

Na tomto místě bude oficiální zadání vaší práce asdffsdfsdfsdf

- Toto zadání je podepsané děkanem a vedoucím katedry,
- musíte si ho vyzvednout na studijním oddělení Katedry počítačů na Karlově náměstí,
- v jedné odevzdané práci bude originál tohoto zadání (originál zůstává po obhajobě na katedře),
- ve druhé bude na stejném místě neověřená kopie tohoto dokumentu (tato se vám vrátí po obhajobě).

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra počítačů



Bakalářská práce

Ruby - Access Control List

Jan Šírl

Vedoucí práce: Ing. Pavel Strnad

Studijní program: Softwarové technologie a management, Bakalářský

Obor: Softwarové inženýrství

21. března 2012

Poděkování

Zde můžete napsat své poděkování, pokud chcete a máte komu děkovat.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a použil jsem pouze podklady uvedené v příloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 18.3.2012

.....

Abstract

In this work is developed a proposal for a library / module source code in the Ruby programming language that addresses the management of access rights. The library / module is designed for object database Cellstore and is solved using the Access Control List.

Abstrakt

Tato práce prezentuje návrh a realizaci knihovny pro správu přístupových práv v programovacím jazyku Ruby.

Knihovna je určena pro objektovou databázi Cellstore a je řešena pomocí Access Control List. Knihovna využívá pro ukládání a dotazování samotnou databázi, se kterou komunikuje pomocí protokolu XML-RPC (Remote Procedure Call) technologií xQuery a xQuery Update Facility.

Očekávají se cca 1 – 2 odstavce, maximálně půl stránky.

Obsah

1	Úvod	1
2	Popis problému, specifikace cíle	3
2.1	Popis řešeného problému	3
2.2	Vymezení cílů	3
2.3	Existující řešení	3
2.3.1	Obecné řešení	3
2.3.2	Oracle	5
2.3.3	phpGACL	5
3	Analýza a návrh řešení	7
3.1	Analýza	7
3.2	Návrh implementace	7
3.2.1	Vstupy a výstupy	7
3.2.2	Ukázky použití	8
3.2.2.1	Příklad kontroly práv	8
3.2.2.2	Příklad nastavení práv	8
3.2.3	Use Case Scénáře	8
3.2.3.1	Ověřování oprávnění k objektu	8
3.2.3.2	Zadávání pravidla (ACE)	9
3.2.4	Use Case Diagram	9
3.2.5	Data flow diagram	9
3.2.6	Class diagram	9
3.2.7	Oprávnění	12
4	Realizace	13
5	Testování	15
6	Závěr	17
A	Testování zaplnění stránky a odsazení odstavců	21
B	Pokyny a návody k formátování textu práce	25
B.1	Vkládání obrázků	25
B.2	Kreslení obrázků	26

B.3	Tabulky	26
B.4	Odkazy v textu	27
B.4.1	Odkazy na literaturu	27
B.4.2	Odkazy na obrázky, tabulky a kapitoly	29
B.5	Rovnice, centrovaná, číslovaná matematika	29
B.6	Kódy programu	30
B.7	Další poznámky	30
B.7.1	České uvozovky	30
C	Seznam použitých zkratk	31
D	UML diagramy	33
E	Instalační a uživatelská příručka	35
F	Obsah přiloženého CD	37

Seznam obrázků

3.1	UseCases	10
3.2	Data flow diagram	11
3.3	Class diagram	12
B.1	Popiska obrázku	26
F.1	Seznam přiloženého CD — příklad	37

Seznam tabulek

2.1	Tabulka Funkčních a nefunkčních požadavků. Note: SRS priority= (must have, should have, nice to have, won't have)	4
2.2	Tabulkové řešení	4
B.1	Ukázka tabulky	26

Kapitola 1

Úvod

Tato bakalářská práce navazuje na můj semestrální projekt. Jejím úkolem bylo vytvořit, navrhnout a realizovat v jazyce Ruby model uživatelských přístupových práv určenou pro objektovou databázi Cellstore. V současnosti neexistuje žádný model uživatelských přístupových práv pro Cellstore.

Součástí práce byl návrh knihovny. Navržený modul realizuje správu řízení přístupu pomocí ACL (Access Control List). Protože moje bakalářská práce je prací implementační, včetně testů navrženého modulu, zaměřil jsem se na specifikaci rozhraní knihovny a na příklady jejího použití.

Výsledkem bakalářské práce je nejen samotná realizace knihovny, ale i podrobná programátorská dokumentace.

Kapitola 2

Popis problému, specifikace cíle

- Popis řešeného problému, vymezení cílů DP/BP a požadavků na implementovaný systém.
- Popis struktury DP/BP ve vztahu k vytyčeným cílům.
- Rešeršní zpracování existujících implementací, pokud jsou známy.

2.1 Popis řešeného problému

Ruby ACL je knihovna spravující přístupová práva pro databáze. Je určena pro objektovou databázi Cellstore. Cellstore v současné době nemá žádný model uživatelských přístupových práv. Hlavním cílem projektu je vytvoření jemně nastavitelných („fine-grained“) řízení přístupu. Toho se docílí pomocí Access Control Listu. ACL obsahuje seznam pravidel jednotlivých přístupů. Pravidlo se nazývá Access Control Entry (ACE). V ACE je uloženo Kdo, nebo Co má jaká Práva přistupovat k jakým Objektům. Těmto třem rozměrům se v problematice přístupových práv říká: Principals, Privileges, Resource Objects.

2.2 Vymezení cílů

Seznam požadavků je popsán v tabulce 2.1.

2.3 Existující řešení

Při řešení vlastního návrhu na model uživatelských přístupových práv jsem vycházel ze dvou zdrojů – známých řešení a jednoho obecného řešení.

2.3.1 Obecné řešení

Obecným řešením je držet si tabulku, kde ve sloupcích budou objekty, ke kterým je možno přistupovat a v řádcích jsou přistupující. V poli pak jsou hodnoty boolean, které vyjadřují buď allow nebo deny. Příklad tabulkového řešení je v tabulce 2.1.

SRS id	SRS description	SRS priority
FUNCTIONAL REQUIREMENTS		
0	Ruby ACL is library for Ruby programming language. Ruby ACL will manage control of access by ACL (Access Control List).	must have
1.0	Ruby ACL will allow to define Principals(Groups, Users, Processes)	must have
1.1	Ruby ACL will allow to define Privileges	must have
1.2	Ruby ACL will allow to define Resource objects	must have
1.3	Ruby ACL will allow to edit or delete Principals(Groups, Users, Processes)	must have
1.4	Ruby ACL will allow to edit or delete Privileges	must have
1.5	Ruby ACL will allow to edit or delete Resource objects	must have
1.6	Ruby ACL will allow to define,edit or delete access control rule	must have
2.0	Ruby ACL will allow to create, load and save ACL into XML file	must have
2.1	XML file will be defined by DTD or XML Schema	must have
2.2	XML file will be well formatted according to W3C recommendations	must have
3.0	Ruby ACL will offer Default-Deny policy (At the beginning nobody cannot access to anywhere) or Default-Allow policy (At the beginning everybody can access to everywhere)	must have
4.0	Ruby ACL will use XML-RPC interface to communicate with database	must have
5.0	Ruby ACL will be tested on eXist DB	must have
NON-FUNCTIONAL REQUIREMENTS		
1.0	Ruby ACL will be programmed in Ruby	must have
2.0	Ruby ACL will be released as RubyGem	must have
3.0	Ruby ACL will have visual representation	won't have
4.0	Ruby ACL will need database with XML-RPC interface	must have

Tabulka 2.1: Tabulka Funkčních a nefunkčních požadavků. Note: SRS priority= (must have, should have, nice to have, won't have)

Who / Where	Surgeryroom	ambulance	Patient's room
Doctors	1	1	1
Nurses	X	1	1
Patients	X	X	1

Tabulka 2.2: Tabulkové řešení

2.3.2 Oracle

Prvním z konkurenčních řešení je řešení prezentované v Oracle® XML DB Developer's Guide, 11g Release 1 (11.1), Part Number B28369-04 na http://docs.oracle.com/cd/B28359_01/appdev.111/b28369/xdb21sec.htm. Ve stati Access Control Lists and Security Classes je popsán koncept firmy ORACLE.

Text popisuje několik podmínek a pojetí řízení přístupu. Každá z popsaných entit, uživatel, role, privilegia, bezpečnostní třídy, Access Control List (ACL) a Access Control Entry (ACE), je realizována deklarativně jako XML dokument nebo fragment.

Bezpečnostní autorizace vyžaduje definovat, kteří uživatelé, aplikace nebo funkce mohou mít přístup k jakým datům nebo jaké druhy operací mohou provádět. Existují tedy tři dimenze: (1) kteří uživatelé mohou, (2) vykonávat jaké činnosti, (3) na jakých datech. V souvislosti s každou jednotlivou dimenzí hovoříme o (1) principals - zmocnitelích, (2) privileges - oprávněných, a (3) objektech, které korespondují s těmito třemi dimenzemi. Principals mohou být uživatelé nebo role/skupiny.

Principals a privileges (dimenze 1 a 2) jsou deklarativním způsobem spojeni v definovaných seznamech řízení přístupu - ACL. Ty jsou pak spojené s třetí dimenzí - daty, různými způsoby. Například úložiště zdrojů nebo tabulky dat Oracle XML DB mohou být ochráněny pomocí PL / SQL procedury DBMS_XDB.setACL nastavením jeho řídicího ACL.

2.3.3 phpGACL

Druhým ze zdrojů, z nichž jsem vycházel, je řešení prezentované v Generic Access Control List with PHP - phpGACL na phpgac1.sourceforge.net/manual.pdf.

Nástroj phpGACL je sada funkcí, která umožňuje použít řízení přístupu na libovolné objekty (webové stránky, databáze, atd.), jiným libovolným objektům (uživatelé, vzdálené počítače, atd.). Stejně jako Oracle nabízí jemně nastavitelnou kontrolu přístupu s jednoduchou a velmi rychlou správou. Je napsán v populárním dynamickém skriptovacím jazyku PHP.

Nástroj phpGACL vyžaduje relační databáze pro ukládání informací k řízení přístupu. Přistupuje k databázi prostřednictvím tzv. abstraktního obalu ADOdb. Je kompatibilní s databázemi, jako PostgreSQL, MySQL a Oracle.

Nástroj phpGACL používá pojmy jako ACO a ARO:

- Access Control Objects (ACO), jsou věci, ke kterým chceme ovládat přístup (např. webové stránky, databáze, pokoje, atd.).
- Access Request Objects (ARO), jsou věci, které žádají o přístup (např. osoby, vzdálené počítače, atd.)
- ARO stromy definují hierarchii skupin a ARO. Skupiny mohou obsahovat jiné skupiny a ARO.
- Výchozí "catch-all" politikou stromu ARO je vždy "DENY ALL".
- Chceme-li přiřadit přístupovou politiku ve stromu směrem dolů, explicitně přiřazujeme oprávnění skupinám a ARO pro každou ACO, pro kterou je potřeba.

Kapitola 3

Analýza a návrh řešení

Analýza a návrh implementace (včetně diskuse různých alternativ a volby implementačního prostředí).

3.1 Analýza

???

3.2 Návrh implementace

Nechal jsem se inspirovat jak Oraclém tak phpGACL. Oba modely řízení přístupových práv mají podobnou strukturu nebo stejnou s jiným pojmenováním. Z Oraclu jsem převzal pojmenování dimenzí: Principals, Privileges, Objects, ze kterých jsem vytvořil hlavní třídy. Ruby ACL je vydávána ve formě balíčkovacího systému RubyGem.

K určení operace a objektu z SQL bude použit SQL parser třetí strany.

3.2.1 Vstupy a výstupy

Vstupy budou parametry:

1. Principals - který přistupující
2. Privileges - může/nesmí provádět jaké operace
3. Objects - na jakých datech

Výstup: Allow/Deny specifikováno v boolean

3.2.2 Ukázky použití

3.2.2.1 Příklad kontroly práv

```
...DB_Application...
```

```
require 'Ruby-ACL'
require 'dbi'
acl = Racl.new()
acl.load("test1")
username = "pepanovak"
password = "tajneheslo"
desired_operation = "select"
desired_object = "dbi:OCIU:mydb/people"
if(acl.acl_check(username, desired_operation, desired_object)) then
db = DBI.connect (separate_db(desired_object), username, password)
query = "select * from people"
stmt = db.prepare(query)
stmt.execute
while row = stmt.fetch do
puts row.join(",")
end
stmt.finish
db.disconnect
else
puts "Access denied to #{desired_object}."
end
```

3.2.2.2 Příklad nastavení práv

```
...DB_Application...
require 'Ruby-ACL'
username = "pepanovak"
access_type = "deny"
desired_privilege = "create"
desired_object = "dbi:OCIU:mydb"
acl = Racl.new("test2")
acl.init_from_db("dbi:OCIU:mydb", "tabulka_Useru_a_Skupin")
acl.set_new_ace(username, access_type, desired_privilege, desired_object)
```

`init_from_db` načte všechny objekty z databáze. Tj. Principals a ResourceObject.

3.2.3 Use Case Scénáře

3.2.3.1 Ověřování oprávnění k objektu

Uživatel má vytvořenou instanci RubyACL, která obsahuje pravidla. Hlavní úspěšný scénář:

1. Uživatel zavolá metodu `acl_check`. Přes tuto metodu se dotázá systému, jestli uživatel/skupina (ne)mají oprávnění ke zdrojovému objektu.
2. a) Systém vrátí `true` v případě, že uživatel/skupina má specifikované nebo vyšší oprávnění.
3. b) Systém vrátí `false` v případě, že uživatel/skupina nemají specifikované nebo vyšší oprávnění.

Rozšíření:

0a) Pokud neexistují pravidla v instanci, systém vrátí `false`, protože nenašel, žádné vyhovující pravidlo.

3.2.3.2 Zadávání pravidla (ACE)

Uživatel má vytvořenou instanci `RubyACL`, která obsahuje pravidla. Hlavní úspěšný scénář:

1. Uživatel zavolá metodu `set_new_ace` a specifikuje údaje (Uživatel/skupina, typ přístupy (`allow/deny`), oprávnění, zdrojový objekt
2. Systém nastaví pravidlo a vrátí 0, když vše proběhlo v pořádku, a 1 když nastala chyba.

Poznámka: Jde vlastně o přiřazování oprávnění uživatelům na objekt

3.2.4 Use Case Diagram

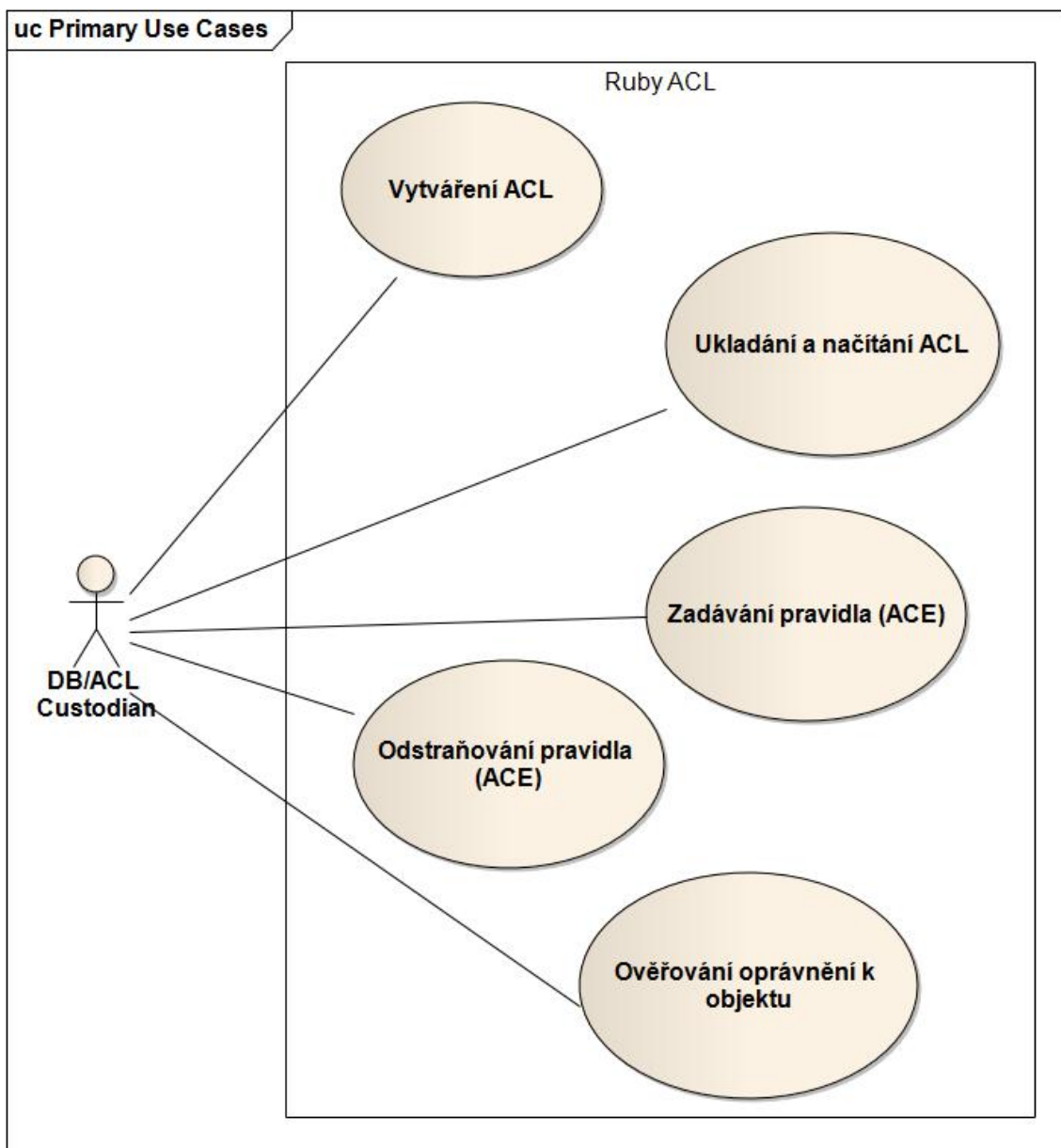
Znázorňuje roli uživatele vůči knihovně. `Ruby ACL` definuje jednoho aktéra, kterým je uživatel/administrátor `ACL` viz "Obrázek 1". Jedná se vlastně o vývojáře `DB Aplikace`. Všechny případy užití předpokládají vytvořenou instanci `RubyACL`, která má vytvořená nějaká pravidla.

3.2.5 Data flow diagram

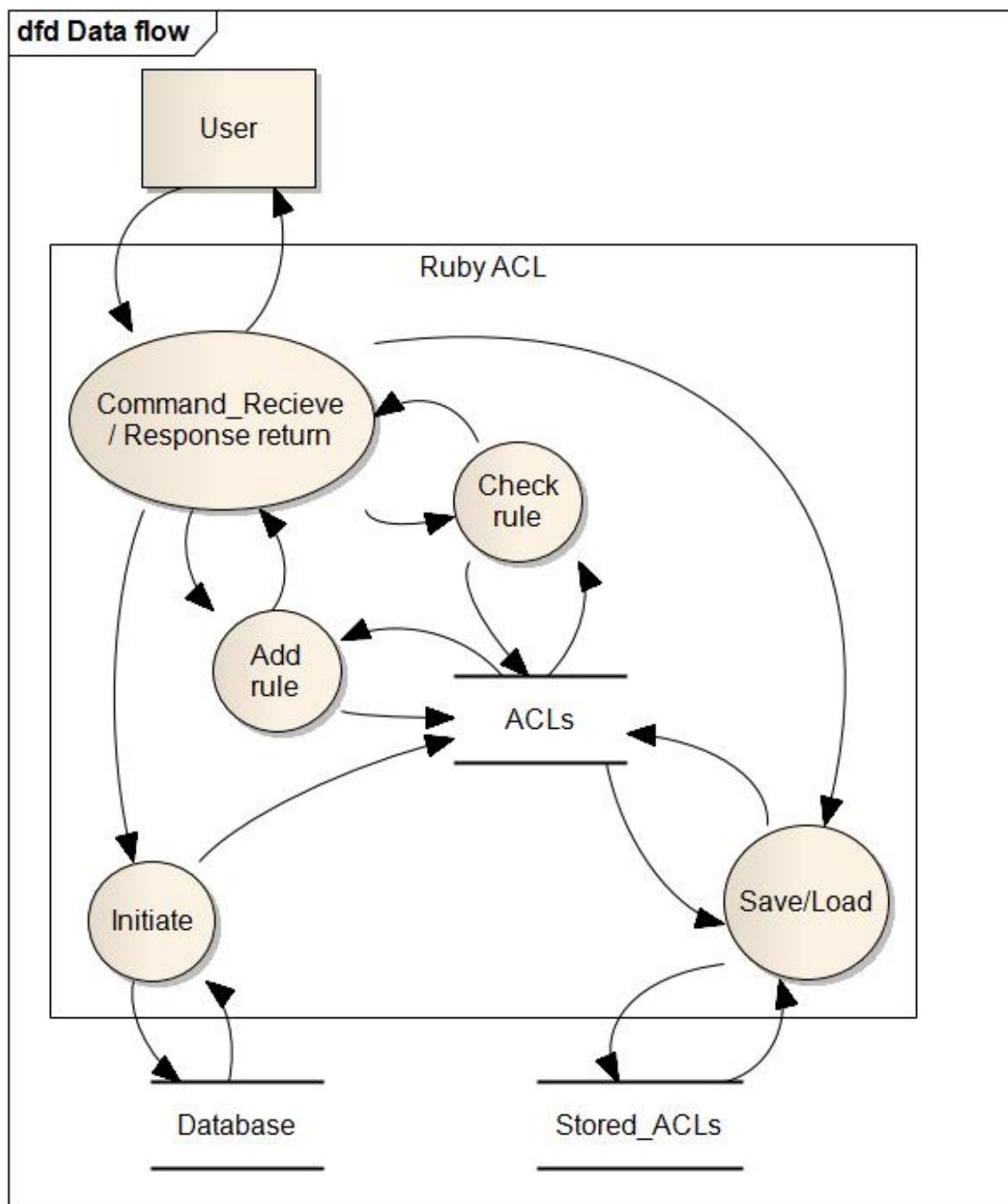
Data flow diagram (viz "Obrázek 2") znázorňuje tok dat mezi jednotlivými funkcemi aplikace. Popisuje funkce a jejich vazby. Uložiště pro `ACL` bude používaná databáze. Pro každou databázi bude jedna instance `Ruby-ACL`.

3.2.6 Class diagram

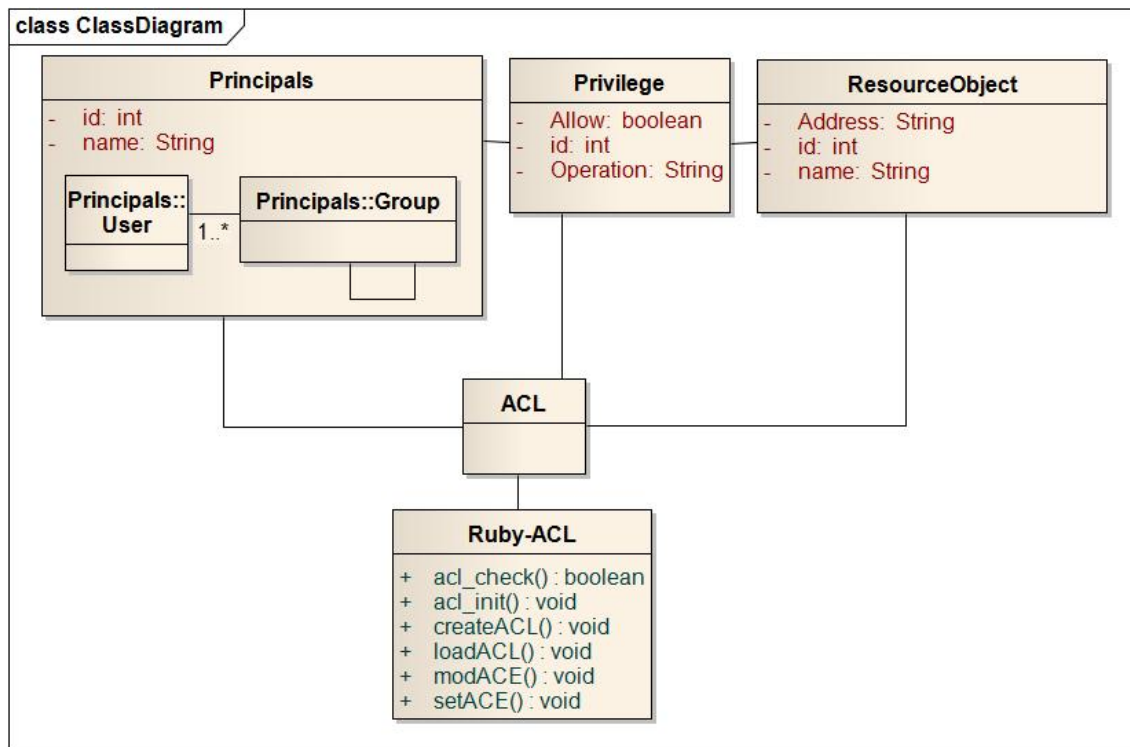
Class diagram (viz "Obrázek 3") znázorňuje základní třídy aplikace a jejich vazby.



Obrázek 3.1: UseCases



Obrázek 3.2: Data flow diagram



Obrázek 3.3: Class diagram

3.2.7 Oprávnění

Ruby-ACL nabídne vlastní přidané privilegia a základní sadu privilegií a převzatých z Oracle potažmo MySQL. Jedná se o privilegia 'ALL PRIVILEGES', 'ALTER', 'CREATE', 'DELETE', 'DROP', 'FILE', 'INDEX', 'INSERT', 'PROCESS', 'REFERENCES', 'RELOAD', 'SELECT', 'SHUTDOWN', 'UPDATE' a 'USAGE'.

Kapitola 4

Realizace

Popis implementace/realizace se zaměřením na nestandardní části řešení.

- Ušetření času db serveru vhodným uspořádáním dotazů. Místo iterace dotazů jeden dotaz s vhodnou podmínkou. (tvoreni clenstvi)
- Použití xQuery Update Facility místo xUpdate. (Možná vložím ukázkou implementace)
- Pro identifikaci a propojení jsem uvažoval mezi xLink a idref. Idref nabízí jednoduchý systém, ale xLink nabídl podrobné specifikace W3C, velké množství návodů a turtoriálů a především se jedná o novější technologii, jejíž osvojení jsem považoval za výhodné. Rozhodl jsem se proto implementovat xLink. Problém nastal při některých vkládání textů a dotazování. EXist DB měla problém s jmeným prostorem xLink i přes skutečnost, že jmený prostor byl uveden. Troufám si říct, že byl uveden správně, protože při většině použití fungoval. Než abych se zabýval mnoho času proč použití xLink nejde, raději jsem přešel na jednoduchý idref. Implementace idref proběhla bez problémů.
- Občas je kód zdvojený, ale má to svoje opodstatnění. Například addmembership a addmembershippriv. Kdyby nebylo zdvojeno, šlo by přidávat privilegie k principal a naopak.
- Je třeba se věnovat uvolňování paměti, protože aplikace bude spuštěna dny až měsíce. Špatné zacházení s pamětí by, proto bylo kritické pro server, na kterém aplikace s Ruby-ACL knihovnou bude spuštěna.

Kapitola 5

Testování

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Kapitola 6

Závěr

- Zhodnocení splnění cílů DP/BP a vlastního přínosu práce (při formulaci je třeba vzít v potaz zadání práce).
- Diskuse dalšího možného pokračování práce.

Literatura

- [1] HAINDL, M. – KMENT, Ľ. – SLAVÍK, P. Virtual Information Systems. In *WSCG'2000 — Short communication papers*, s. 22–27. University of West Bohemia, Pilsen, 2000.
- [2] Příspěvatelé Wikipedie. *Framework* [online]. 2009. [cit. 10. 9. 2009]. Dostupné z: <<http://cs.wikipedia.org/wiki/Framework>>.
- [3] Příspěvatelé Wikipedie. *Object-relational mapping* [online]. 2009. [cit. 6. 12. 2009]. Dostupné z: <http://en.wikipedia.org/wiki/Object-relational_mapping>.
- [4] ŽÁRA, J. – BENEŠ, B. – FELKEL, P. *Moderní počítačová grafika*. Computer Press s.r.o, Brno, 1st edition, 1998. In Czech.
- [5] SLAVÍK, P. Grammars and Rewriting Systems as Models for Graphical User Interfaces. *Cognitive Systems*. 1997, 4, 3/4, s. 381–399.
- [6] web:cstug. CSTUG — \LaTeX Users Group — hlavní stránka. <http://www.cstug.cz/>, stav z 2. 3. 2009.
- [7] web:infodp. K336 Info — pokyny pro psaní diplomových prací. <https://info336.felk.cvut.cz/clanek.php?id=400>, stav ze 4. 5. 2009.
- [8] web:infogs. Knihovna Grafické skupiny. <http://www.cgg.cvut.cz/Bib/library/>, stav z 30. 8. 2001.
- [9] web:ipe. Grafický vektorový editor pro práce vhodný pro práci \LaTeX em. <http://tclab.kaist.ac.kr/ipe/>, stav z 4. 5. 2009.
- [10] web:latexdocweb. \LaTeX — online manuál. <http://www.cstug.cz/latex/lm/frames.html>, stav ze 4. 5. 2009.
- [11] web:latexwiki. Wiki Books \LaTeX . <http://en.wikibooks.org/wiki/LaTeX/>, stav z 3. 4. 2009.

Příloha A

Testování zaplnění stránky a odsazení odstavců

Tato příloha nebude součástí vaší práce. Slouží pouze jako příklad formátování textu.

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili? Určitě existuje nějaká pěkná latinská věta, která se k tomuhle testování používá, ale co mají dělat ti, kteří se nikdy latinsky neučili?

Příloha B

Pokyny a návody k formátování textu práce

Tato příloha samozřejmě nebude součástí vaší práce. Slouží pouze jako příklad formátování textu.

Používat se dají všechny příkazy systému L^AT_EX. Existuje velké množství volně přístupné dokumentace, tutoriálů, příruček a dalších materiálů v elektronické podobě. Výchozím bodem, kromě Googlu, může být stránka CSTUG (Czech Tech Users Group) [6]. Tam najdete odkazy na další materiály. Většinou dostačující a přehledně organizovanou elektronikou dokumentaci najdete například na [10] nebo [11].

Existují i různé nadstavby nad systémy T_EX a L^AT_EX, které výrazně usnadní psaní textu zejména začátečníkům. Velmi rozšířený v Linuxovém prostředí je systém Kile.

B.1 Vkládání obrázků

Obrázky se umísťují do plovoucího prostředí **figure**. Každý obrázek by měl obsahovat **název** (`\caption`) a **návěští** (`\label`). Použití příkazu pro vložení obrázku `\includegraphics` je podmíněno aktivací (načtením) balíku `graphicx` příkazem `\usepackage{graphicx}`.

Budete-li zdrojový text zpracovávat pomocí programu `pdflatex`, očekávají se obrázky s příponou `*.pdf`¹, použijete-li k formátování `latex`, očekávají se obrázky s příponou `*.eps`.²

Příklad vložení obrázku:

```
\begin{figure}[h]
\begin{center}
\includegraphics[width=5cm]{figures/LogoCVUT}
\caption{Popiska obrazku}
\label{fig:logo}
```

¹pdflatex umí také formáty PNG a JPG.

²Vzájemnou konverzi mezi snad všemi typy obrázku včetně změn velikostí a dalších vymožeností vám může zajistit balík ImageMagick (<http://www.imagemagick.org/script/index.php>). Je dostupný pod Linuxem, Mac OS i MS Windows. Důležité jsou zejména příkazy `convert` a `identify`.



Obrázek B.1: Popiska obrázku

DTD	construction	elimination
	in1 A B a:sum A B in1 A B b:sum A B	case([_:A]a)([_:B]a)ab:A case([_:A]b)([_:B]b)ba:B
+	do_reg:A -> reg A	undo_reg:reg A -> A
*, ?	the same like and + with empty_el:empty	the same like and + with empty_el:empty
R(a,b)	make_R:A->B->R	a: R -> A b: R -> B

Tabulka B.1: Ukázka tabulky

```
\end{center}
\end{figure}
```

B.2 Kreslení obrázků

Zřejmě každý z vás má nějaký oblíbený nástroj pro tvorbu obrázků. Jde jen o to, abyste dokázali obrázek uložit v požadovaném formátu nebo jej do něj konvertovat (viz předchozí kapitola). Je zřejmě vhodné kreslit obrázky vektorově. Celkem oblíbený, na ovládání celkem jednoduchý a přitom dostatečně mocný je například program Inkscape.

Zde stojí za to upozornit na kreslicí programe Ipe [9], který dokáže do obrázku vkládat komentáře přímo v latexovském formátu (vzroce, stejné fonty atd.). Podobné věci umí na Linuxové platformě nástroj Xfig.

Za pozornost ještě stojí schopnost editoru Ipe importovat obrázek (jpg nebo bitmap) a krelit do něj latexovské popisky a komentáře. Výsledek pak umí exportovat přímo do pdf.

B.3 Tabulky

Existuje více způsobů, jak sázet tabulky. Například je možno použít prostředí `table`, které je velmi podobné prostředí `figure`.

Zdrojový text tabulky B.1 vypadá takto:

```

\begin{table}
\begin{center}
\begin{tabular}{|c|l|l|l|}
\hline
\textbf{DTD} & \textbf{construction} & \textbf{elimination} & \\
\hline
 $\mid$  &  $\verb+in1|A|B$  a:sum A B+ &  $\verb+case([_:A]a)([_:B]a)ab:A+\backslash\backslash$  & 
 $\verb+in1|A|B$  b:sum A B+ &  $\verb+case([_:A]b)([_:B]b)ba:B+\backslash\backslash$  \\
\hline
 $\&\&$  &  $\verb+do\_reg:A \rightarrow reg A+\&\verb+undo\_reg:reg A \rightarrow A+\backslash\backslash$  \\
\hline
 $\ast,?\$$  & the same like  $\mid$  and  $\&\&$  & the same like  $\mid$  and  $\&\&\backslash\backslash$  & 
with  $\verb+empty\_el:empty+$  & with  $\verb+empty\_el:empty+\backslash\backslash$  \\
\hline
 $R(a,b)$  &  $\verb+make\_R:A\rightarrow B\rightarrow R+$  &  $\verb+a: R \rightarrow A+\backslash\backslash$  & 
 $\&\verb+b: R \rightarrow B+\backslash\backslash$  \\
\hline
\end{tabular}
\end{center}
\caption{Ukázka tabulky}
\label{tab:tab1}
\end{table}
\begin{table}

```

B.4 Odkazy v textu

B.4.1 Odkazy na literaturu

Jsou realizovány příkazem `\cite{odkaz}`.

Seznam literatury je dobré zapsat do samostatného souboru a ten pak zpracovat programem bibtex (viz soubor `reference.bib`). Zdrojový soubor pro bibtex vypadá například takto:

```

@Article{Chen01,
  author   = "Yong-Sheng Chen and Yi-Ping Hung and Chiou-Shann Fuh",
  title    = "Fast Block Matching Algorithm Based on
              the Winner-Update Strategy",
  journal  = "IEEE Transactions On Image Processing",
  pages    = "1212--1222",
  volume   = 10,
  number   = 8,
  year     = 2001,
}

```

```

@Misc{latexdocweb,

```

```

author = "",
title = "{\LaTeX} --- online manuál",
note = "\verb|http://www.cstug.cz/latex/lm/frames.html|",
year = "",
}
...

```

Pozor: Sazba názvů odkazů je dána BibTeX stylem (`\bibliographystyle{abbrv}`). BibTeX tedy obvykle vysází velké pouze počáteční písmeno z názvu zdroje, ostatní písmena zůstanou malá bez ohledu na to, jak je napíšete. Přesněji řečeno, styl může zvolit pro každý typ publikace jiné konverze. Pro časopisecké články třeba výše uvedené, jiné pro monografie (u nich často bývá naopak velikost písmen zachována).

Pokud chcete BibTeXu napovědět, která písmena nechat bez konverzí (viz `title = "{\LaTeX} --- online manuál"` v předchozím příkladu), je nutné příslušné písmeno (zde celé makro) uzavřít do složených závorek. Pro přehlednost je proto vhodné celé parametry uzavírat do uvozovek (`author = "..."`), nikoliv do složených závorek.

Odkazy na literaturu ve zdrojovém textu se pak zapisují:

```

Podívejte se na \cite{Chen01},
další detaily najdete na \cite{latexdocweb}

```

Vazbu mezi soubory `*.tex` a `*.bib` zajistíte příkazem `\bibliography{}` v souboru `*.tex`. V našem případě tedy zdrojový dokument `thesis.tex` obsahuje příkaz `\bibliography{reference}`.

Zpracování zdrojového textu s odkazy se provede postupným voláním programů `pdflatex <soubor>` (případně `latex <soubor>`), `bibtex <soubor>` a opět `pdflatex <soubor>`.³

Níže uvedený příklad je převzat z dříve existujících pokynů studentům, kteří dělají svou diplomovou nebo bakalářskou práci v Grafické skupině.⁴ Zde se praví:

```

...
j) Seznam literatury a dalších použitých pramenů, odkazy na WWW stránky, ...
Pozor na to, že na veškeré uvedené prameny se musíte v textu práce
odkazovat -- [1].
Pramen, na který neodkazujete, vypadá, že jste ho vlastně nepotřebovali
a je uveden jen do počtu. Příklad citace knihy [1], článku v časopise [2],
statí ve sborníku [3] a html odkazu [4]:
[1] J. Žára, B. Beneš;, and P. Felkel.
    Moderní počítačová grafika. Computer Press s.r.o, Brno, 1 edition, 1998.
    (in Czech).

```

³První volání `pdflatex` vytvoří soubor s koncovkou `*.aux`, který je vstupem pro program `bibtex`, pak je potřeba znovu zavolat program `pdflatex` (`latex`), který tentokrát zpracuje soubory s příponami `.aux` a `.tex`. Informaci o případných nevyřešených odkazech (cross-reference) vidíte přímo při zpracovávání zdrojového souboru příkazem `pdflatex`. Program `pdflatex` (`latex`) lze volat vícekrát, pokud stále vidíte nevyřešené závislosti.

⁴Několikrát jsem byl upozorněn, že web s těmito pokyny byl zrušen, proto jej zde přímo necituji. Nicméně příklad sám o sobě dokumentuje obecně přijímaný konsensus ohledně citací v bakalářských a diplomových pracích na KP.

- [2] P. Slavík. Grammars and Rewriting Systems as Models for Graphical User Interfaces. *Cognitive Systems*, 4(4--3):381--399, 1997.
- [3] M. Haindl, Š. Kment, and P. Slavík. Virtual Information Systems. In *WSCG'2000 -- Short communication papers*, pages 22--27, Pilsen, 2000. University of West Bohemia.
- [4] Knihovna grafické skupiny katedry počítačů:
<http://www.cgg.cvut.cz/Bib/library/>

... abychom výše citované odkazy skutečně našli v (automaticky generovaném) seznamu literatury tohoto textu, musíme je nyní alespoň jednou citovat: Kniha [4], článek v časopisu [5], příspěvek na konferenci [1], [www odkaz](#) [8].

Ještě přidáme další ukázkou citací online zdrojů podle české normy. Odkaz na wiki o frameworkch [2] a ORM [3]. Použití viz soubor `reference.bib`. V seznamu literatury by nyní měly být živé odkazy na zdroje. V `reference.bib` je zcela nový typ publikace. Detaily dohledal a dodal Petr Dlouhý v dubnu 2010. Podrobnosti najdete ve zdrojovém souboru tohoto textu v komentáři u příkazu `\thebibliography`.

B.4.2 Odkazy na obrázky, tabulky a kapitoly

- Označení místa v textu, na které chcete později čtenáře práce odkázat, se provede příkazem `\label{navesti}`. Lze použít v prostředích `figure` a `table`, ale též za názvem kapitoly nebo podkapitoly.
- Na návěští se odkážeme příkazem `\ref{navesti}` nebo `\pageref{navesti}`.

B.5 Rovnice, centrováná, číslovaná matematika

Jednoduchý matematický výraz zapsaný přímo do textu se vysází pomocí prostředí `math`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$`.

Kód `$ S = \pi * r^2 $` bude vysázen takto: $S = \pi * r^2$.

Pokud chcete nečíslované rovnice, ale umístěné centrováně na samostatné řádky, pak lze použít prostředí `displaymath`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$$`. Zdrojový kód: `$$$ S = \pi * r^2 $$$` bude pak vysázen takto:

$$S = \pi * r^2$$

Chcete-li mít rovnice číslované, je třeba použít prostředí `equation`. Kód:

```
\begin{equation}
  S = \pi * r^2
\end{equation}
```

```
\begin{equation}
  V = \pi * r^3
\end{equation}
```

je potom vysázen takto:

$$S = \pi * r^2 \quad (\text{B.1})$$

$$V = \pi * r^3 \quad (\text{B.2})$$

B.6 Kódy programu

Chceme-li vysázet například část zdrojového kódu programu (bez formátování), hodí se prostředí *verbatim*:

```
(* nickname2 *)
Lego> Refine in1
      (do_reg (nickname1 h));
Refine by in1 (do_reg (nickname1 h))
  ?4 : pcddata
  ?5 : pcddata
      (* surname2 *)
Lego> Refine surname1 h;
Refine by surname1 h
  ?5 : pcddata
      (* email2 *)
Lego> Refine undo_reg (email1 h);
Refine by undo_reg (email1 h)
*** QED ***
```

B.7 Další poznámky

B.7.1 České uvozovky

V souboru `k336_thesis_macros.tex` je příkaz `\uv{}` pro sázení českých uvozovek. „Text uzavřený do českých uvozovek.“

Příloha C

Seznam použitých zkratek

2D Two-Dimensional

ABN Abstract Boolean Networks

ASIC Application-Specific Integrated Circuit

⋮

Příloha D

UML diagramy

Tato příloha není povinná a zřejmě se neobjeví v každé práci. Máte-li ale větší množství podobných diagramů popisujících systém, není nutné všechny umísťovat do hlavního textu, zvláště pokud by to snižovalo jeho čitelnost.

Příloha E

Instalační a uživatelská příručka

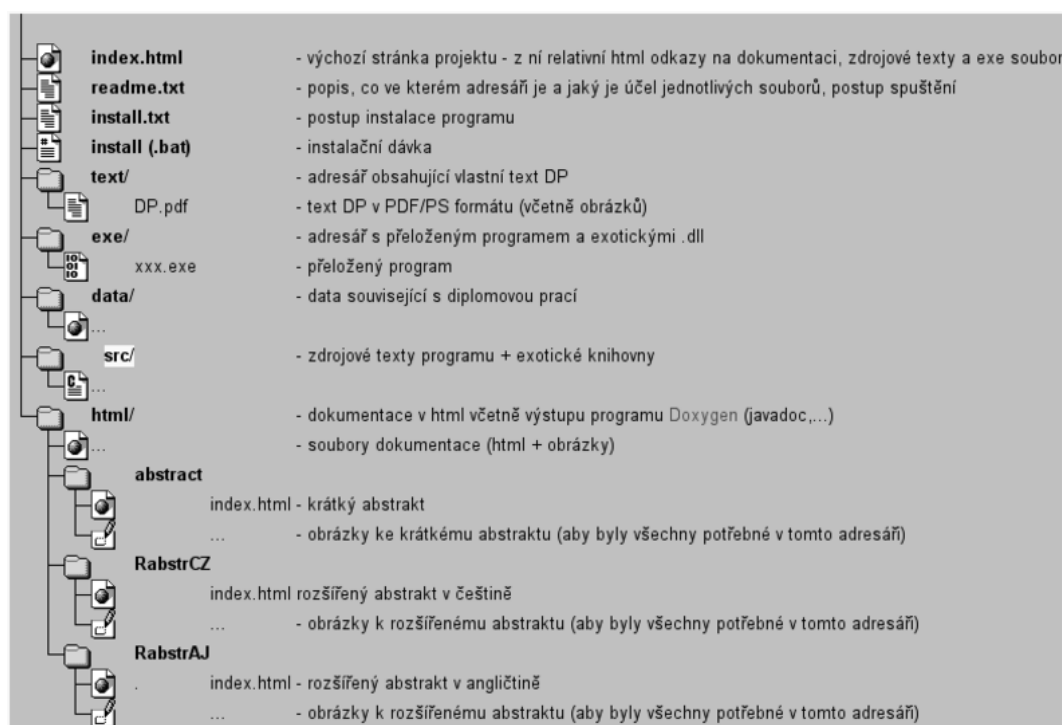
Tato příloha velmi žádoucí zejména u softwarových implementačních prací.

Příloha F

Obsah přiloženého CD

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce. (viz [7]):



Obrázek F.1: Seznam přiloženého CD — příklad

Na GNU/Linuxu si strukturu přiloženého CD můžete snadno vyrobit příkazem:

```
$ tree . >tree.txt
```

Ve vzniklém souboru pak stačí pouze doplnit komentáře.

Z **README.TXT** (případně index.html apod.) musí být rovněž zřejmé, jak programy instalovat, spouštět a jaké požadavky mají tyto programy na hardware.

Adresář **text** musí obsahovat soubor s vlastním textem práce v PDF nebo PS formátu, který bude později použit pro prezentaci diplomové práce na WWW.