokumenty umístěné na počítačových serverech jsou adresovány pomocí URL, jehož součástí je i doména a jméno počítače. Název naprosté většiny těchto serverů začíná zkratkou www, i když je možné používat libovolné jméno vyhovující pravidlům URL.

Protokol HTTP je dnes již používán i pro přenos jiných dokumentů než jen souborů ve tvaru HTML a výraz World Wide Web se postupně stává pro laickou veřejnost synonymem pro internetové aplikace [22].

5.1.1.1 Webový server

Jak jsme si řekli v předchozí kapitole, zkratka www označuje dokumenty na webovém serveru. Abychom mohli provozovat www stránky, musíme mít na počítači nějaký webový server nainstalovaný. Než se ale pro nějaký rozhodneme, povíme si, co to vlastně webový server je. Termín **Webový server** může znamenat:

- Počítač, který je odpovědný za vyřizování požadavků HTTP od klientů - programů zvaných webový prohlížeč. Vyřízením požadavků se rozumí odeslání webové stránky. Webové stránky jsou obvykle dokumenty HTML.
- Počítačový program, který provádí činnosti popsané v předchozím bodě.

Zdroje informací webového serveru jsou následující. Webový server má v zásadě dvě možnosti, jak získávat informace, které vrací klientům:

- Jsou to buď předem připravené datové soubory (HTML stránky), které webový server bez změny poskytne klientovi (tzv. statický obsah).
- Teprve na základě požadavku klienta jsou data shromážděna (přečtena ze souboru, databáze, nebo nějakého koncového zařízení), zformátována a připravena k prezentaci ve formátu HTML a poskytnuta webovému prohlížeči (tzv. dynamický obsah).

K dynamickému vytváření obsahu se používá celá řada různých technologií (PERL, PHP, ASP, ASP.NET, JSP apod.). Statický obsah je schopen server poskytnout významně rychleji než dynamický. Na druhé straně pomocí dynamického obsahu lze poskytovat mnohem větší obsah informací a lze reagovat i na různé „ad hoc“ dotazy klientů. Proto se v praxi v mnoha případech oba způsoby poskytování obsahu kombinují - například pomocí cachování [5].

Apache HTTP Server

Apache HTTP Server je softwarový webový server s otevřeným kódem pro Linux, BSD, Microsoft Windows a další platformy. V současné době dodává prohlížečům na celém světě většinu internetových stránek. Dle tvrzení svých tvůrců [1] je Apache HTTP Server nejpopulárnější server na internetu. V květnu 1999 běžel na 57 % všech serverů a v listopadu 2005 jeho používanost dosáhla 69 % [6].

5.1.2 URL a http

V této části si popíšeme pojmy URL a http. Povíme si, k čemu slouží a kde se používají.

5.1.2.1 URL

URL je zkratka anglického výrazu Uniform Resource Locator. Je to řetězec znaků s definovanou strukturou a slouží k přesné specifikaci umístění zdrojů informací (ve smyslu dokument nebo služba) na Internetu [22].

URL definuje doménovou adresu serveru, umístění zdroje na serveru a protokol, kterým je možné zdroj zpřístupnit.

Jednotlivá pole v URL:

protokol, doménové jméno, port, specifikace souboru, parametry

Některá pole jsou nepovinná – buď nemají význam, nebo se předpokládá předdefinovaná hodnota, závislá např. na protokolu (např. pro HTTP je implicitní port 80), nebo na aplikaci (pro webový prohlížeč HTTP).

Příklad pro www stránku:

http://www.stud.fit.vutbr.cz:80/~xdraho04/DPI/index.php

protokol: http

server: stud.fit.vutbr.cz

port: 80 – jelikož pro http je port 80 implicitní, není ho třeba v tomto konkrétním případě uvádět

specifikace souboru: /~xdraho04/DPI/index.php – je uveden včetně cesty (adresáře) v rámci serveru

5.1.2.2 HTTP

HTTP (Hyper Text Transfer Protocol) je internetový protokol určený původně pro výměnu hypertextových dokumentů ve formátu HTML. Používá obvykle port TCP/80, verze 1.1 protokolu je definována v RFC 2616. Tento protokol je spolu s elektronickou poštou tím nejvíce používaným a zasloužil se o obrovský rozmach internetu v posledních letech.

V současné době je používán i pro přenos dalších informací. Pomocí rozšíření MIME umí přenášet jakýkoli soubor (podobně jako e-mail), používá se společně s formátem XML pro tzv. webové služby (spouštění vzdálených aplikací) a pomocí aplikačních bran zpřístupňuje i další protokoly, jako je např. FTP nebo SMTP.

HTTP používá jako některé další aplikace tzv. jednotný lokátor prostředků (URL, Uniform Resource Locator), který specifikuje jednoznačné umístění nějakého zdroje v Internetu.

K protokolu HTTP existuje také jeho bezpečnější verze HTTPS, která umožňuje přenášená data šifrovat a tím chránit před odposlechem či jiným narušením.

Činnost protokolu

Protokol funguje způsobem dotaz-odpověď. Uživatel (pomocí programu, obvykle internetového prohlížeče) pošle serveru dotaz ve formě čistého textu, obsahujícího označení požadovaného dokumentu, informace o schopnostech prohlížeče apod. Server poté odpoví pomocí několika řádků textu popisujících výsledek dotazu (zda se dokument podařilo najít, jakého typu dokument je atd.), za kterými následují data samotného požadovaného dokumentu.

Pokud uživatel bude mít po chvíli další dotaz na stejný server (např. proto, že uživatel v dokumentu kliknul na hypertextový odkaz), bude se jednat o další, nezávislý dotaz a odpověď. Z hlediska serveru nelze poznat, jestli tento druhý dotaz jakkoli souvisí s předchozím. Kvůli této vlastnosti se protokolu HTTP říká bezestavový protokol – protokol neumí uchovávat stav komunikace, dotazy spolu nemají souvislost. Tato vlastnost je nepříjemná pro implementaci složitějších procesů přes HTTP (např. internetový obchod potřebuje uchovávat informaci o identitě zákazníka, o obsahu jeho „nákupního košíku“ apod.). K tomuto účelu byl protokol HTTP rozšířen o tzv. HTTP cookies, které umožňují serveru uchovávat si informace o stavu spojení na počítači uživatele.

5.2 HTML

HTML je zkratka z anglického HyperText Markup Language - značkovací jazyk pro hypertext. Je jedním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci stránek na Internetu.

Jazyk je podmnožinou dříve vyvinutého rozsáhlého univerzálního značkovacího jazyka SGML (Standard Generalized Markup Language). Vývoj HTML byl ovlivněn vývojem webových prohlížečů, které zpětně ovlivňovaly definici jazyka [22].

5.2.1 Vývoj a historie

Jazyk HTML byl vyvinut v roce 1990 jako jazyk pro vytváření www dokumentů. V roce 1991 byla vydána verze 0.9. Po roce 1993 kdy byl vyvinut prohlížeč Mosaic pro počítače PC a Macintosh, došlo k velkému rozvoji webu a bylo nutné definovat pro HTML nové standardy. V roce 1994 byla vydána verze 2.0. Aby nedocházelo k tomu, že různé firmy nabízely různé upravené verze jazyka HTML, které byly nekompatibilní s verzemi od ostatních výrobců, bylo založeno **World Wide Web Consortium**. Konsorcium sjednotilo verze od různých výrobců a dohodlo se s nimi na základních principech a komponentách nových standardů. Konsorcium bylo založeno v roce 1994 a další verze HTML, tentokrát verze 3.2, již byla vydána tímto konsorciem. Následovala verze 4.0 a nejnovější verzí je verze 4.01, která byla vydána v roce 1999 a dále se již nevyvíjí, protože má být nahrazena novějším XHTML (Extended HTML) . Mezinárodní konsorcium W3C standardizuje jednotlivé verze HTML, XHTML a jejich varianty Transitional, Strict a Frameset. Rozdílů mezi HTML a XHTML je hned několik [23]:

Některé věci platily už v HTML, XHTML je striktně vyžaduje:

- Všechny atributy mají hodnoty v uvozovkách.
- Zákaz křížení tagů.

Jaké jsou rozdíly XHML oproti HTML:

- Tagy a atributy jsou malými písmeny.
- Nepárové tagy končí lomítkem.
- Párové tagy jsou párové povinně.
- Všechny atributy musejí mít hodnotu.
- Interní javascript a styly se zapisují jiným způsobem.
- Dokument má mít XML prolog.
- Dokument požaduje správný doctype.

5.2.1.1 Současné verze HTML a XHTML

Mezi nejčastější verze patří:

- HTML 4.01 (4.0) přechodové (transitional) / striktní (strict)
- XHTML 1.0 přechodové (transitional) / striktní (strict)
- XHTML 1.1

Jakou verzi HTML dokument využívá, by mělo být uvedeno v hlavičce formou konstrukce <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" >

5.2.2 Popis jazyka

Jazyk **HTML** je charakterizován množinou značek a jejich atributů pro danou verzi definovaných. Mezi značky se uzavírají části textu dokumentu, a tím se určuje význam (*sémantika*) obsaženého textu. Názvy jednotlivých značek se uzavírají mezi úhlové závorky ("<" a „>“). Část dokumentu uzavřená mezi značkami tvoří tzv. **element** (prvek) dokumentu. Součástí obsahu elementu mohou být další vnořené elementy. Atributy jsou doplňující informace, které upřesňují vlastnosti elementu [10].

Pro každou verzi existuje definice pravidel **DTD** (*Document Type Definition*). Od verze 4.01 musí být odkaz na deklaraci DTD v dokumentu uveden pomocí klíčového slova DOCTYPE. DTD definuje pro určitou verzi, které elementy je možné používat a s jakými atributy.

Dokument může mimo značkování obsahovat další prvky:

- Direktivy - začínají znaky „<!“, jsou určeny pro zpracovatele dokumentu (prohlížeč).
- Komentáře - pomocné texty pro programátora nejsou součástí obsahu dokumentu a nezobrazují se (prohlížeč je ignoruje). Příklad komentáře je níže.
- Kód skriptovacích jazyků.
- Definice událostí a kód pro jejich obsluhu.

5.2.2.1 Struktura dokumentu

Dokument v jazyku **HTML** má předepsanou strukturu:

- Deklarace DTD - je povinná až ve verzi 4.01, je uvedena direktivou <!DOCTYPE.
- Kořenový element - element html (značky <html> a </html>) reprezentuje celý dokument. Je nepovinný, ale je doporučeno ho používat.
- Hlavička elementu - jsou to metadata, která se vztahují k celému dokumentu. Definují např. název dokumentu, jazyk, kódování, klíčová slova, popis, použitý styl zobrazení. Hlavička je uzavřena mezi značky <head> a </head>.
- Tělo dokumentu - obsahuje vlastní text dokumentu. Vymezuje se značkami <body> a </body>.

5.2.2.2 Příklad zdrojového kódu

Příklad **HTML** pro verzi 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<!-- toto je komentář -->
  <head>
    <title>Titulek stránky</title>
  </head>
<!-- tělo dokumentu -->
  <body>
    <h1>Nadpis stránky</h1>
    <p>Toto je tělo dokumentu</p>
  </body>
</html>
```

5.3 CSS

CSS je zkratka pro anglický název *Cascading Style Sheets*, česky **tabulky kaskádových stylů**. Je to jazyk pro popis způsobu zobrazení stránek napsaných v jazycích HTML, XHTML nebo XML.

Jazyk byl navržen standardizační organizací W3C. Byly vydány zatím dvě verze specifikace CSS1 a CSS2 (plus CSS 2.1), pracuje se na verzi CSS3.

Hlavním smyslem je umožnit návrhářům oddělit vzhled dokumentu od jeho struktury a obsahu. Původně to měl umožnit už jazyk HTML, ale v důsledku nedostatečných standardů a konkurenčního boje výrobců prohlížečů se vyvinul jinak. Starší verze HTML obsahují celou řadu elementů, které nepopisují obsah a strukturu dokumentu, ale i způsob jeho zobrazení. Z hlediska zpracování dokumentů a vyhledávání informací není takový vývoj žádoucí.

5.3.1 Výhody a nevýhody používání CSS

Nejprve si popíšeme výhody, které nám přináší používání CSS. Nezapomenu ovšem ani na nevýhody.

5.3.1.1 Výhody

Používání kaskádových stylů ve srovnání se samotným HTML v praxi přináší výhody:

- Rozsáhlejší možnosti - CSS nabízí rozsáhlejší formátovací možnosti než samotné HTML. Např. pro formátování bloku textu – tj. určení vzdálenosti od jejich elementu či okraje stránky nenabízí HTML nic. CSS má vlastnosti padding a margin. V HTML by bylo potřeba vytvořit složitou konstrukci vnořených tabulek.
- Konzistentní styl - Na všech stránkách webové prezentace by měly být všechny nadpisy stejné úrovně, seznamy, zdůrazněné části textu apod. stejného stylu. S použitím

formátovacích možností HTML je to obtížné – u každého objektu v každém dokumentu se vzhled objektu stále znovu nastavuje. S použitím CSS je to velmi jednoduché. Vytvoří se soubor stylu, který se připojuje k HTML dokumentu. Ve všech dokumentech jsou pak objekty stejného vzhledu.

- Oddělení struktury a stylu.
- Dynamická práce se styly - Provést změnu stylu webu, který pro formátování vzhledu využívá jen možnosti HTML, znamená najít a nahradit všechny značky a změnit atributy mnoha dalších značek. V případě používání CSS znamená změna stylu webu přepsání jediného souboru – souboru stylů.
- Formátování XML dokumentů.
- Větší kompatibilita alternativních webových prohlížečů.
- Kratší doba načítání stránky.

Výhodou CSS oproti starému formátování v HTML je, že kód a obsah webu je uložen v souboru .html a veškerý design a formátování se načítá z jednoho souboru .css, který je většinou společný pro celý web. To znamená, že pokud máte v plánu změnu designu webu, stačí změnit pouze jeden soubor .css a změna se aplikuje na celý web. Také se soubor CSS uloží do mezipaměti prohlížeče a pokud není změněn, tak se načítá pouze jednou a zobrazení stránek se velmi urychlí.

Mohou také existovat různé styly pro různá výstupní zařízení. Webdesigner má tak možnost prostřednictvím CSS stylů dokumentu určit, jak bude vypadat na papíře, při projekci či na PDA apod. Specifikace CSS nezapomínají dokonce ani na zrakově postižené - je možno napsat styly pro hlasový syntetizátor nebo hmatovou čtečku Braillova písma.

Je také možnost upravit formátování podle prohlížeče, kterým si uživatel danou stránku zobrazuje. Jednoduše si vytvoříte více souborů .css (např. styl1.css a styl2.css) a podle prohlížeče, který si o stránku požádá, připojíte jiný soubor. Tím se dá eliminovat problém různé interpretace kódu jednotlivými prohlížeči.

5.3.1.2 Nevýhody

Hlavní nevýhodou CSS je zatím stále špatná podpora v majoritních prohlížečích. Různé prohlížeče interpretují stejný CSS kód jinak a je někdy velmi obtížné jej napsat tak, aby se na všech (resp. na několika vybraných) prohlížečích výsledek zobrazil stejně. Situace se ale v roce 2006 značně zlepšuje, v souvislosti s tím se s napětím očekával příchod Internet Exploreru 7, který by měl postupně vytlačit svého předchůdce IE 6, který byl častým zdrojem problémů. Nicméně ani IE 7 se striktně nedrží definice CSS 2.1.

5.4 PHP

Tato kapitola se věnuje PHP. Nejprve nám objasní, co to vlastně PHP a jaké existují verze. Také se seznámíme s historii PHP a nástrojem phpMyAdmin.

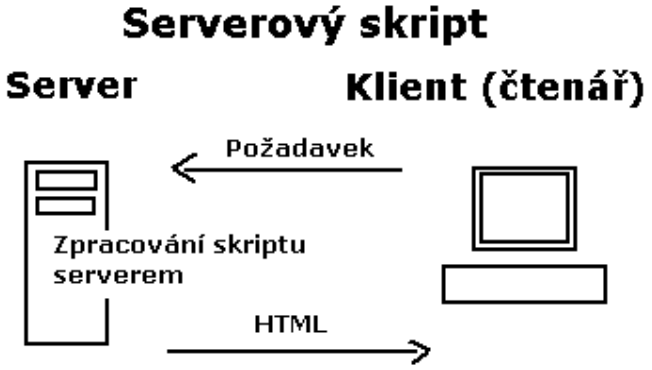
5.4.1 Co to je PHP

PHP (rekurzivní zkratka *PHP: Hypertext Preprocessor*, „PHP: Hypertextový preprocesor“, původně *Personal Home Page*) je skriptovací programovací jazyk, určený především pro programování dynamických internetových stránek. Nejčastěji se začleňuje přímo do struktury jazyka HTML, XHTML či WML, což je velmi výhodné pro tvorbu webových aplikací. PHP lze ovšem také použít i k tvorbě konzolových a desktopových aplikací [7].

5.4.2 Jak PHP pracuje a jaké máme verze

PHP skripty jsou prováděny na straně serveru, k uživateli je přenášen až výsledek jejich činnosti. Syntaxe jazyka kombinuje hned několik programovacích jazyků (Perl, C, Pascal a Java). PHP je nezávislý na platformě, skripty fungují bez úprav na mnoha různých operačních systémech. Obsahuje rozsáhlé knihovny funkcí pro zpracování textu, grafiky, práci se soubory, přístup k většině databázových serverů (mj. MySQL, ODBC, Oracle, PostgreSQL, MSSQL), podporu celé řady internetových protokolů (HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP, …)

PHP se stalo velmi oblíbeným především díky jednoduchosti použití a tomu, že kombinuje vlastnosti více programovacích jazyků a nechává tak vývojáři částečnou svobodu v syntaxi. V kombinaci s databázovým serverem (především s MySQL nebo PostgreSQL) a webovým serverem Apache je často využíván k tvorbě webových aplikací. Díky velmi častému nasazení na serverech se vžila zkratka LAMP – tedy spojení Linux, Apache, MySQL a PHP nebo Perl.



Obr. 25. Serverový skript

5.4.2.1 Historie a současnost

Od roku 1994 je PHP jedním z nejpoužívanějších způsobů tvorby dynamicky generovaných WWW stránek. Jeho tvůrce (Rasmus Lerdorf) jej vytvořil pro svou osobní potřebu přepsáním z Perlu do jazyka C. Sada skriptů byla vydána ještě v témž roce pod názvem *Personal Home Page Tools*, zkráceně PHP.

V polovině roku se systém PHP spojil s programem *Form Interpreter* stejného autora. Tak vzniklo PHP/FI 2.0. Zeev Suraski a Andi Gutmans v roce 1997 přepsali parser a zformovali tak základ PHP3. Současně byl název změněn na dnešní podobu *PHP hypertext procesor*. PHP3 vyšlo v roce 1998, bylo rychlejší, obsahovalo více funkcí. Také běželo i pod operačním systémem Windows.

V roce 2000 vychází PHP verze 4, o čtyři roky později pak verze 5 s vylepšeným objektovým přístupem, podobným jazyku Java. Zatím poslední verzí je 5.2.1 z roku 2007 [22].

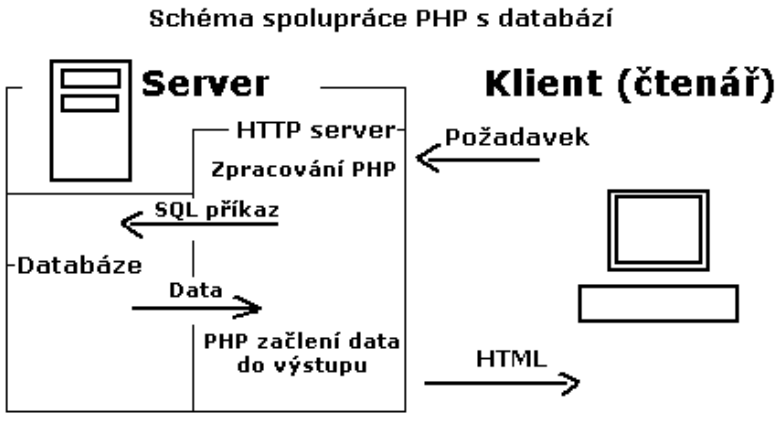
5.4.2.2 phpMyAdmin

phpMyAdmin je nástroj napsaný v jazyce PHP umožňující jednoduchou správu obsahu databáze MySQL prostřednictvím webového rozhraní. V současné době umožňuje vytvářet/rušit databáze, vytvářet/upravovat/rušit tabulky, provádět SQL příkazy a spravovat klíče. Jedná se o jeden z nejpopulárnějších nástrojů pro správu databáze. Je k dispozici v 52 jazycích.

5.5 Databáze

Databáze je určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento systém se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením *databáze* – v závislosti na kontextu – myslí jak uložená data, tak i software (SŘBD).

Jak pracuje webový server s podporou PHP a databáze SQL, je ukázáno na následujícím obrázku.



Obr. 26. Schéma spolupráce PHP s databází

Pro práci s databází existuje několik databázových systémů. Patří mezi ně Oracle, MS SQL, My SQL a Postgre SQL. Každý z těchto databázových systémů má své výhody a nevýhody. Výběr záleží na uživateli, na prostředí, ve kterém pracuje, i na konkrétní aplikaci. Pro svou aplikaci jsem zvolil databázový systém MySQL, a proto ze nadále budu věnovat jen tomuto systému, a trochu blíže ho popíši.

5.5.1 MySQL

MySQL je databázový systém, vytvořený švédskou firmou MySQL AB. Jeho hlavními autory jsou Michael „Monty“ Widenius a David Axmark. Je považován za úspěšného průkopníka dvojího licencování – je k dispozici jak pod bezplatnou licencí GPL, tak pod komerční placenou licencí.

MySQL je multiplatformní databáze. Komunikace s ní probíhá – jak už název napovídá – pomocí jazyka SQL. Podobně jako u ostatních SQL databází se jedná o dialekt tohoto jazyka s některými rozšířeními.

Pro svou snadnou implementovatelnost (lze jej instalovat na Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích. Velmi oblíbená a často nasazovaná je kombinace MySQL, PHP a Apache jako základní software webového serveru.

MySQL bylo od počátku optimalizováno především na rychlost, a to i za cenu některých zjednodušení: má jen jednoduché způsoby zálohování, a až donedávna nepodporovalo pohledy, triggery, a uložené procedury. Tyto vlastnosti jsou doplňovány teprve v posledních letech, kdy začaly nejčastějším uživatelům produktu – programátorům webových stránek – již poněkud scházet [22].
Přehled podporovaných vlastností:

- cizí klíče (od verze 3.23 podporovány v tabulkách typu InnoDB)
- transakce (od verze 3.23 podporovány v tabulkách typu InnoDB)
- podpora různých znakových sad a časových pásem v datech (od verze 4.1)
- poddotazy (od verze 4.1)
- uložené procedury (od verze 5.0)
- triggery (od verze 5.0)
- pohledy (od verze 5.0)

5.6 Editory a prohlížeče

Nezbytným prostředkem pro vytváření www aplikací jsou editory a prohlížeče. Existuje několik typů editorů, o kterých si povíme v kapitole 5.6.1. V kapitole 5.6.2 se zase věnuji prohlížečům webových stránek.

5.6.1 Editory

HTML dokument je obyčejný textový dokument, obsahující kromě vlastního textu značky, které textu přiřazují určitý význam. K psaní WWW-stránek tedy není třeba žádný speciální software, může postačit obyčejný textový editor, např. Poznámkový blok (Notepad). Samozřejmě ale pro vytváření webových stránek existuje nepřeberné množství editorů. V zásadě je můžeme rozdělit na strukturní a WYSIWYG editory.

5.6.1.1 Strukturní editory

Strukturní editory jsou takové editory, ve kterých pracujete přímo se zdrojovým kódem dokumentu, je proto nutné znát **HTML**. Průběžný vzhled si zde samozřejmě můžete kdykoli zkontrolovat jednoduchým přepnutím do prohlížeče. Jsou to např. PsPad, HTML-Kit, Vim, Emacs, 1stPage, Golden HTML Editor nebo HomeSite. Já osobně používám Context nebo PhpEditor. Pro uživatele Linuxu pak například Quanta. Tyto editory práci usnadňují jednak vizuálním odlišením vlastního textu od HTML kódu, jednak tím, že umožňují jednotlivé tagy snadno zadávat, a množstvím dalších práci zlehčujících funkcí.

5.6.1.2 WYSIWYG editory

Ve WYSIWYG editoru již nepracujete s kódem, ale se vzhledem stránky. Práce s nimi je možná pohodlnější, ale výsledný kód bývá obvykle poněkud zmatečný. Troufám si tvrdit, že vytvořit validní HTML dokument tyto editory schopné nejsou. Jistě ale mohou být užitečnou pomůckou.

5.6.2 Prohlížeče

Abyste si mohli stránky zobrazit, potřebujete internetový prohlížeč. Je nutné si uvědomit, že každý prohlížeč bohužel interpretuje kód trochu jinak a tudíž se v různých prohlížečích může www stránka zobrazit odlišně. Nejsprávněji podle standardů stránku zobrazí Mozilla. Nejrozšířenějším prohlížečem je Internet Explorer (cca 90% uživatelů), proto je více než vhodné si v něm stránky alespoň zkontrolovat a pokusit se odstranit alespoň základní nedostatky. Dalšími rozšířenějšími prohlížeči jsou Opera a Netscape [23].

5.7 Výběr a zhodnocení prostředků pro vývoj www aplikace

Pro tvorbu systému pro podporu e-learningu jsem zvolil webovou aplikaci s využitím HTML, CSS PHP a MySQL. Pro správu obsahu databáze MySQL prostřednictvím webového rozhraní jsem zvolil nástroj phpMyAdmin.

Důvody proč jsem vybral PHP pro svůj systém jsou následující:

- Je volně šířitelné.
- Velmi často používané.
- Existuje mnoho manuálů a jazyk má širokou podporu.
- Dochází k dalšímu vývoji a vylepšením.
- Je rychlé.

Důvody, proč jsem vybral MySQL pro svůj systém, jsou následující:

- Podporuje PHP.
- Dochází k jeho dalšímu vývoji.

Pro vytvoření systému pro podporu e-learnigu jsem využil těchto produktů:

- PHP 5.2.1
- Apache 2.0.55
- MySQL 4.1.22
- phpMyAdmin 2.7.0
- Mozilla Firefox 1.5.0.9
- phpEditorIDE 5.5.3

Jako operační systém jsem využil Windows XP, a proto všechny tyto prostředky byly ve verzi pro toto prostředí. Pro tvorbu aplikace jsem společně s PHP využil HTML a CSS pro formátování vzhledu aplikace. MySQL jsem použil pro vytvoření databáze uživatelů, abych zamezil vstupu nepovolaným uživatelům. Jako editor jsem použil výše zmíněný phpEditorIDE, což je freewarový program umožňující tvorbu www stránek. Tento program umožňuje zvýrazňovat syntaxi a také obsahuje základní nástroje pro PHP, HTML a další.

Díky těmto nástrojům se mi povedlo vytvořit systém pro podporu e-learningu, který je poměrně jednoduchý a přehledný.

6 Programový systém pro podporu e-learningu

V této kapitole se budu zabývat samotným systémem pro podporu e-learningu. Nejprve definuji požadavky na tento systém a objasním jak jsem postupoval při návrhu tohoto systému. Dále se budu věnovat implementační části, ve které popíši, jak byly vytvořeny jednotlivé části aplikace (systému). V posledních podkapitolách popisuji způsob testování funkčnosti a také návod na instalaci.

6.1 Specifikace požadavků a analýza

Tato část se zabývá specifikací požadavků pro systém pro podporu e-learningu. Většina všeobecných požadavků je uvedena v kapitole 4, kde byly tyto požadavky obecně popsány. V této kapitole si tyto požadavky popíšeme podrobněji a popíšeme si i konkrétní požadavky na výslednou aplikaci.

6.1.1 Obecné požadavky

Mezi obecné požadavky na systém pro podporu e-learningu patří:

- Možnost využití systému na jakémkoliv počítači – aplikace je vytvořena pomocí webových stránek. Pro její spuštění je nutné mít nainstalován operační systém MS Windows, Linux, Mac OS a nějaký internetový prohlížeč (Mozilla Firefox, Internet Explorer, Opera, Netscape Navigátor). Aplikace je optimalizována pro prohlížeč Mozilla Firefox, ale měla by fungovat na všech ostatních.
- Internet nebo webový server – dalším kritériem pro spuštění je nutnost připojení k internetu nebo instalace PHP, databázového systému MySQL a webového serveru (nejlépe Apache).
- Jednoduchost a přehlednost - ovládání systému je jednoduché, grafické prostředí přehledné a rychlost odezvy na příkazy uživatele je přijatelná (závisí na HW konfiguraci počítače, případně rychlosti připojení k internetu).

6.1.2 Funkční požadavky

Funkční požadavky bych popsal následovně:

- Přihlašování do systému – do systému se lze přihlásit pomocí jména a hesla, jež jsou uvedena v databázi uživatelů. Ta lze editovat pomocí phpMyAdmin.
- E-learningová aplikace – systém musí umožňovat výuku jazyka pro specifikaci omezení objektů systému. Aplikace obsahuje kompletní OCL specifikaci včetně obrázků a příkladů

- Ověření znalostí – proto aby systém umožňoval prověřování znalostí a sledování pokroku v pochopení prezentované látky, byla vytvořena otázková a testová část, která je na konci každé kapitoly.
- Jazyková podpora – aby byl systém univerzální, byl vytvořen jako dvojjazyčný. První jazyková verze je anglická, kdy veškeré materiály jsou zpracovány v anglickém jazyce. Další verzí je verze česká, kdy je přeložena do češtiny velká část specifikace včetně části testové. To umožňuje využívání aplikace i studentům, kteří nejsou v angličtině až tak zběhlí.

6.2 Návrh

Před zahájením diplomové práce bylo důležité rozvrhnout a naplánovat jednotlivé činnosti tvorby práce tak, abych při tvorbě systému pro podporu e-learningu, nezapomněl na žádný důležitý požadavek na tento systém.

Při návrhu na systém pro podporu e-learningu jsem vycházel z ročníkového projektu. I když jsem v rámci diplomové práce vytvořil zcela novou aplikaci, nechal jsem se inspirovat dřívější prací a zachoval jsem základní strukturu aplikace. To znamená, že jsem využil základní poznatky získané během práce na ročníkovém projektu a také jsem zachoval rozložení stránek aplikace. Jelikož aplikace vytvořená v rámci ročníkového projektu byla po grafické stránce zcela primitivní, musel jsem vytvořit nové grafické rozhraní, které by bylo přehledné a přitom mnohem více přitažlivé.

Jak již jsem zmínil vytvořil jsem proto zcela novou aplikaci využívající kaskádové styly pro formátování www stránky. Takto vytvořené stránky jsou přehlednější a graficky propracovanější (viz kapitola 6).

Další věcí, kterou jsou se zabýval, je přihlašovací část, která znemožní vstoupit na stránky osobě, jež nezná jméno a heslo. Pro to jsem využil databáze MySQL, která obsahuje tabulku uživatelů se jménem a heslem. Vstoupit do databáze uživatelů je možné pomocí programu phoMyAdmin. Poté lze tuto databázi přímo upravovat ať už přidáváním nebo odebíráním položek v tabulce uživatelů.

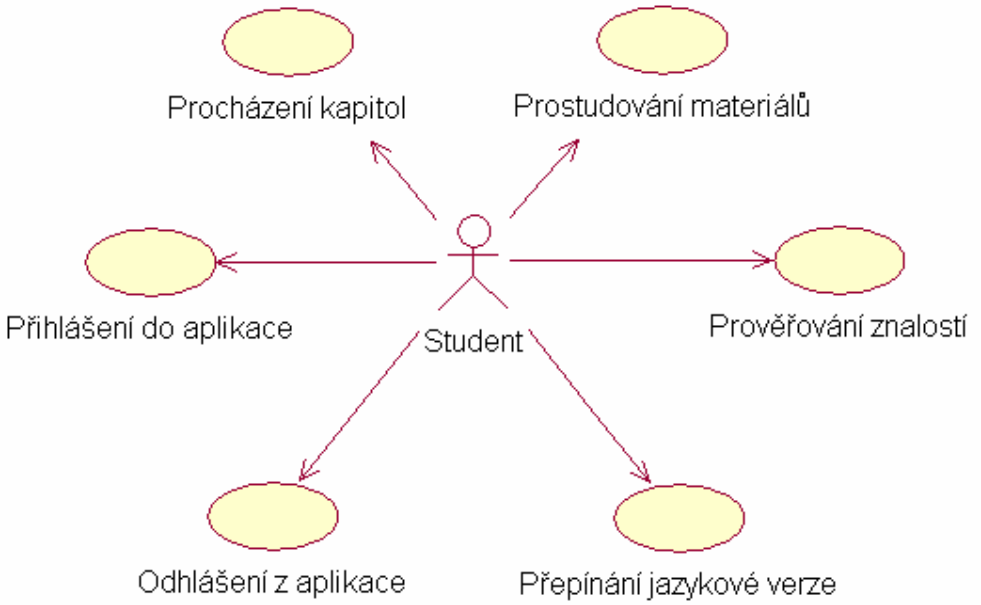
Po přihlášení se již zobrazí úvodní stránka se studijním materiálem. Ten je rozdělen do několika kapitol, kde se každá kapitola zabývá určitou částí specifikace omezení objektů systému. Součásti textu jsou i obrázky a příklady OCL výrazů. Obrázky obsahují diagramy tříd vztahující se k jednotlivým kapitolám. Diagramy jsou ve většině případů zmenšené, avšak pouhým kliknutím na jakýkoliv zmenšený obrázek se nám tento obrázek otevře v původní velikosti v novém okně.

Nedílnou součástí e-learningové aplikace je část, která umožní studentovi prověřování znalostí a sledování pokroku v pochopení prezentované látky. To jsem vyřešil vytvořením otázkové a testové části, která se nachází na konci každé kapitoly. Tato část se skládá z několika otázek, které se ptají na probíranou látku, a z testu. Test je tvořen 10 otázkami. Každá otázka má 2 odpovědi a na studentovi je, aby vybral tu správnou. Podrobněji tuto část popíši v kapitole 6.3.3.

Posledním krokem návrhu systému pro výuku e-learnigu bylo vytvořit systém, který by umožňoval studium zahraničním studentům. Proto jsem vytvořil 2 jazykové verze, které lze snadno přepínat. Jako hlavní jazyk jsem zvolil angličtinu, a tudíž kompletní specifikace je napsána anglicky. Druhá jazyková verze je česká, v níž je přeložena velká část specifikace i testová část.

Před přihlášením si student vybere, v jakém jazyce chce danou problematiku studovat, a podle toho zvolí, jestli vstoupí do anglické nebo české verze. Během pročítání specifikace se lze kdykoliv přepnout do druhé jazykové verze kliknutím na příslušnou ikonu vlaječky. Takto jsem vyřešil i situaci, pokud by studující v češtině potřeboval vědět, jak je daný problém popsán přesně v angličtině. Jelikož se jedná o technický text, je ve většině případů anglické názvosloví jasnější a jednoznačnější.

Následující Use Case diagram zobrazuje možnosti využití systému pro uživatele. Jsou zde zobrazeny základní požadavky na systém, to znamená, že systém by měl umožňovat uživateli všechny tyto činnosti. Viz. obr. 27.



Obr. 27. Use Case diagram aplikace

Podrobněji se jednotlivým částem aplikace budu věnovat v kapitole 6.3, kde si ukážeme způsob implementace jednotlivých součástí systému.

6.2.1 Ročníkový návrh aplikace

V rámci ročníkového projektu bylo vytvořeno základní prostředí e-learningové aplikace. Tato aplikace již obsahovala většinu informací o specifikaci objektů systému. Umožňovala pročítání těchto informací a přehled celou specifikací. Obsahovala ukázky příkladů UML výrazů a také různé diagramy.

6.2.1.1 Formát učebního textu

V této kapitole se budu zabývat formátem učebního textu, který jsem zvolil pro svůj projekt. Pro svůj projekt jsem si zvolil webové prostředí, a to hned z několika důvodů. Hlavním důvodem bylo to, aby byla lehce dostupná kdekoliv na škole. K tomu posloužil školní server, kam ji stačilo nahrát. Pokud budeme chtít aplikaci spustit a používat stačí, když si sedneme k jakémukoliv počítač s připojením k internetu a zadáme správnou webovou adresu do okna prohlížeče. Dalším důvodem byla snaha vytvořit co nejkompatibilnější aplikaci, která byla spustitelná na jakémkoliv počítači kdekoliv na světě. Jedinou nevýhodou tohoto přístupu je nutnost připojení k internetu. Tento problém jsem vyřešil možností dodání aplikace na CD-ROMu a jejím následném spuštění lokálně na jednotlivém počítači bez nutnosti připojovat se na internet. Důležitou roli při výběru bylo také to, že takováto aplikace je velmi přehledná a také v pozdějších fázích snadno upravitelná.

Nyní si povíme něco o použitých technologiích a postupech, které jsem použil při vytváření semestrálního projektu. Projekt jsem vytvořil jako www stránky s pomocí php skriptů. Tento způsob se mi zdál pro tento projekt jako nejvýhodnější.

K vlastnostem WWW stránek patří:

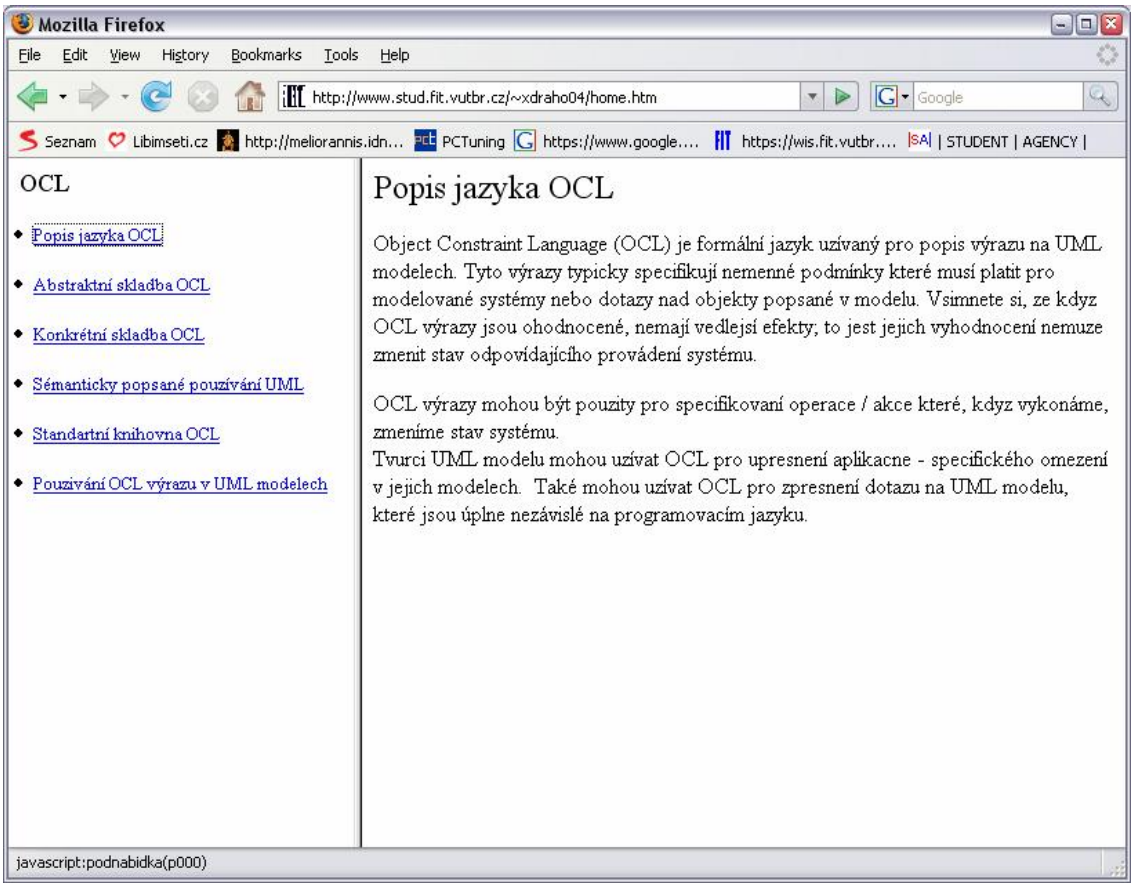
- Používané technologie jsou pouze HTML a v omezené míře PHP bez využití MySQL .
- Veškerý obsah je statický .
- Nehodí se pro prezentaci většího množství strukturovaných dat které je potřeba dynamicky třídit, např. rozsáhlejší ceníky.

6.2.1.2 Vzhled a popis aplikace

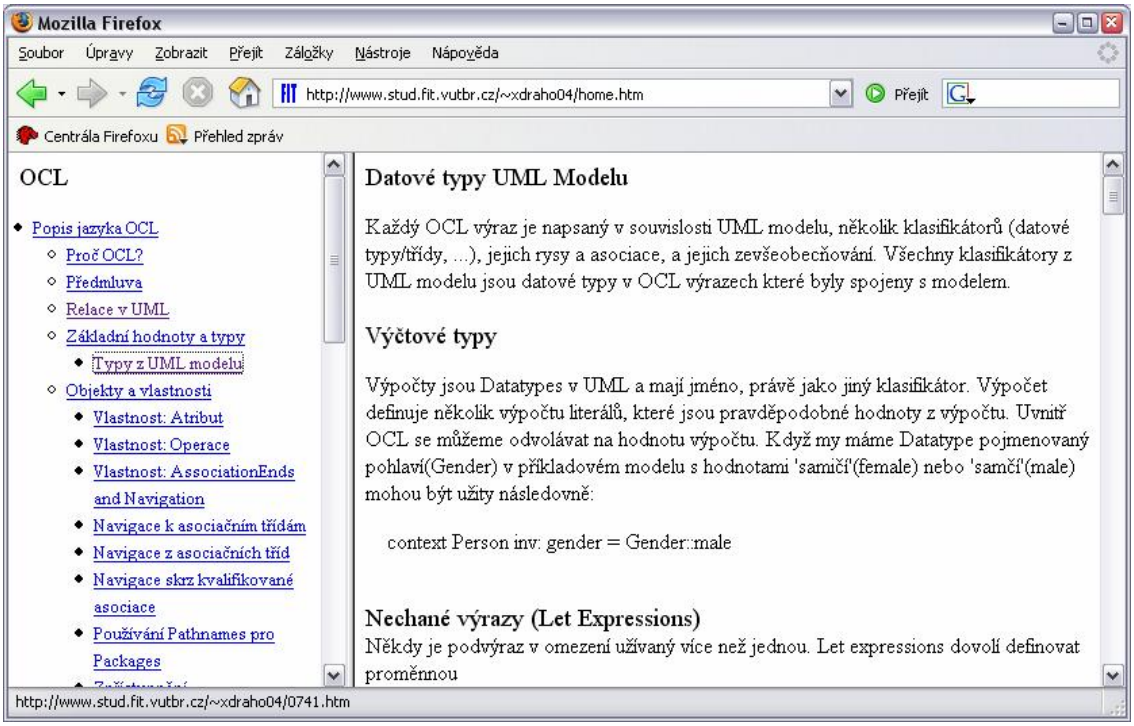
Jak již jsem zmínil v předchozí kapitole, k vytvoření e-learningové aplikace jsem si zvolil webové prostředí a www stránky. Nyní si ukážeme, jak tyto stránky vypadají, jaká je jejich struktura.

Stránky jsou vytvořeny v jazyce html a pomocí PHP skriptů. Po grafické stránce jsou poměrně jednoduché, zato ale velmi přehledné a srozumitelné.

Na obrázku vidíme otevřenou aplikaci. Ta obsahuje 2 okna, z nichž jedno, to vlevo, je tzv. menu okno, ve kterém můžeme vidět obsah celé aplikace. Obsah je rozdělen do 6 hlavních bodů, které se zabývají určitou částí problematiky OCL. Každý bod je vlastně jedna kapitola, která obsahuje několik podkapitol. Podkapitoly se zobrazí, pokud klikneme myší na kapitolu, která nás zajímá. Opětovným kliknutím myši na hlavní kapitolu se nám zobrazené podkapitoly opět schovají jak je zobrazeno na následujících obrázcích. Tímto způsobem můžeme rozbalovat i podkapitoly. To zobrazují obrázky 21. a 22.

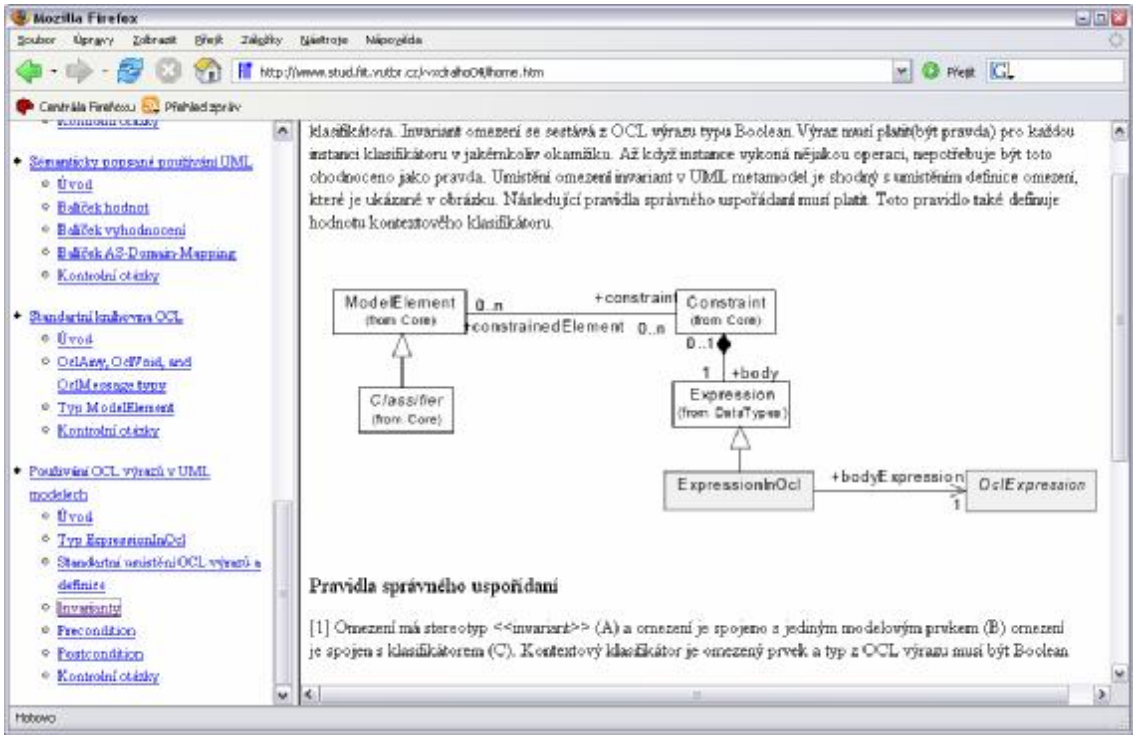


Obr. 21. Aplikace – nerozvinuté menu



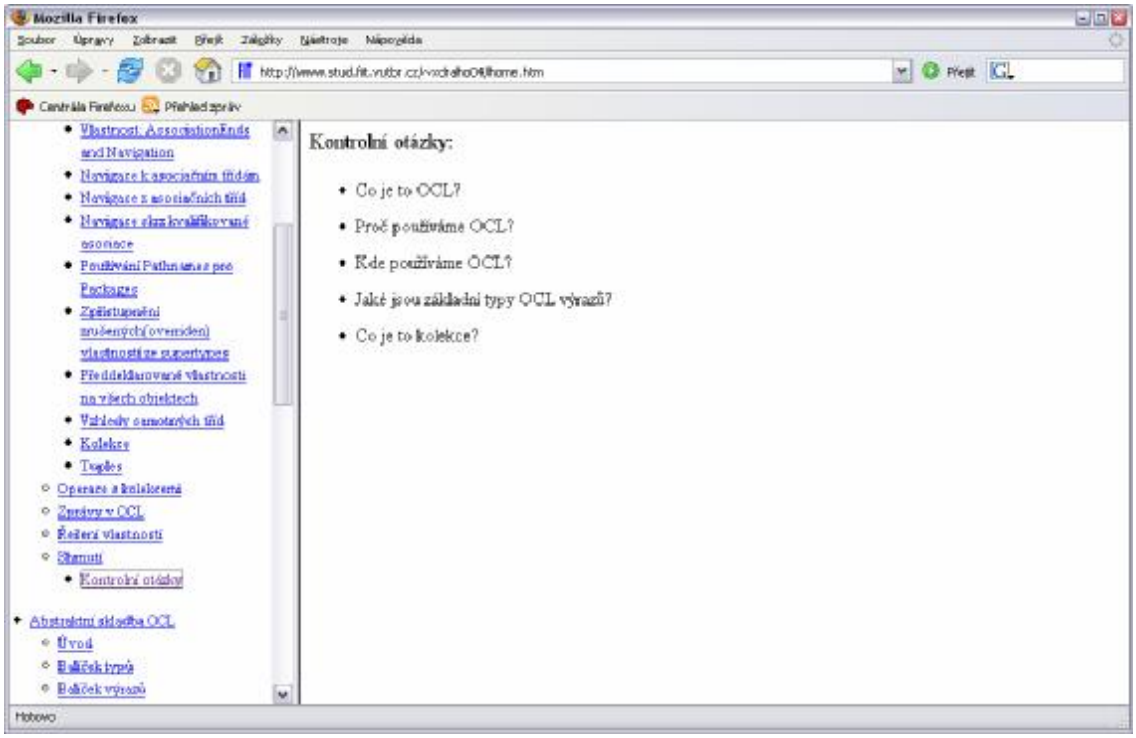
Obr.22. Aplikace – rozvinuté menu

V pravém, větším okně, se nám při kliknutí na jednotlivé kapitoly zobrazí jejich obsah. Ten je uveden velkým nadpisem a dále textem vysvětlujícím danou problematiku. Text je doplněn obrázky a příklady OCL výrazů, což je patrné z obrázku 23.



Obr. 23. Aplikace – diagramy

Na konci každé kapitoly v menu je podkapitola otázky, ve které se nachází několik kontrolních otázek sledujících pokrok v pochopení prezentované látky. Viz. obr. 24.



Obr. 24. Aplikace - otázky

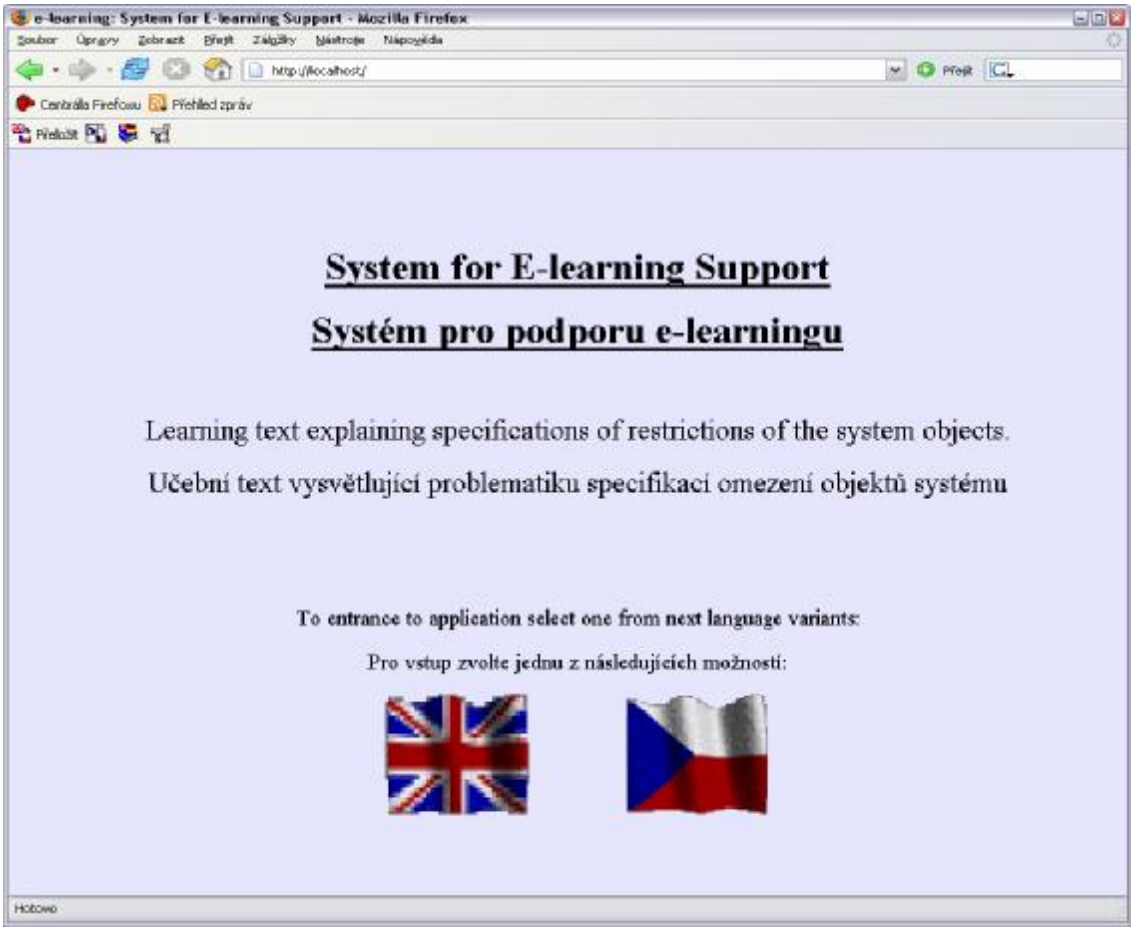
Kompatibilita: Tato aplikace byla testována v prostředí Microsoft Internet Explorer 6.0 a Mozilla Firefox 1.5.

6.3 Implementace

V této kapitole se budu podrobně věnovat všem aspektům aplikace a její tvorby. Postupně popíši jak jsem přistoupil k tvorbě grafického rozhraní, tvorbě jazykové verze nebo testové části. Uvedu zde části zdrojových kódů a náhledy do aplikace. Kompletní zdrojové texty jsou umístěny v příloze 2.

6.3.1 Puštění aplikace a výběr jazykové verze

První stránka slouží pro výběr jazykové verze. Nachází se zde nadpis a jsme vybídnuti k výběru jazykové verze. Tato stránka je napsána pomocí HTML bez PHP. Je to vlastně jen takový rozcestník (viz obr. 28.).



Obr. 28. Úvodní stránka – Výběr jazykové verze

Ukázka hlavičky stránky:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

… tato hlavička nám říká o jaký typ dokumentu se jedná (viz. Kapitola 5.2)

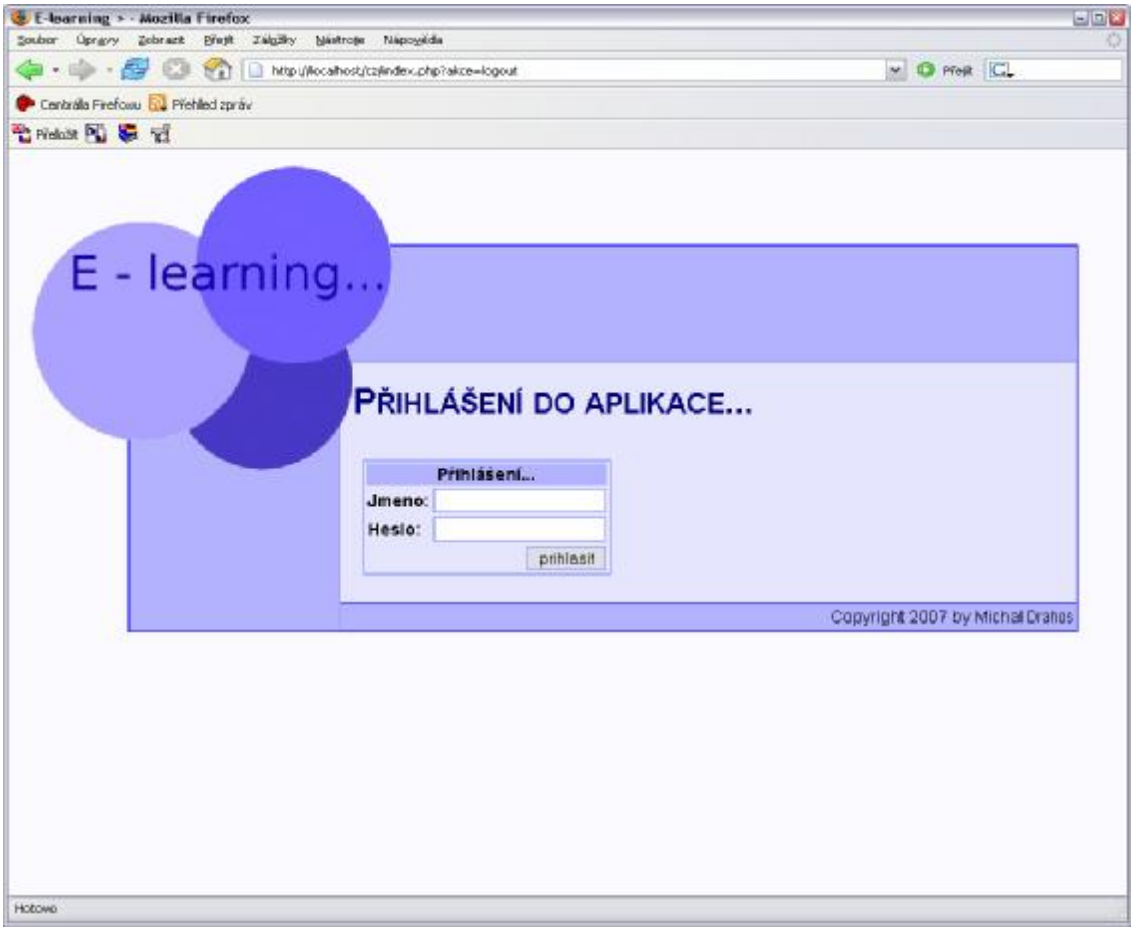
```
<html>
<head>
<meta http-equiv='Content-Type' content='text/html; charset=ISO-8859-2' />
```

… tento meta tag nám zaručuje správné zobrazování češtiny (viz. 6.3.4)

```
<title>e-learning: System for E-learning Support</title>
```

6.3.2 Modul přihlašování

Po výběru jazykové verze se nám zobrazí přihlašovací okno aplikace. Po vyplnění správného jména a hesla můžeme pokračovat. Pokud vyplníme nesprávné údaje nebo žádné, systém nám o tom vypíše příslušnou hlášku. Následující obrázek zobrazuje přihlašovací stránku v české verzi (viz 6.3.4). Obrázek 29. zobrazuje přihlašovaní obrazovku do aplikace.



Obr.29. Přihlášení do aplikace

Přihlašování se provádí vyplněním příslušných údajů do formuláře (viz obr. 29.). Pro studijní účely byl vytvořen učet s jménem ***student*** a heslem také ***student***. Pokud vyplníme jiné údaje vstoupit do aplikace se nám nepodaří. Systém ověřuje zadané údaje s údaji v databázi uživatelů. Tudíž je pro správnou funkci celé aplikace nutné se připojit k databázi. HTML formulář pro přihlášení vypadá následovně:

```
<form action=\"index.php?article=1\" method=\"POST\">\n
 <input type=\"hidden\" name=\"odeslat\" value=\"1\">
 <table id=\"login\">\n
 <tr>\n
 <th colspan=\"2\">Přihlášení...</th>\n
 </tr>\n
 <tr>\n
 <td class=none><strong>Jméno:</strong></td>\n
 <td class=none><input type=\"text\" class=log name=\"login\"></td>\n
 </tr>\n
 <tr>\n
```

```
<td><strong>Heslo:</strong></td>\n
<td><input type=\"password\" name=\"password\"></td>\n
</tr>\n
<tr>\n
<td></td>\n
<td><input type=\"submit\" class=\"button\"  value=\"přihlásit\"></td>\n
</tr>\n
</table>\n
</form>\n
```

U tagu form jsou uvedeny 2 parametry, které se týkají zpracování dat. Jsou to parametry ***action*** a ***method***. Parametr ***action*** udává, který skript bude data zpracovávat a parametr ***method*** udává jakým způsobem budou data předána. Pro předávání dat se využívají 2 metody a to GET a POST. V tomto případě využívám metodu POST, jelikož tato metoda zasílá data jako samotný http objekt, což je v případě hesla bezpečnější, jelikož tyto údaje nejsou vidět jako v případě metody GET, kdy jsou zobrazovány přímo v url.

6.3.2.1 Databáze

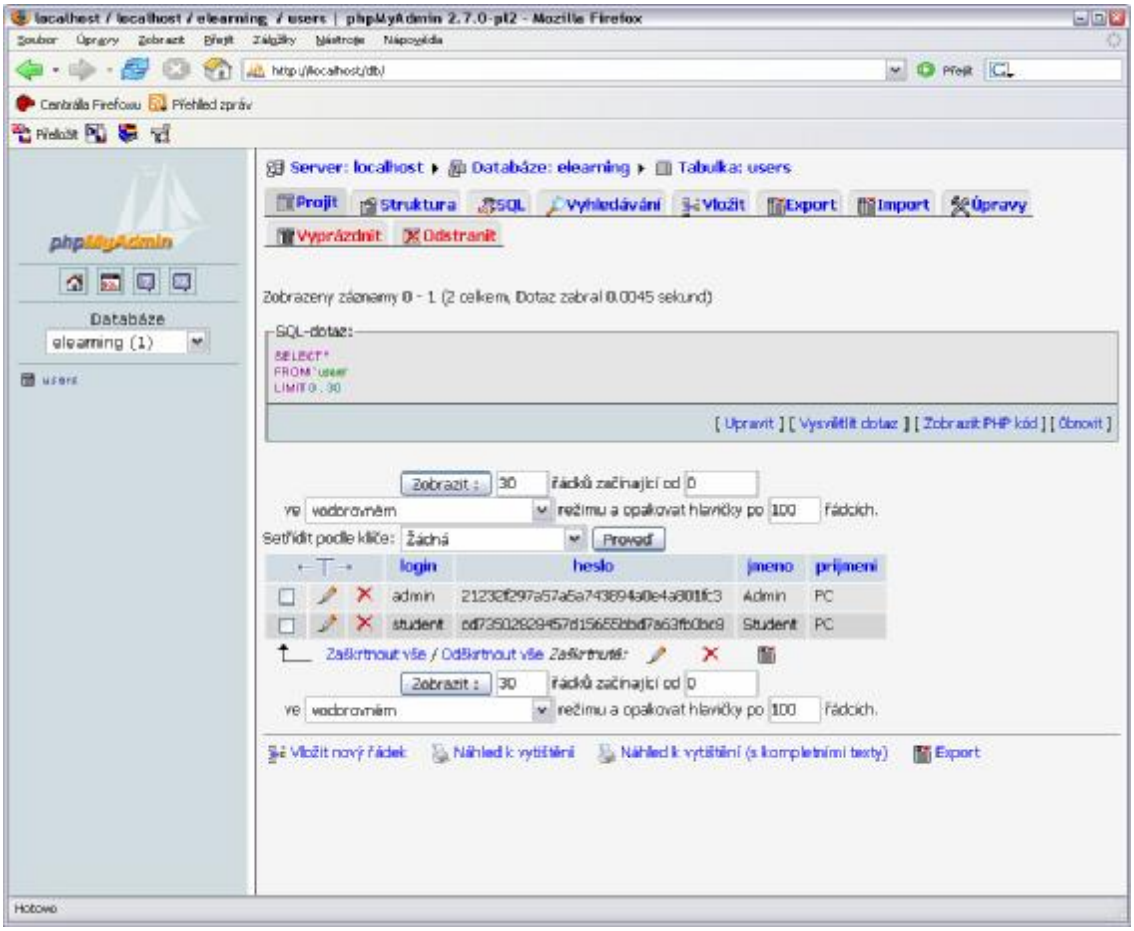
Obsahem databáze, která se nazývá ***elearning*** je tabulka ***users***, která obsahuje údaje o uživatelích. Jsou to údaje ***jméno, příjmení, login*** a ***heslo***. Pro účely přihlášení jsou podstatné údaje ***login*** a ***heslo***. Jak vypadá SQL příkaz pro vytvoření tabulky uživatelů je ukázáno zde:

```
CREATE TABLE `users` (
  `login` varchar(20) collate latin2_czech_cs NOT NULL default '',
  `heslo` varchar(50) collate latin2_czech_cs NOT NULL default '',
  `jmeno` varchar(20) collate latin2_czech_cs NOT NULL default '',
  `prijmeni` varchar(20) collate latin2_czech_cs NOT NULL default '',
  PRIMARY KEY  (`login`)
) ENGINE=MyISAM DEFAULT CHARSET=latin2 COLLATE=latin2_czech_cs;
```

Po vytvoření tabulky ji musíme naplnit údaji:

```
INSERT INTO `users` (`login`, `heslo`, `jmeno`, `prijmeni`) VALUES ('student', 'cd73502828457d15655bbd7a63fb0bc8', 'Student', 'PC');
```

Vytváření databáze a tabulky je prováděno přes prostředí phpMyAdmin. Prostředí phpMyAdmin ukazuje obrázek 30.



Obr. 30. phpMyAdmin

Jak tento nástroj nainstalovat si popíšeme později v kapitole 6.3.7.

Při běhu aplikace je potřeba se připojovat k MySQL databázi. K tomuto využívám následující PHP funkci:

```
function dbcon()
{
 $spojeni=mysql_connect($GLOBALS["dbserver"],$GLOBALS["dbuser"],$GLOBALS["dbpass"]);
 if (!$spojeni):
   echo "<div align=\"center\">Spojeni se serverem nelze vytvorit!</div>";
   exit;
 endif;
 mysql_select_db($GLOBALS["dbname"],$spojeni);
 return $spojeni;
}
$dbspojeni=dbcon();
$GLOBALS["dbspojeni"]=&$dbspojeni;
```

Je samozřejmé, že místo údajů "dbserver","dbuser","dbpass" a "dbname", musí být správné údaje o databázovém serveru, jménu databáze a správné přihlašovací údaje.

6.3.2.2 Zabezpečení přihlašovaní

Do systému se je možné přihlásit pouze jako uživatel student. Pokud tak neučiníme nemůžeme nijak pokračovat ve výuce. Systém rozeznává jeden typ uživatele. Pokud bychom měli zájem, můžeme do systému přidat více uživatelů, a to editováním tabulky uživatelů, ovšem tito uživatelé budou mít stejná práva.

Zabezpečení je prováděno na základě existence uživatelského účtu. Pro úspěšné přihlášení je nutné znát jméno (login) a heslo. Tyto údaje jsou uvedeny v tabulce uživatelů jak již jsem zmínil dříve. Uživatelská jména jsou ukládána textově, avšak pro heslo jsem zvolil bezpečnější způsob. Heslo je v tabulce uloženo jako výsledek hashovaní funkce. V implementaci jsem zvolil funkci MD5, kterou PHP obsahuje. Pro vytvoření hesla stačí vložit heslo v textové podobě a zvolit typ zašifrování. V našem případě je to funkce MD5. Po potvrzení máme v tabulce uložen již výsledek hashovaní funkce. I když se ukázalo že MD5 není odolná vůči kolizím, a lze ji tudíž rozšifrovat, pokud by útočník získal z tabulky hash a dokázal k němu nalézt kolizi. Pro zabezpečení toho systému je tento způsob dostačující. Pokud bychom chtěli tento systém více zabezpečit, museli bychom zvolit jiný typ zabezpečení, popřípadě použít další technologie (např. SSL).

Ukázka hashovaní funkce:

Pokud bychom použili heslo ***student***, pak bude v tabulce uživatelů uložen výsledek funkce MD5, který bude vypadat takto *cd73502828457d15655bbd7a63fb0bc*.

6.3.3 Grafické rozhraní

Důležitou vlastností systému pro podporu e-learnigu je jeho přehlednost. Systém musí být přehledný a jednoznačný, neměl by obsahovat zbytečně moc funkcí, které by uživatele rozptylovaly. Proto jsem se rozhodl vytvořit jednoduché a přitom přehledné grafické rozhraní, které by bylo uživatelsky příjemné.

Základem bylo vytvořit konstrukci, do které bych pak umístil studijní text. Při vytváření vzhledu systému jsem vycházel z ročníkového projektu, kde byla základem 2 okna. Jedno s obsahem umožňujícím procházení jednotlivých kapitol a druhé, které by obsahovalo studijní text. Tohoto formátu jsem se držel i při tvorbě diplomové práce.

Implementaci grafického prostředí jsem provedl v několika fázích. Nejprve jsem si musel připravit prvky grafické aplikace, jednotlivé obrázky, a poté jsem upravoval vzhled každé stánky pomocí CSS a HTML.

První stránkou aplikace je stránka umožňující volbu jazykové verze. Tato stránka je vytvořena pouze pomocí HTML. Jedná se pouze o jednoduchý rozcestník. Viz obr. 29. Až po volbě jazykové

verze se nám načte stránka, jejíž vzhled byl vytvořen pomocí CSS. Je to stránka obsahující okno pro zadání přihlašovacích údajů. Tato stránka se nám zobrazí vždy před přihlášením a po odhlášení z aplikace. Tato stránka je ukázána na obrázku 30.

Jádrem aplikace je soubor index.php, respektive stránka index.php. Na této stránce se rozhoduje, zda se zobrazí přihlašovací okno, nebo již okno se studijním materiálem. Do stránky je vkládán soubor menu.php v případě, že se uživatel přihlásí. Pokud ne, zobrazí se okno pro přihlášení do aplikace. Tato stránka obsahuje základní rozvržení všech prvků, které aplikace obsahuje. Stránka je vytvořena několika rámy. Tyto rámy můžeme vidět už v přihlašovacím okně (viz obr. 30.). Názorněji jsou ovšem vidět po přihlášení v okně s menu a studijním materiálem.

6.3.3.1 Využití CSS

Rozvržení těchto rámu je nastaveno v souboru styles.css. Jedná se o CSS soubor, který obsahuje definice CSS stylů, které se používají na každé stránce. V souboru index.php je vložen tag <link rel="stylesheet" href="styles.css">, který nám zajišťuje toto propojení. Jak již jsem zmínil všechny styly jsou uvedeny v souboru styles.css. Jedná se o styly definované pro jednotlivé tagy v HTML (h1, h2, h3, div), nebo jsou vytvořeny a pojmenované nové styly. Ty jsou pak přiřazovány atributem *class*. Například v souboru styles.css máme definovaný styl example následovně:

```
p.example {
  border: 1px solid #7c8bff;
  border-left: 10px solid #7c8bff;
  margin-right: 1em;
  margin-top: 1em;
  margin-bottom: 1em;
  margin-left: 0.5em;
  padding: 15px;
  padding-left: 50px;
  background: url(example.png) no-repeat 2% 50%;
```

Pokud chceme na nějaké stránce tento styl aplikovat na odstavec musíme před tento text vložit tag *nějaký odstavec* . Vše si ukážeme na příkladu.

Příklad:

Takto vypadá odstavec, na který chceme aplikovat daný styl.

context TypeName **inv**:
'this is an OCL expression with stereotype <<invariant>> in the
context of TypeName' = 'another string'

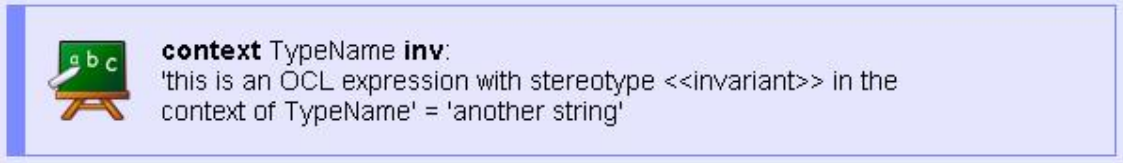
Po vložení tagu vypadá tento odstavec v jazyce HTML následovně:

```
<p class="example">   <b>context</b> TypeName <b>inv</b>:<br>
   'this is an OCL expression with
```

stereotype <<invariant>> in the

   context of TypeName' = 'another string'</p>

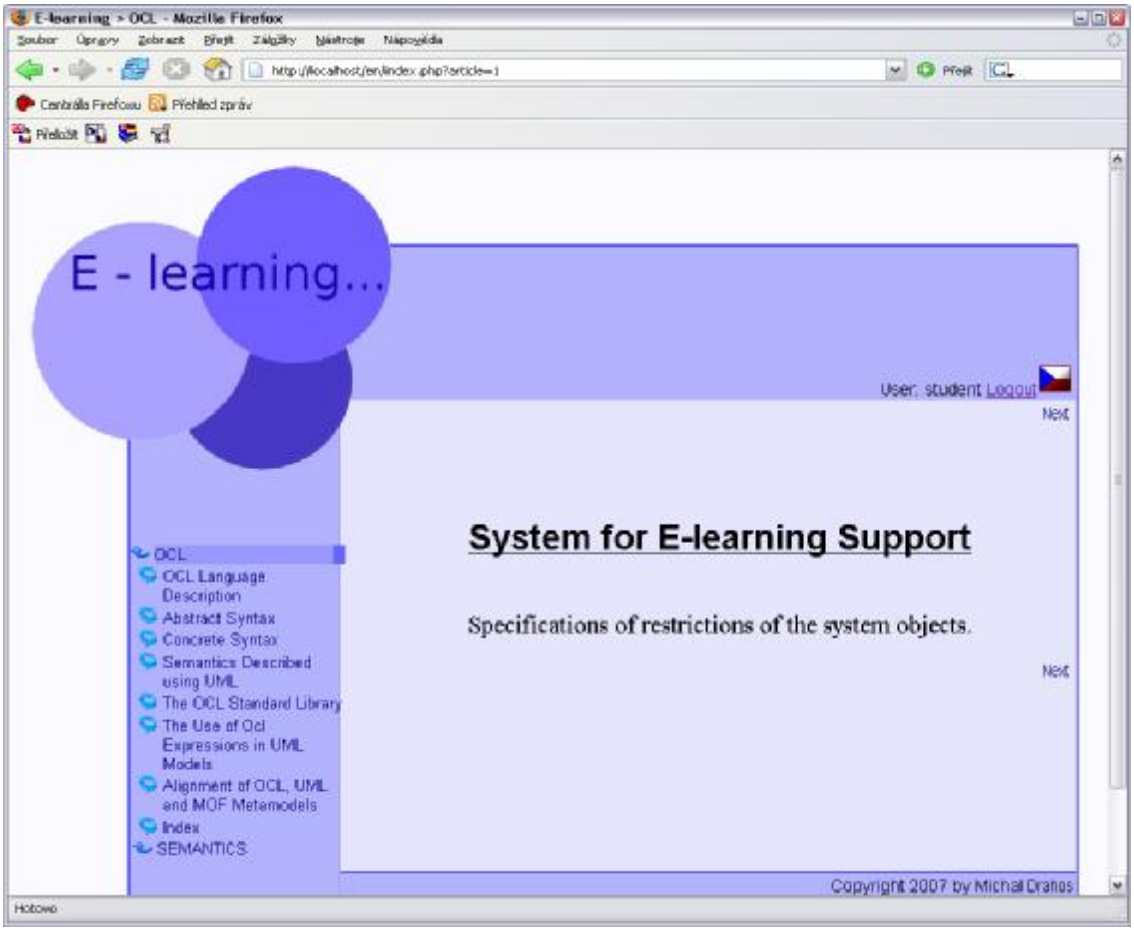
Následující obrázek 31. zobrazuje, jak vypadá výsledek v okně aplikace.



Obr. 31. Využití stylu

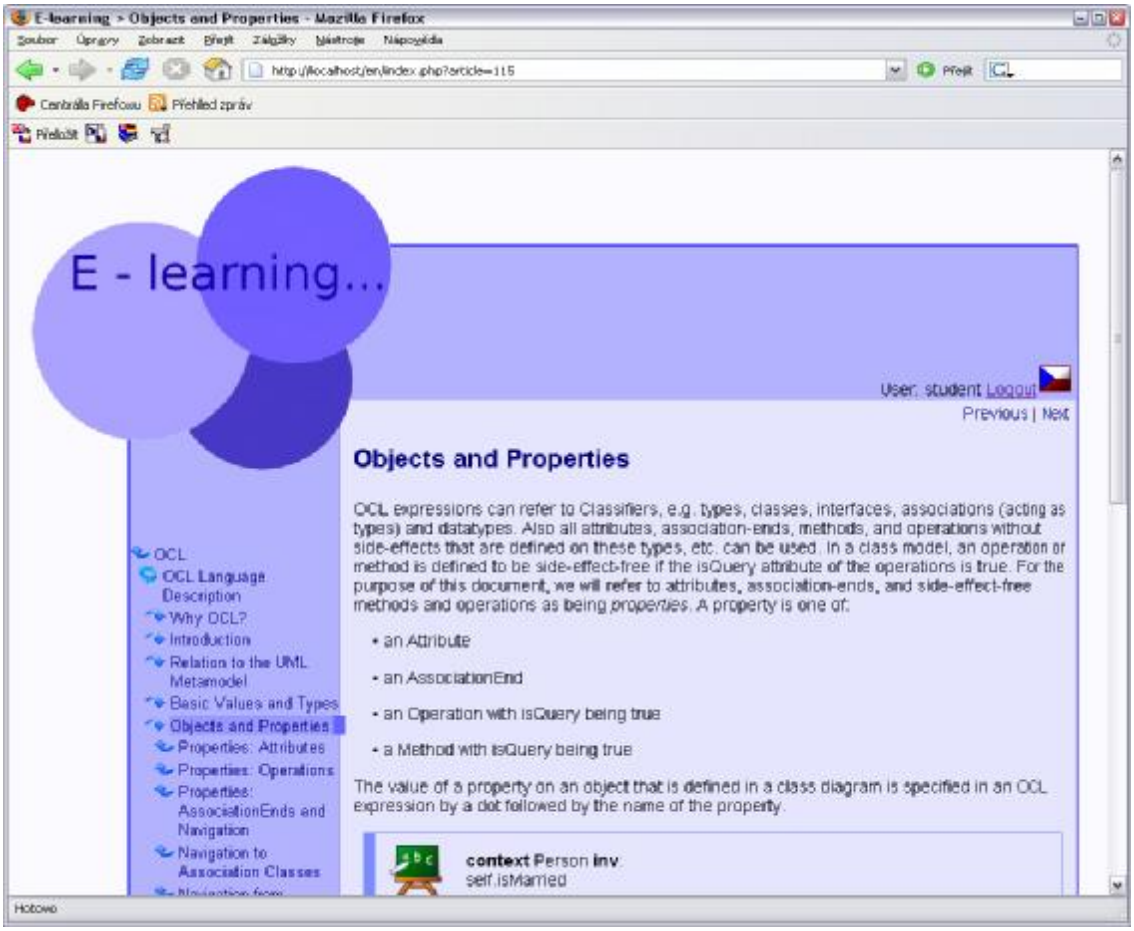
6.3.3.2 Rozvržení stránky aplikace

Každá stránka aplikace je rozdělena na 4 části, přesněji 4 rámy. V horní části se nachází rám obsahující obrázek a dále údaje o uživateli, odkaz pro odhlášení a ikonku pro změnu jazykové verze. Ve spodní části je rám obsahující pouze údaje o tvůrci aplikace. Nejdůležitější části aplikace jsou ovšem 2 rámy mezi horním a dolním rámem. Tyto 2 rámy tvoří jádro aplikace, jelikož se v nich zobrazují veškeré informace o specifikaci omezení objektů systému. Jak tato stránka vypadá, ukazuje následující obrázek (viz obr. 32.).



Obr. 32. Vzhled stránky po přihlášení (anglická verze)

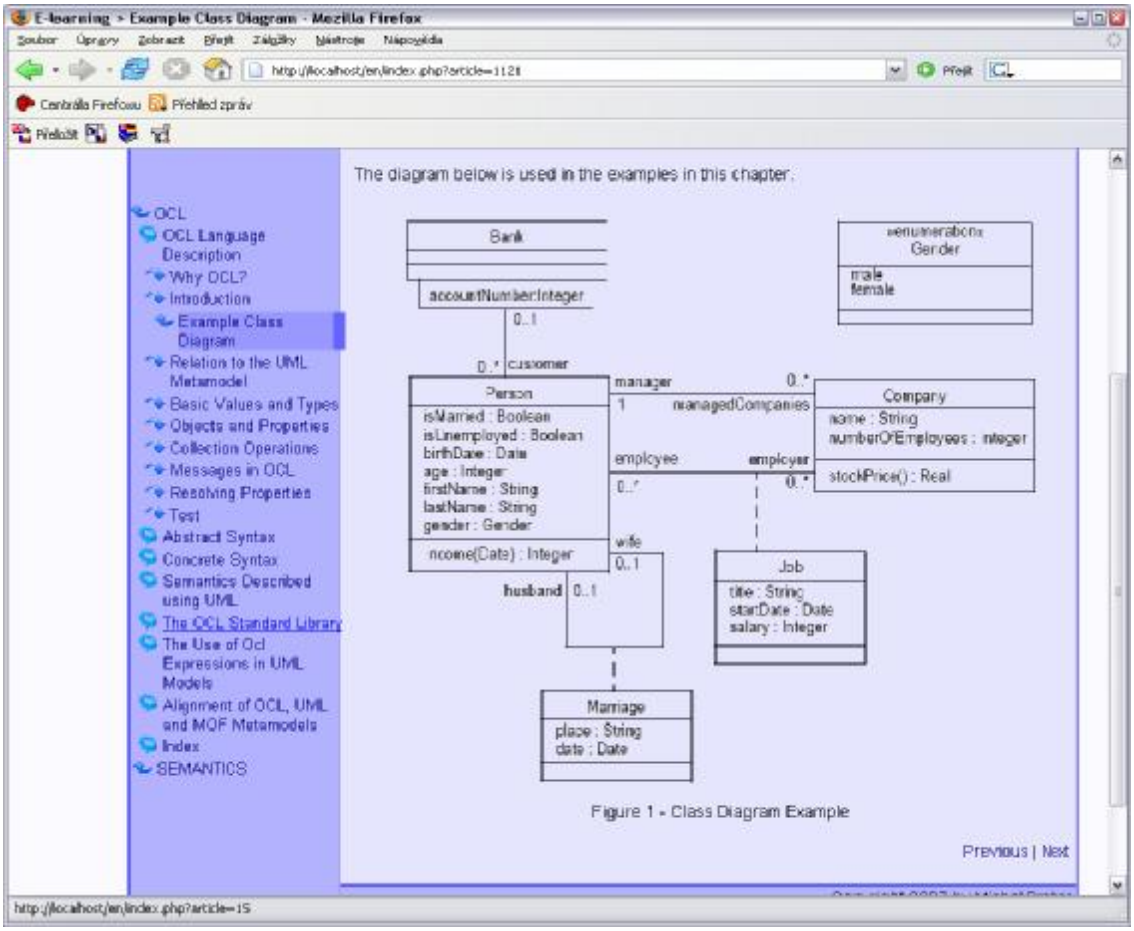
Jak můžeme vidět na obrázku 32., v levé časti se nachází menší rám obsahující menu. Jedná se o obsah umožňující procházet jednotlivé kapitoly specifikace omezení objektů systému. Po kliknutí na jednotlivé kapitoly se nám zobrazí další podkapitoly. Pokud i ty obsahují nějaké podkapitoly, opětovným kliknutím se nám tyto kapitoly rozbalí (viz. obr. 33.).



Obr. 33. Okno aplikace - rozvinuté menu (anglická verze)

Pokud budeme v nějaké zanořené kapitole a chtěli bychom se vrátit do hlavní kapitoly, stačí na tu kapitolu kliknout a menu se nám zavine zpět tak, že budeme vidět pouze kapitoly na stejné úrovni.

V pravé části aplikace se zobrazuje obsah jednotlivých kapitol. Jedná se o studijní text pojednávající o specifikaci omezení objektů systému. Text je doplněn příklady a obrázky obsahující diagramy tříd (viz obr. 34.,36.).

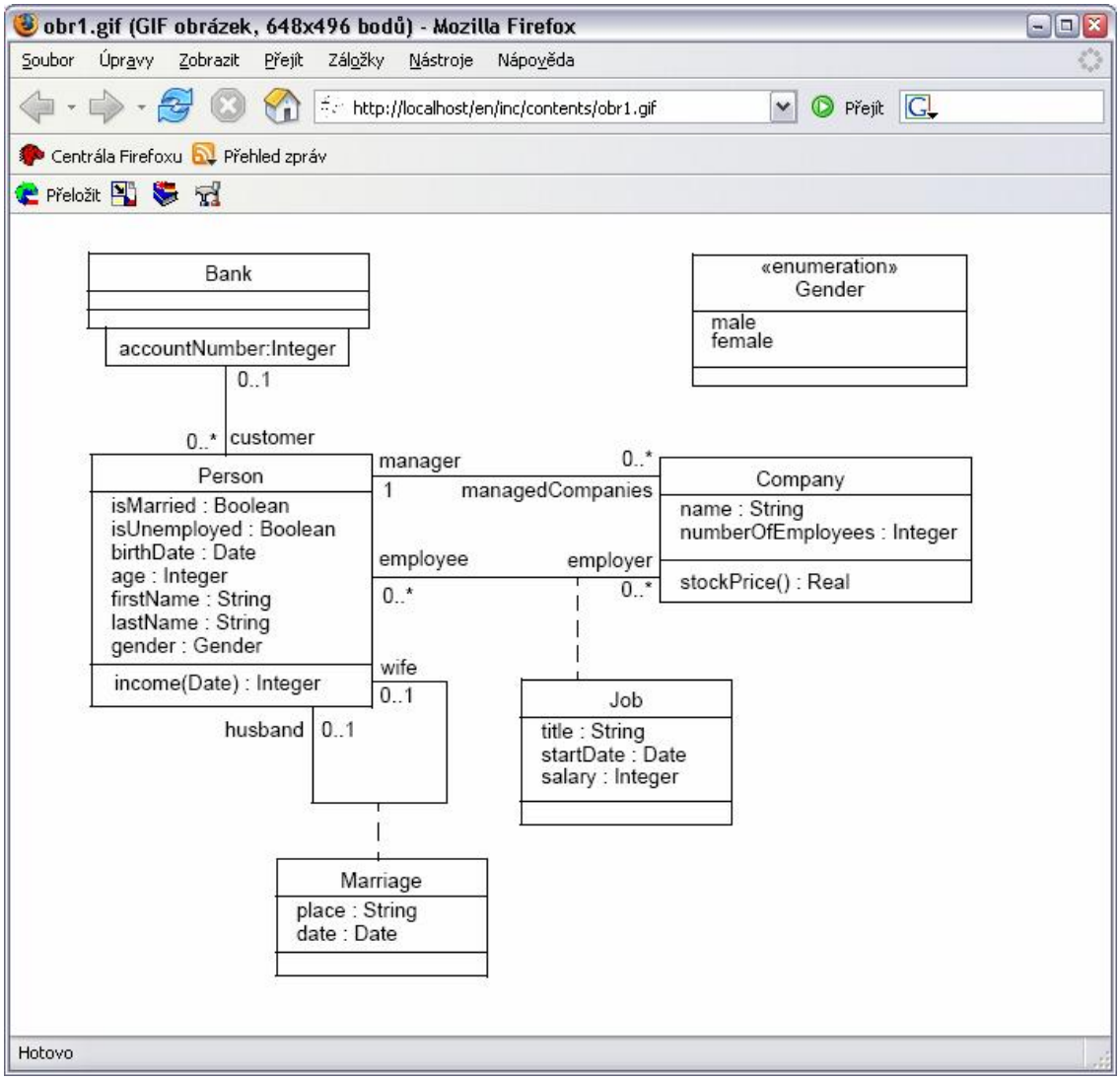


Obr. 34. Aplikace – diagram tříd

Z důvodů velké velikosti diagramů jsem tyto obrázky musel trochu zmenšit, aby se vešly do okna aplikace. Proto mohou být některé obrázky méně kvalitní. Proto lze na každý obrázek kliknout a on se nám otevře v plné velikosti v novém okně (viz obr. 35.). Všechny obrázky, u kterých bylo nutné provést tuto změnu, jsou tudíž i odkazy. To je ukázáno v následujícím příkladu:

```
<p align="center"><a href="inc/contents/obr1.gif" target="_blank"><img src="inc/contents/obr1.gif"
id="obr" alt="diagram trid" ></a><br>
Figure 1 - Class Diagram Example </p>
```

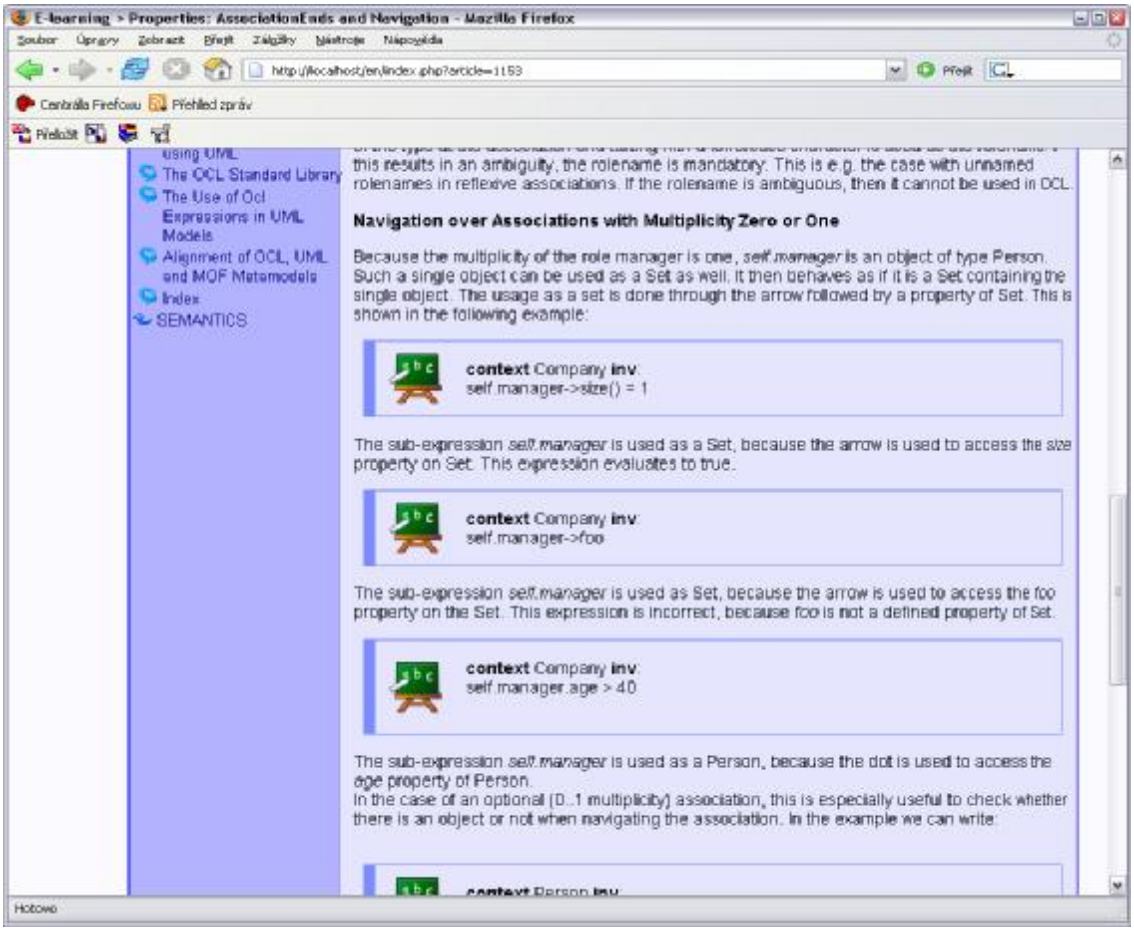
Obrázky, jejichž velikost nepřesahuje velikost okna aplikace, byly ponechány v původní velikosti.



Obr. 35. Náhled na obrázek v původní velikosti

Pro zpřehlední jsem příklady OCL výrazů zvýraznil, jak je vidět na obrázku 36. Toho jsem dosáhl pomocí kaskádových stylů (viz. kapitola 6.3.3.1).

Nezbytnou součástí této aplikace je možnost procházení studijního materiálu stránku po stránce. Proto každá stránka obsahuje odkazy na stránku předchozí a následující, jak je vidět na obrázcích 32.,33.,34. Tyto odkazy se nacházejí vždy vpravo nad a pod textem, kterým se daná kapitole zabývá.



Obr. 36. Aplikace – text s příklady OCL výrazů

6.3.4 Menu

V této kapitole popíšu postup vytváření menu, neboli obsahu. Při vytváření jsem vycházel z ročníkového projektu a dokumentace k OCL. Proto jsem zvolil podobnou strukturu, jakou má tato dokumentace.

Každá kapitola je rozdělena do několika podkapitol. Aby toto menu bylo přehledné, zvolil jsem takzvanou rozbalovací strukturu. To znamená, že na začátku jsou vidět jen hlavní kapitoly, a až po kliknutí na určitou kapitolu se ukáží další podkapitoly. Pokud se chceme vrátit zpět stačí zase kliknout na nadřazenější kapitolu a podkapitoly se zavinou zpět pod kapitolu, pod kterou náleží. Následující příklad ukazuje, jakým způsobem je menu vytvořeno.

```
if ($l2 == 7) {
  echo "<li><a href=\"$link=16\"><span "; if ($id==16) echo "class=\"selected\""; echo ">Alignment of OCL, UML and MOF Metamodels</span></a>
  <ul class=\"level2\">";
   if ($l3==1) {
```

```
    echo "<li><a href=\"$link=171\"><span "; if ($id==171) echo "class=\"selected\""; echo
">Introduction</span></a>";
  }  else echo "<li><a href=\"$link=171\">Introduction</a></li>";
  if ($l3==2) {
    echo "<li><a href=\"$link=172\"><span "; if ($id==172) echo "class=\"selected\""; echo ">Use of
the UML Metamodel</span></a>";
  }  else echo "<li><a href=\"$link=172\">Use of the UML Metamodel</a></li>";
```

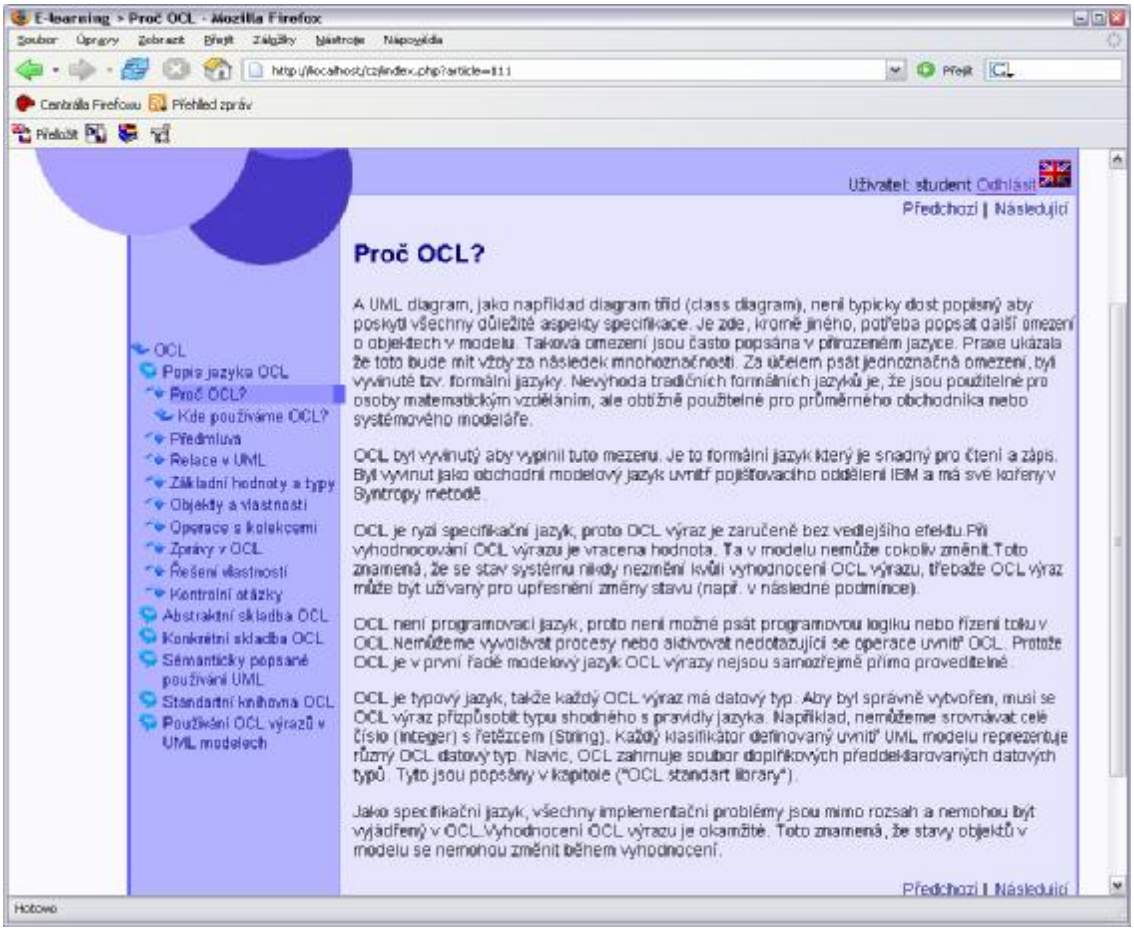
Toto je pouze malá část popisující zápis jedné kapitoly s dvěma podkapitolami. Výsledný soubor obsahuje kolem 600 řádků textu.

6.3.5 Zpracování studijních materiálů

Nejdůležitější a nejvíce časově náročnou prací v rámci tvorby diplomové práce bylo zpracování materiálů specifikace omezení objektů systému. Vycházel jsem ze UML (2.0) OCL Specifikace [21], která je dostupná na adrese <http://www.omg.org/docs/ptc/03-10-14.pdf>. Abych mohl tento systém pro podporu e-learningu vytvořit, musel jsem z každé stránky této specifikace vytvořit html stránku, kterou jsem pak použil ve svém systému. Tato specifikace obsahuje přes 200 stran textu včetně tabulek a obrázků. Musel jsem proto procházet tuto specifikaci stránku po stránce a postupně vytvářet html stránky, které by obsahovaly kompletní OCL specifikaci. Jelikož jsem vytvářel 2 jazykové verze, musel jsem značnou část textu překládat a obsah zpracovaného materiálu se téměř zdvojnásobil (viz kapitola 6.3.6). Zčásti jsem vycházel i z ročníkového projektu, ve kterém jsem již měl část této specifikace ve formátu html. Také jsem musel vytvořit diagramy tříd, ke kterým se vztahují jednotlivé příklady OCL výrazů.

6.3.6 Jazykové verze

Dle návrhu systému jsem vytvořil 2 jazykové verze. Kompletní specifikace OCL je zpracována v anglickém jazyce a druhým jazykem je čeština. Do češtiny jsem přeložil velkou část specifikace včetně testové části. Tak je tento systém vhodný i pro studium dané problematiky v českém jazyce. Bohužel není tato specifikace přeložena kompletně, tudíž se jí nelze věnovat pokud uživatel nemá alespoň základy angličtiny. Pro vytvoření kompletní české verze musí dojít k přeložení zbývajících částí a upravení zdrojových kódů obsahující jednotlivé kapitoly specifikace. Mezi jednotlivými jazykovými verzemi se lze jednoduše přepínat a to kdykoliv během studia materiálů. Stačí kliknout na ikonu vlaječky. Jak vypadá aplikace v české verzi zobrazuje obrázek 37.



Obr. 37. Aplikace v české verzi

6.3.6.1 Jazykové rozhraní

Tato aplikace obsahuje anglické a české jazykové rozhraní a to včetně diakritiky. Aby nedocházelo k problémům se zobrazováním diakritiky, zvolil jsem pro kódování všech PHP a HTML stránek kódování ISO-8859-2. Tato informace je uvedena v hlavičce v následujícím formátu:

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

Kódování ISO-8859-2 jsem použil z důvodů využití systému na co největším počtu počítačů. Toto kódování je používáno na většině uniových serverů, a tudíž jeho volba je nejvýhodnější. Systém jsem testoval i na webovém serveru www.webzdarma.cz, na kterém nedocházelo k žádným problémům se zobrazováním diakritiky.

6.3.7 Testová část

Dle zadání musí být součástí systému část, která by umožňovala prověřování znalostí a sledování pokroku v pochopení prezentované látky. Proto se na konci každé z kapitol se nachází testová část. Ta se skládá z několika kontrolních otázek a testu.

6.3.7.1 Kontrolní otázky

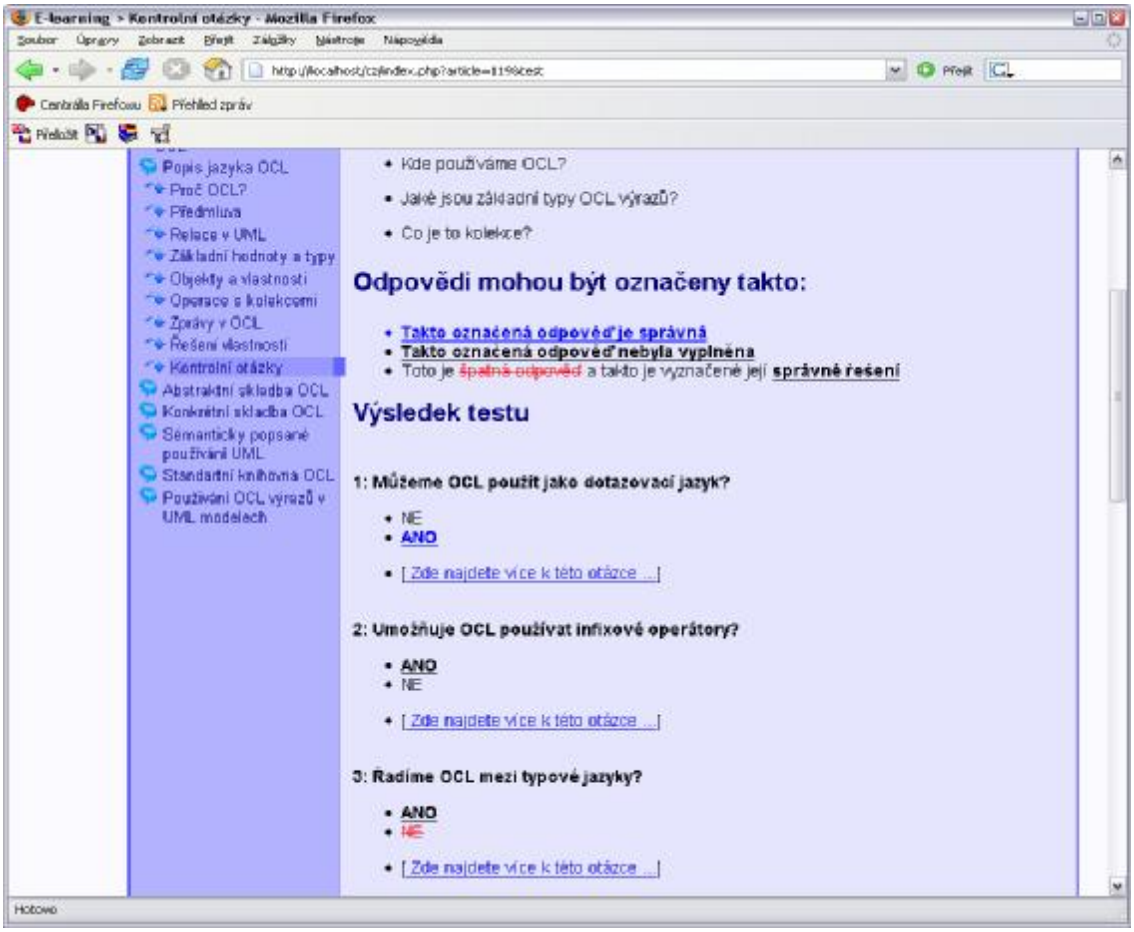
Kontrolní otázky jsou koncipovány tak, aby ověřily, zda uživatel pochopil danou problematiku. Na konci každé kapitoly je soubor obsahující několik takovýchto otázek a dále php příkaz vkládající k těmto otázkám testovou část. Jak vypadá PHP příkaz pro vložení testu, ukazuje následující příklad:

```
<?php
  include('test.php');
?>
```

6.3.7.2 Test

Součástí každé kontrolní části je test. Test obsahuje 10 otázek, každá otázka má 2 odpovědi, z toho jednu správnou. Na uživateli je vybrat správnou odpověď.

Testové otázky a odpovědi jsou uloženy v souboru oXXX.php. Ten patří k souboru XXX.php, ve kterém jsou kontrolní otázky. V souboru test.php se nachází funkce, která načte příslušné testové otázky a odpovědi náležející k dané kapitole ze souboru oXXX.php a ty zobrazí pod kontrolními otázkami. Tyto testové otázky zobrazí v různém pořadí a změní i pořadí odpovědí, takže test při každém načtení vypadá trochu jinak. To zabrání uživateli naučit se odpovědi v nějakém pořadí.



Obr. 38. Test (česká verze)

Také se zde nachází vyhodnocovací část. Ta provede kontrolu vyplnění testu a test vyhodnotí. Výsledek pak napíše na konec ve formě, kolik odpovědí bylo správných. Po kontrole testu se nám také zobrazí, které odpovědi byly správné, které byly špatné a na které se neodpovědělo. Také se zobrazí odkaz na kapitolu, ke které se daná otázka vztahuje. Viz obr. 38.

6.3.8 Testování a funkčnost

Systém jsem vytvářel a testoval lokálně na počítači s MS Windows XP SP2. Současně jsem měl nainstalované PHP, MySQL a webový server Apache ve verzích uvedených v kapitole 5.7, které jsou pro funkčnost a implementaci systému nezbytné. Funkčnost a vzhled systému jsem ověřoval v prohlížeči Mozilla Firefox 1.5, Internet Explorer 6.0, Opera 9.1. Systém je plně funkční pod všemi zmíněnými prohlížeči, avšak je optimalizován pro prohlížeč Mozilla Firefox, a proto doporučuji používat aplikaci s tímto prohlížečem.

K ověření funkčnosti jsem dále testoval systém na webovém serveru zeus.anoweb.cz (www.webzdarma.cz) s nainstalovaný OS Linux 2.6.18, PHP 4.3.4, MySql 5.0.27, Apache 2.0.59 a phpMyAdmin 2.6.0. URL adresa aplikace je http://ocl.wz.cz. Během testování a ověřování funkčnosti nedošlo k žádným komplikacím a systém splňoval všechny požadavky.

6.3.9 Aktualizace obsahu

V době dokončení systému se na internetu objevila novější verze UML 2.0 OCL specifikace. Pokud bychom chtěli stávající verzi aktualizovat, bylo by potřeba postupně projít novou specifikaci, zjistit rozdíly a ty pak přidat do systému. Tuto změnu by bylo možné provést pouze úpravou zdrojových kódů. Musel by se upravit obsah jednotlivých kapitol a případně i menu.

6.3.10 Instalace

Pro instalaci systému je nutné mít nejprve nainstalováno webový server Apache, PHP a databázový server MySQL. Všechny soubory musí být umístěny v adresáři webového serveru. Instalace databáze je možná pomocí SQL skriptu Elearning.sql. Ten vytvoří tabulku uživatelů a naplní ji údaji o uživateli. Dále je nutné nastavit parametry pro připojení k databázi v souboru connect.php. Po provedení všech nastavení je systém nainstalován a dostupný po zadání URL do okna prohlížeče. Vstupní stránka aplikace se nazývá index.php.

Pro otestování funkčnosti systému je možné využít instalaci na webovém serveru. Adresa www stránek je http://ocl.wz.cz a přihlašovací údaje jsou následující. Login je ***student*** a heslo také ***student***. Poslední kontrola funkčnosti byla provedena 13.5.2007.

Jelikož jsem vytvářel systém, podporující 2 jazykové verze, obsahuje tento systém všechny komponenty pro každou z jazykových verzí. Proto je nutné při instalaci nezapomenout, že existují 2 verze a každá z verzí obsahuje podobné soubory. Každá verze má svůj adresář, ve kterém se nachází

další podadresáře, jež obsahují všechny potřebné soubory pro danou verzi (obrázky, texty, ikony, testovací část, systém přihlašování) .

Adresářová struktura systému

[\]		
	Index.php	- Spouštěcí soubor – rozcestník umožňující spuštění cz nebo en verze.
[\cz]		- Adresář české verze - obsahuje ikony, soubor pro kontrolu přihlášení, úvodní stránku české verze, soubor obsahující styly.
	[\cz\inc\]	- Obsahuje soubor menu.php, což je soubor pro české menu.
	[\cz\inc\contents]	- Obsahuje všechny obrázky a texty pro českou verzi, včetně testů.
	[\cz\include]	- Obsahuje soubor connect.php zajišťující připojení k databázi.
	[\cz\sql]	- Obsahuje elearning.sql skript pro vytvoření databáze.
[\en]		- Struktura a obsah anglické verze je stejná jako struktura české, tato složka obsahuje anglickou verzi systému včetně studijních materiálů.
	[\en\inc\]	
	[\en\inc\contents]	
	[\en\include]	
	[\en\sql]	

7 Závěr

V diplomové práci jsem se zabýval problematikou elektronického vzdělávání a jazykem pro specifikaci omezení objektů systému. Základem bylo prostudovat problematiku tvorby elektronických dokumentů vhodných pro samostatné studium a seznámit se s jazykem pro specifikace omezení objektů systému. Tyto znalosti jsem dále využil při tvorbě systému pro podporu e-learningu. Tento systém obsahuje studijní materiály zabývající se specifikací OCL UML 2.0.

Při návrhu systému jsem navázal na ročníkový projekt, ze kterého jsem přejal část materiálů. Cílem bylo navrhnout aplikaci, která by splňovala požadavky na sytém pro podporu e-learnigu. Aplikace rozděluje tuto specifikaci do několika kapitol. Uživatel pak může tyto kapitoly postupně procházet a seznamovat se tak s danou problematikou. Na konci každé kapitoly je testová část, která uživateli umožní prověřování znalostí a sledování pokroku v pochopení prezentované látky.

Aby to bylo možné, musel jsem zpracovat kompletní specifikaci OCL UML 2.0 a uzpůsobit ji tak, aby ji bylo možné zakomplementovat do mnou navrženého systému. Specifikace obsahuje text popisující problematiku OCL, dále také příklady OCL výrazů a obrázky diagramů tříd, ke kterým se jednotlivé OCL výrazy vztahují. Všechny tyto části specifikace jsem musel upravit do formátu, umožňujícího zobrazení těchto materiálů pomocí systému pro podporu e-learningu. Vytvořil jsem graficky přehlednou aplikaci, do které se uživatel může přihlásit a poté studovat materiál obsahující specifikaci OCL. Aplikace také obsahuje diagramy a příklady OCL výrazů. Součástí aplikace je testová část, kde se na konci každé kapitoly nachází test s kontrolními otázkami, který po vyplnění vyhodnotí znalosti uživatele.

Aplikaci jsem implementoval v PHP, přihlašování je umožněno díky propojení na databázi MySQL a grafické rozhraní je vytvořeno pomocí HTML a CSS. Aplikace obsahuje 2 jazykové verze. Je proto vhodná pro ty studenty, kteří si mohou zvolit, zda daný předmět chtějí studovat česky nebo anglicky. Také je určena pro zahraniční studenty studující na Fakultě informačních technologií.

Sytém obsahuje všechny funkce, které byly obsaženy v návrhu a je plně funkční. Splňuje veškeré požadavky zadání. Jako možné rozšíření bych navrhoval optimalizaci systému pro všechny internetové prohlížeče, čímž by bylo dosaženo možnosti širšího využití. Jako další rozšíření bych navrhoval implementovat vyhledávácí funkci, která by umožňovala rychlejší orientaci ve specifikaci. Před využitím systému ve výuce bych doporučoval provést aktualizaci, jelikož v době dokončování systému byla vydána aktualizovaná verze OCL UML 2.0 specifikace.

Během vytváření diplomové práce jsem se obeznámil s jazykem pro specifikaci omezení objektů sytému a s problematikou tvorby e-learningových aplikací. Zjistil jsem jaká jsou úskalí, jaké výhody a nevýhody elektronického vzdělávání a vytvořil jsem aplikaci, při jejímž vytváření jsem využil těchto znalostí.

Literatura

[1] Apache HTTP Server Project, The Apache Software Foundation. Dokument dostupný na URL http://httpd.apache.org (květen 2007).

[2] Arlow J., Neustat I.:UML a unifikovaný proces vývoje aplikací, Computer Press, 2003, ISBN: 807226947X.

[3] Clark, R. C., Mayer R. E., E-Learning and science of instruction: proven guidelines for customers and designers of multimedia learning, John Wiley & Sons, Inc., 2003, ISBN: 0787960519.

[4] EDO E-learningové distanční opory, EDO project team. Dokument dostupný na URL http://edo.upol.cz/documents.php?sid=4835ad2a7ee2ffab974c3ea0a082f189&tid=elearning (květen 2007).

[5] Filip, A: OCL - Object Constraint Language Dokument dostupný na URL http://nb.vse.cz/~zelenyj/it380/eseje/xfila02/OCL.htm (květen 2007).

[6] History of e-Learning, KnowledgeNet.com, Inc. Dokument dostupný na URL http://knowledgenet.com/corporateinformation/ourhistory/history.jsp (květen 2007).

[7] Horton, W.: Designing Web-Based Training : How to Teach Anyone Anything Anywhere Anytime, Wiley; 1 edition, ISBN: 047135614X.

[8] Horton, W.: E-learning Tools and Technologies. John Wiley & Sons, Inc.; 1st edition (January 10, 2003), ISBN: 0471444588.

[9] Hulán, R: Instalace poslední verze Apache, MySQL a PHP na Windows. Dokument dostupný na URL http://radekhulan.cz/item/instalace-posledni-verze-apache-mysql-a-php-na-windows/ (květen 2007).

[10] Janovský, D: Jak psát web. Dokument dostupný na URL http://www.jakpsatweb.cz (květen 2007).

[11] Lacko, L.: PHP a MySQL, Hotová řešení, Computer Press, 2006, ISBN: 8025112497.

[12] Mach, J.: PHP pro úplné začátečníky, CP Books, a.s., 2005, ISBN: 8072268341.

[13] Molhanec, M.: UML - Unified Modeling Language. Dokument dostupný na URL http://www.osu.cz/katedry/kip/aktuality/sbornik99/mohlanec1.html (květen 2007).

[14] Neformální historie e-learningu, Škola Online. Dokument dostupný na URL http://www.skolaonline.cz/scripts/detail.php?id=4205 (květen 2007).

[15] Objektově orientovaná analýza a návrh systému. Dokument dostupný na URL http://agents.felk.cvut.cz/teaching/ppv/files/OOA.pdf (květen 2007).

[16] Palus: Objektově orientovaná analýza a návrh systému. Dokument dostupný na URL http://mpavus.wz.cz/index.php (květen 2007).

[17] Program Erasmus, Vysoké učení technické v Brně. Dokument dostupný na URL http://vvztahy.vutbr.cz/erasmus.php (květen 2007).

[18] Richta, K.: Unifikovaný modelovací jazyk UML. Dokument dostupný na URL http://si.vse.cz/archiv/clanky/2003/08_richta.pdf (květen 2007).

[19] Shmuller, J.: Myslíme v jazyku UML, knihovna programátora, Grada Publishing, spol. s r.o.,2001, ISBN: 8024700298.

[20] Stiller, E., LeBlanc, C.: Project-Based Software Engineering. An Object-Oriented Approach, Addison Wesley, 2002.

[21] UML 2.0 OCL Specification. Dokument dostupný na URL http://www.omg.org/docs/ptc/03-10-14.pdf (květen 2007).

[22] Wikipedie – Otevřená encyklopedie. Dokument dostupný na URL http://cs.wikipedia.org/wiki/ (květen 2007).

[23] Základy HTML a XHTML, Web tvorba – Dokument dostupný na URL http://www.webtvorba.cz/xhtml/zaklady-html.html (květen 2007).

[24] Zendulka, J.: Projektování programových systémů - 7 Jazyk UML (Unified Modeling Language), FIT VUT Brno. Dokument dostupný na URL http://www.fit.vutbr.cz/study/courses/PPS/public/pdf/7_umlch.pdf (květen 2007).

Seznam příloh

Příloha 1. CD-R:

Přiložené CD obsahuje:

- kompletní zdrojové kódy vytvořené aplikace (html, php, css)
- instalační SQL skript
- informační soubory install.txt a readme.txt
- text diplomové práce ve formátech .doc a .pdf

Adresářová struktura přiloženého CD:

[\]

readme.txt

install.txt

[\aplikace] - zdrojové soubory aplikace

[\diplomova_prace] - text diplomové práce

[\webovy_server] - instalační soubory nutné pro spuštění aplikace

[\zdorojove_kody] - kompletní zdrojové kódy zabaleny do jednoho souboru