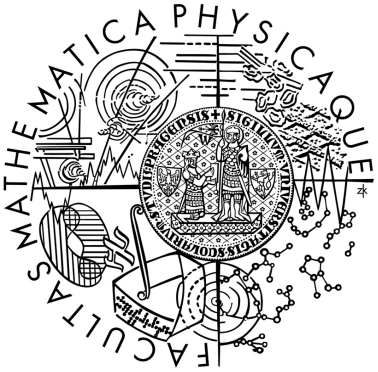
UNIVERZITA KARLOVA V PRAZE
MATEMATICKO-FYZIKÁLNÍ FAKULTA

DIPLOMOVÁ PRÁCE



David Bertoli

Kompresní korpus

Katedra softwarového inženýrství
Vedoucí diplomové práce: Mgr. Jan Lánský
Studijní program: Informatika, softwarové systémy

Rád by som poďakoval vedúcemu mojej diplomovej práce, Mgr. Janovi Lánskému, za cenné rady, konzultácie a námety, ktoré som vo svojej práci použil.
Podobne patrí moje poďakovanie mojím priateľom zo štúdia na MFF, v prvom rade Mgr. Katsiaryne Chernik, ktorá ma k tejto téme priviedla a Mgr. Tomášovi Magyarovi, ktorý mi pomáhal so štatistickou časťou práce.

Prehlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičiavaním práce.

V Prahe dňa 7.12.2008 David Bertoli

Obsah

Názov práce: Kompresný korpus
Autor: David Bertoli
Katedra: Katedra softwarového inženýrství
Vedúci diplomovej práce: Mgr. Jan Lánský
e-mail vedúceho: zizelevak@gmail.com

Abstrakt

Cieľom tejto práce je navrhnúť novú štruktúru korpusu, ktorého zdrojové súbory nebudú prístupné užívateľom, aby nedochádzalo k špecifickému vývoji kompresných algoritmov s cieľom dosiahnuť uspokojivé výsledky v meraných parametroch. Ďalším bodom požiadaviek je vytvorenie testovacej aplikácie, ktorá by bola schopná otestovať jednotlivé programy (algoritmy) nad vytvoreným korpusom, a zároveň by mala byť schopná uchovávať výsledky testov pre komplexné štatistické vyhodnotenie kvality otestovaných programov a ich následné sprístupnenie registrovaným užívateľom.

Kľúčové slová: korpus, kompresný program, štatistické vyhodnotenie

Title: Compression corpus
Author: David Bertoli
Department: Department of Software Engineering
Supervisor: Mgr. Jan Lánský
Supervisor's e-mail address: zizelevak@gmail.com

Abstrakt

The goal of this thesis is to design new structure of corpus, whose data won't be accessible for users, so that users can't specifically develop their compression algorithms focused on reaching good results in measured parameters. Next point of requirements is to develop test application, which will be able to test programs (algorithms) using designed corpus and store results of tests for creating complex statistic feedback of tested programs. Feedback will be accessible for signed users.

Keywords: corpus, compression program, statistic feedback

Kapitola 1

Úvod

Doposiaľ bol vyvinutý veľký počet kompresných programov. Zo začiatkov vývoja kompresných programov sa účinnosť jednotlivých algoritmov merala odhadom autora, ktorý kompresný program navrhol. Týmto spôsobom nebolo možné jednoznačne určiť, ktorý program je najefektívnejší. Preto pre kvalitnejšie porovnávanie kompresných algoritmov bol navrhnutý Calgary corpus [1] textových súborov. Po čase sa tento korpus ukázal ako nevyhovujúci, najmä z dôvodu špeciálneho pripravovania algoritmov na tento korpus, tým pádom výsledky testovania nad Calgary korpusom boli irelevantné. Nedostatok Calgary korpusu mal riešiť Cantebury corpus [2]. Cantebury korpus obsahuje základ rozličných súborov, čím odstraňuje problém špeciálneho vývoja algoritmu na daný typ súborov. Cantebury corpus ale nerieši zvyšné problémy.

Oba korpusy obsahujú malé množstvo dát, rádovo jednotky súborov. Z výsledkov dvoch kompresných programov (algoritmov) nad týmito dvoma korpusmi ([1], [2]) niekedy nie je možné jednoznačne určiť, ktorý z týchto dvoch programov je celkovo lepší, pri porovnávaní výsledného kompresného pomeru a rýchlosti kompresie, z dôvodu malých odchýlok, ktoré sa objavia pri malom počte testovaných súborov. Taktiež každý kompresný algoritmus môže dosiahnuť lepšie výsledky nad určitou kategóriou dát a nad inou kategóriou nedosahuje očakávané výsledky. Pri takto malých počtoch súborov býva výsledok ovplyvnený náhodou.

Negatívne vlastnosti Calgary a Cantebury korpusov by odstránil korpus, obsahujúci veľké množstvo, rádovo tisícok až desaťtisícov súborov, rozčlenených hierarchicky do rôznych kategórii a subkategórií. Úloha náhody by pre dva programy porovnávané nad podobným mnoho-súborovým bola zanedbateľná a tým pádom výsledky dosiahnuté nad týmto mnoho-súborovým korpusom by boli dôveryhodné.

1.1 Vymedzenie cieľov

Cieľom tejto práce je navrhnúť štruktúru mnoho-súborového korpusu, ktorého zdrojové súbory nebudú prístupné užívateľom, aby nedochádzalo k špecifickému vývoji kompresných algoritmov s cieľom dosiahnuť uspokojivé výsledky v meraných parametroch. Ďalším bodom požiadaviek je vytvorenie testovacej aplikácie, ktorá by bola schopná otestovať jednotlivé programy (algoritmy) nad vytvoreným mnoho-súborovým korpusom, a zároveň by mala byť schopná uchovávať výsledky testov pre komplexné štatistické vyhodnotenie kvality otestovaných programov a ich následné sprístupnenie registrovaným užívateľom.

Výsledná práca by mala obsahovať nasledujúce požiadavky:

- *definíciu hierarchickej štruktúry efektívneho mnoho-súborového korpusu, hierarchická štruktúra korpusu by mala byť rozčlenená do kategórií a ich subkategórií, ktoré budú obsahovať veľké množstvo dát, rádovo tisícok až desaťtisícok súborov pre každú kategóriu*

- *vytvorenie webevoj aplikácie na testovanie kompresných programov, ktorú budú môcť používať iba registrovaný užívatelia, aplikácia bude testovať kompresné programy autorov nad vytvoreným korpusom, aplikácia musí byť schopná implementácie nezávislej od operačného systému. Testovacia webová aplikácia musí testovať tri základné parametre - kompresný pomer - CR, rýchlosť kompresie – TC, rýchlosť dekompresie - TD, a zisťovanie korektností kompresie po následnej dekompresii dát. Užívatelia (autori kompresných programov) uploadujú svoje programy do webovej aplikácie (Corpus systému), zadávajú parametre pre tieto programy, testujú ich a majú možnosť prezerania výsledkov testov. Admini používajú Corpus systém na správu aktívneho Corpusu, správu užívateľských účtov a prezeranie výsledkov jednotlivých testov. Corpus nesmie byť dostupný verejne.*

- *štatistické vyhodnotenie výsledkov testov kompresných programov pomocou štatistických hypotéz a vytvorenie reportov zo štatistických dát do rôznych formátov (napr. TEX, R-Graph)*

1.2 Štruktúra práce

Pre splnenie prvého z cieľov je potrebný komplexný rozbor navrhovanej hierarchickej štruktúry corpusu, najmä jeho potreba rozdelenia do jazykových kategórií. Správna štruktúra navrhnutého corpusu je veľmi dôležitá pre odstránenie negatívnych vlastností Cantebury corpusu a Calgary corpusu, a naopak využitie čo najviac pozitívnych vlastností týchto dvoch korpusov. Pre splnenie druhého cieľa je potrebný správny výber aplikačného servera, databáze a programovacej technológie, pomocou ktorej bude vývoj Corpus systému (webovej aplikácie) najefektívnejší s cieľom implementácie v najrozšírenejších operačných systémoch. Pre splnenie posledného tretieho cieľa je dôležitý správny výber štatistickej metódy, s tým aby niektoré výsledné hodnoty testov prekračujúce limitné parametre neovplyvnili celkové štatistické porovnanie testov kompresných programov.

Ďalší text tejto práce je štruktúrovaný nasledovne. V druhej kapitole je podrobne popísaný návrh mnoho-súborového corpusu. Kapitola 3 je určená návrhu Corpus Systému, ktorý zahrňuje webovú aplikáciu pre užívateľa a admina, databázu pre ukladanie kompresných programov, výsledných dát ich testov ďalších dát potrebných pre správne fungovanie Corpus Systému. Kapitola 4 popisuje výber zvolenej štatistickej metódy pre vyhodnocovanie kompresných programov, jej klady a zápory. Piata kapitola popisuje možnosti rozšírenia práce. Záverečné zhodnotenie splnenia cieľov je popisané v šiestej kapitole.

Súčasťou tejto diplomovej práce je aj inštalačné CD pre Corpus Systém. Ďalej práca obsahuje podrobnú inštalačnú príručku a manuály pre užívateľa a admina Corpus Systému.

Kapitola 2

Návrh štruktúry korpusu

Predmetom tejto kapitoly je rozbor návrhu hierarchickej štruktúry mnoho-súborového korpusu a následné popísanie dôvodov voľby štruktúry navrhovaného korpusu, jeho porovnanie s Calgary corpusom a Cantebury corpusom. Výsledkom návrhu má byť hierarchicky štruktúrovaný korpus, rozčlenený podľa typu dát do kategórii a subkategórií. Štruktúra musí byť schopná obsiahnuť rádovo tisíce až desaťtisíce súborov v každej kategórii, čím sa odstránia náhodne výsledky testov na efektívnosť jednotlivých kompresných programov. Pre lepšie zorientovanie v návrhu štruktúry sú v textu zobrazené doplňujúce diagramy jednotlivých kategórii štruktúry korpusu.

2.1 Štruktúra korpusu

Základná kostra štruktúry korpusu vychádza z požiadaviek na vylepšenie existujúcich korpusov (Cantebury corpus a Calgary corpus). Pri existujúcich korpusoch je problém s náhodnými výsledkami testov efektívnosti kompresných algoritmov spôsobený malým počtom dát obsiahnutých v korpusoch.

Prioritou 1 pre novú štruktúru je najrealistickejšia reprezentácia typov súborov, na ktoré budú v budúcnosti jednotlivé kompresné algoritmy použité. Z tohto dôvodu, korpus musí obsiahnuť dáta rádovo desaťtisíc súborov všetkých najpoužívanejších typov a formátov.

2. prioritnou vlastnosťou korpusu je jeho veľkosť, respektíve čas potrebný na dostačujúce otestovanie kompresného programu. Riešením je stromová hierarchická štruktúra korpusu, rozčlenená do kategórií a subkategórií typov súborov, z čoho vyplýva možnosť testovať algoritmus iba nad určitou časťou korpusu, čiže dopracovanie sa výsledkov v reálnom čase.

3. prioritnou vlastnosťou novej štruktúry je efektívna a zrozumiteľná správa dát v korpuse adminom (správcom systému, korpus). Vyhovujúcim riešením je rozdelenie veľkého množstva dát do hierarchickej štruktúry kategórií a subkategórié podľa rôznych parametrov dát, typov súborov a ich formátov.

Hierarchická štruktúra kategórií a subkategórií rieši aj ďalšiu požiadavku na korpus – priorita 4 – vyhodnocovanie výsledkov testov kompresných algoritmov vzhľadom na jeden určitý typ dát a následné vytvorenie poradia pre najefektívnejšie programy komprimujúce texty napísané napríklad anglickou abecedou.

5. prioritnou vlastnosťou korpusu je jeho implementácia v aplikácii Corpus systému. Pre programovacie postupy je najvýhodnejšia forma prenosu a uloženia podobnej štruktúry akou je korpus, použitie štandardizovaného formátu, ktorým je XML (Extensible Markup Language). XML jazyk umožňuje vytváranie hierarchických štruktúr rozčlenených na kategórie a subkategórie, tým pádom návrh štruktúry korpusu a jeho implementácia nemusia podliehať žiadnej transformácii do dvoch rozdielnych formátov.

2.1.1 Prvá úroveň štruktúry korpus

Hierarchia korpusu je založená na štruktúre vetvenia stromu, kde uzly predstavujú jednotlivé kategórie a subkategórie, listy stromu reprezentujú vlastné dáta (súbory) korpusu. Uzly stromu slúžia na orientáciu v korpuse, pre vyhľadávanie dát podľa konkrétneho požadovaného typu. Stromová štruktúra postupuje zvrchu (koreň stromu) na dol (listy), pričom sa postupne rozvetvuje v uzloch stromu. Tak isto význam kategórií klesá od vrchných úrovní po listy. Najvyššie úrovne predstavujú najvšeobecnejšie rozdelenia typov dát, najnižšie úrovne spolu s celou cestou kategórií v strome presne definujú konkrétny typ dát v liste a veľkosť súboru mapovaného týmto listom. Stromová štruktúra rodičov a potomkov je veľmi výhodná pre neskoršie modifikácie subkategórií na ktorejkoľvek hladine stromu.

Najvyššia úroveň korpus štruktúry obsahuje kategórie rozdelené podľa príslušnosti súborov k aktivitám, ktoré sú používané užívateľmi alebo užívateľskými aplikáciami.

Kategórie:

- *documents* – obsahuje subkategórie formátov najpoužívanejších dokumentov, ktoré sa používajú pri publikovaní rôznych druhov kníh, článkov, textov, manuálov...

- *web* – ďalšia kategória obsahujúca texty, články a sokumenty zobrazené na webových stránkach štandardným jazykom HTML (Hypertext Markup Language)

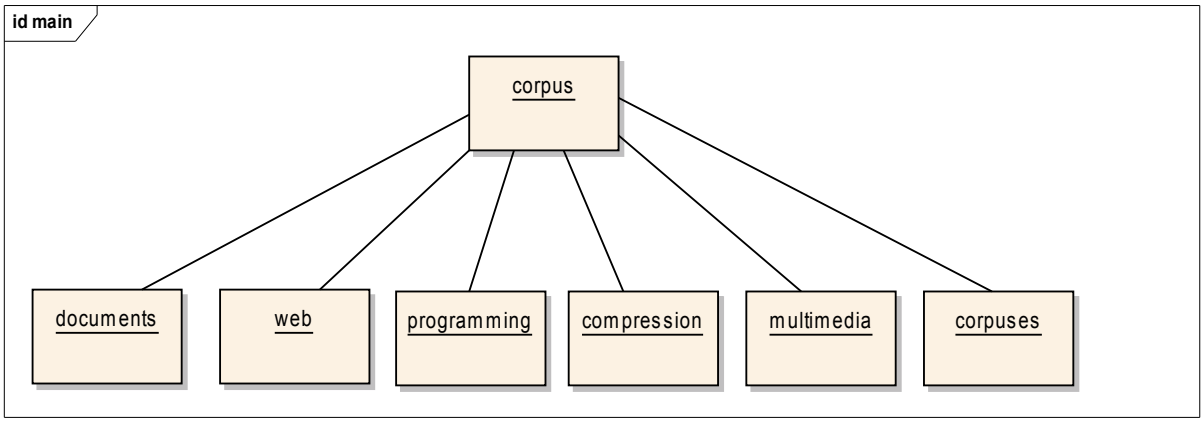
- *programing* – kategória ktorá zahŕňa súbory zdrojových kódov a kompilovaných kódov najpoužívanejších vývojárskych/programátorských a skriptovacích jazykov

- *compression* – obsahuje súbory, ktoré sú výsledkom najpoužívanejších kompresných programov

- *multimedia* – kategória obsahujúca multimediálne súbory najpoužívanejších audio a video kompresných formátov

- *corpuses* – obsahuje vo svojich subkategóriách dáta ďalších známych používaných korpusov

Diagram 1 – 1. úroveň korpusu

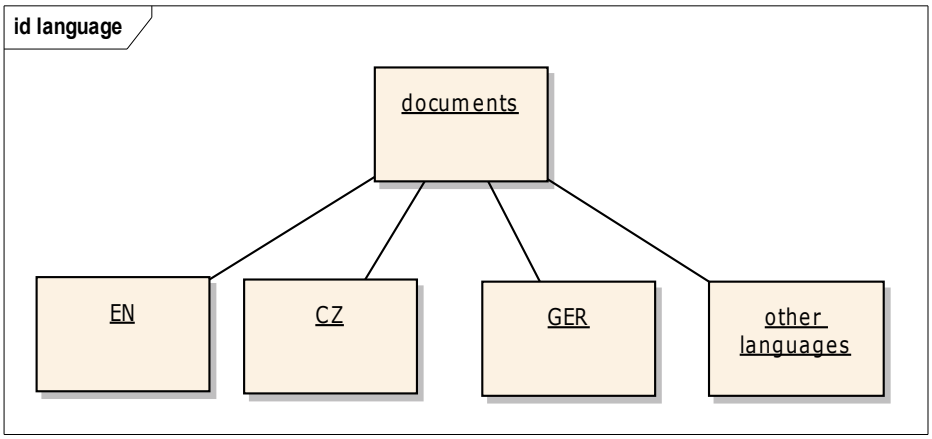


2.1.2 Kategória documents

Kategória *documents* je najrozsiahlejšia a najvýznamnejšia z kategórií. obsahuje subkategórie formátov najpoužívanejších dokumentov, ktoré sa používajú pri publikovaní rôznych druhov kníh, článkov, textov, manuálov. Skladá sa z troch hlavných podúrovní – úroveň *language*, úroveň typov dokumentov podľa obsahu a úroveň typov formátov súborov dokumentov.

Úroveň *language* vetví korpus na subkategórie podľa národného jazyka, v ktorom bol dokument napísaný. Jednotlivé národne jazyky používajú abecedu v určitej znakovej sade. Kompresné programy pri kompresii dvoch súborov obsahujúcich texty vytvorené dvoma rozličnými abecedami znakových sád, nevrátia pre každý prípad rovnaké výsledky troch sledovaných parametrov kompresie (*CR*, *TC, TD*). Preto je význam rozdelenia dokumentov do subkategórií jazykov dôležitý.

Diagram 2 – úroveň language



Úroveň *document content* vetví korpus na subkategórie podľa typu obsahu textu v dokumente. Jednotlivé subkategórie tejto vetve korpusu rozdeľujú dokumenty podľa veľkosti napr. *book*, *publication*, *dictionary*. Ďalšou úrovňou je vymedzenie množiny slov jazyka najčastejšie použitých pre danú kategóriu podľa odboru použitia dokumentu napr. *free time*, *manual*, technical. Táto úroveň je rozšíriteľná do konkrétnejších užšie špecifikovaných oblastí použitia významu textu v dokumente podľa okruhu záujmu napr. *poetry*, *prose*, *math*, *sport*.

Úroveň *file type* vetví korpus na subkategórie podľa typu súboru, ktorý reprezentuje daný dokument. Obsahuje 3 hlavné subkategórie *Office dokumenty*, zvyšné špeciálne formáty dokumentov napr. *pdf*, *tex*, a poslednou kategóriou v úrovni *file type* sú súbory plain textov, ktoré sú rozšírené ešte o úroveň znakových sád. Každý typ súboru používa špecifickú abecedu slov alebo znakov ktorá je rozdielna od zvyšných typov súborov. Táto špecifická abeceda slov a znakov definuje vlastný typ súboru (najčastejšie v hlavičke súboru). Častejšie používanie konkrétnej množiny znakov ovplyvňuje výsledky sledovaných parametrov kompresných programov (*CR, TC, TD*).

Poslednou úrovňou ktorá je zhodná pre všetky kategórie korpusu je úroveň rozdelenia dát v listoch stromu podľa veľkosti súborov. Veľkosti súborov sú vždy rozdelené do troch kategórií :

- *malé súbory* – kategória obsahuje súbory veľkosti KB
- *stredne veľké súbory* – kategória vymedzená pre súbory veľkostí MB
- *veľké súbory* – obsahom kategórie sú súbory veľkostí GB

Diagram 3 - úroveň document content

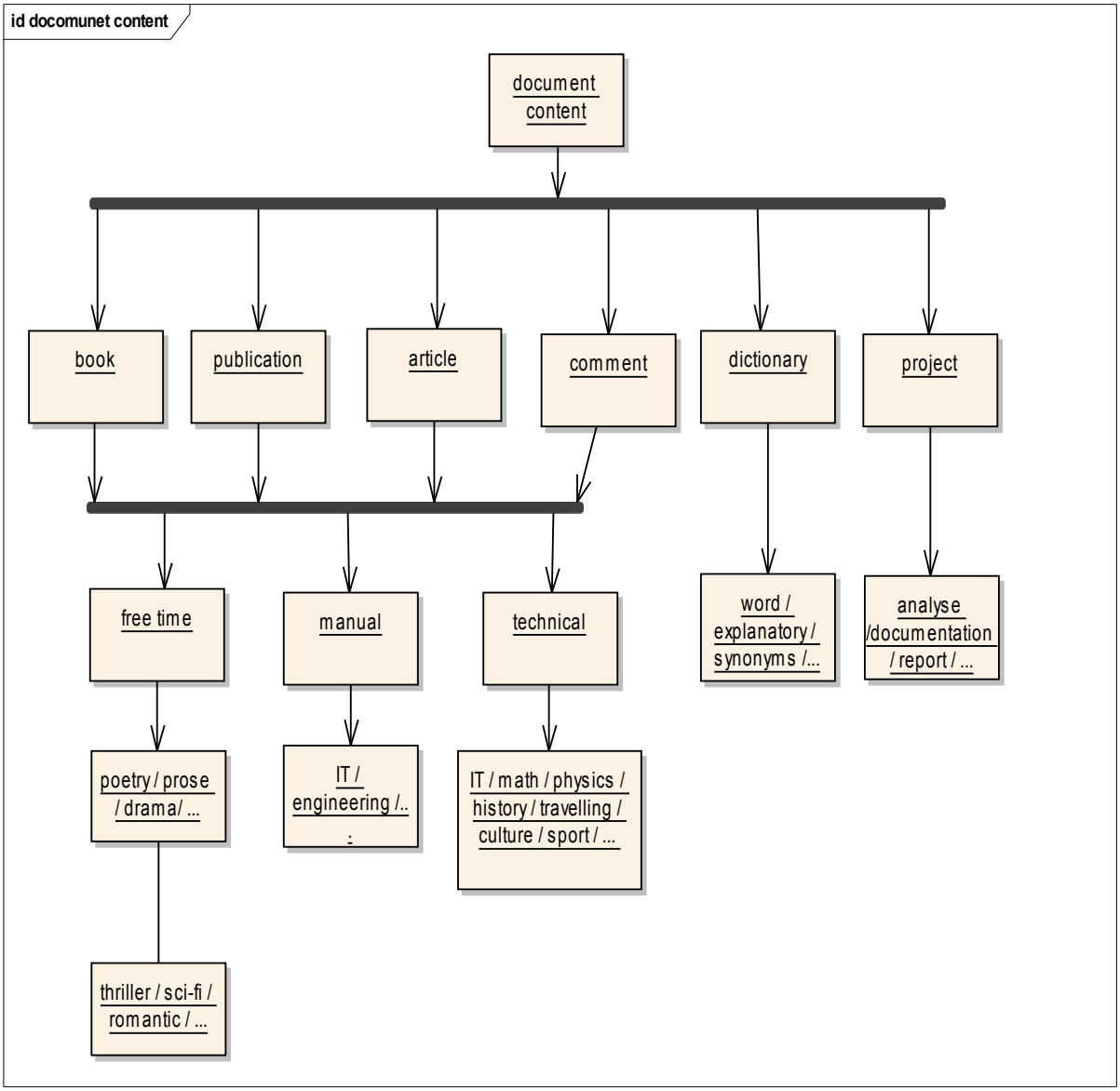
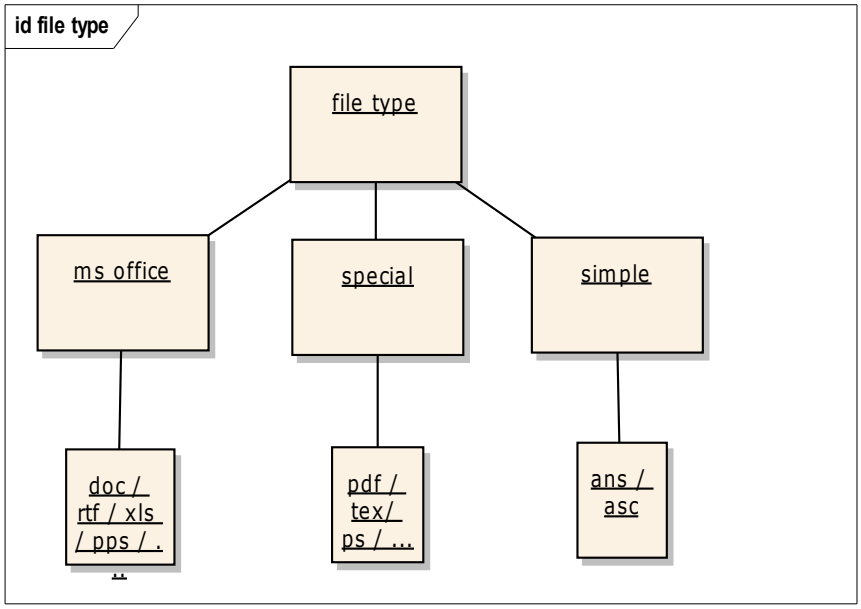


Diagram 4 - úroveň file type

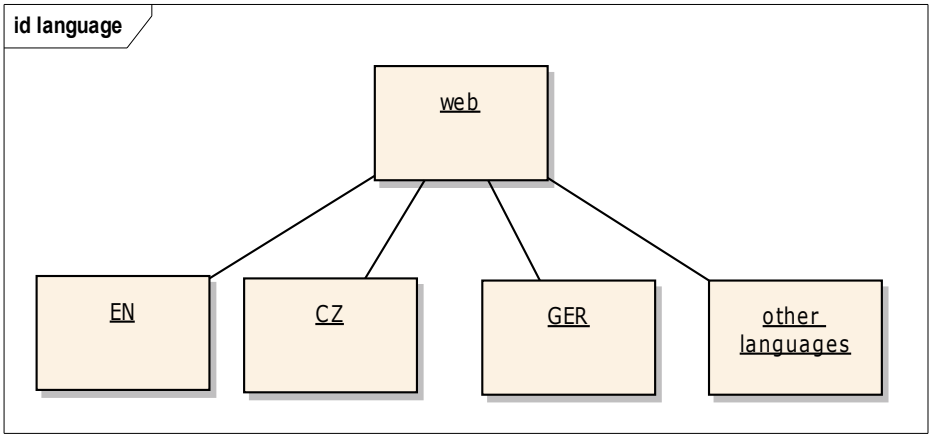


2.1.3 Kategória web

Kategória *web* je navrhnutá pre texty, články a dokumenty zobrazené na webových stránkach štandardným jazykom HTML (Hypertext Markup Language) a XHTML (Extensible Hypertext Markup Language). Kategória je podobná svojou štruktúrou kategórii *documents*, pretože zoskupuje podľa delenia obsahu druhy textov zrovnateľných v *documents* kategórii. Hlavná hierarchia tejto kategórie sa skladá z 3 základných úrovní - úroveň *language*, úroveň typov stránok podľa obsahu a úroveň typov technológií použitých na vytvorenie výslednej stránky.

Úroveň *language* vetví korpus kategóriu web na subkategórie podľa národného jazyka, v ktorom bol dokument napísaný. Jednotlivé národne jazyky používajú abecedu v určitej znakovej sade. Kompresné programy pri kompresii dvoch súborov obsahujúcich texty vytvorené dvoma rozličnými abecedami znakových sád, nevrátia pre každý prípad rovnaké výsledky troch sledovaných parametrov kompresie (kompresný pomer, rýchlosť kompresie, rýchlosť dekompresie). Preto je dôležitý význam rozdelenia dokumentov do subkategórií jazykov.

Diagram 5 - úroveň language



Úroveň *web content* vetví korpus na podskupiny podľa typu obsahu textu na stránke. Jednotlivé subkategórie tejto vetve korpusu rozdeľujú stránky podľa veľkosti napr. *forum*, *publication*, *dictionary*. Ďalšou úrovňou je vymedzenie množiny slov jazyka najčastejšie použitých pre danú kategóriu podľa odboru pre ktorý je stránka vytvorená napr. *free time*, *manual*, *technical*. Táto úroveň je rozšíriteľná do konkrétnejších užšie špecifikovaných oblastí použitia významu textu v dokumente podľa okruhu záujmu napr. *poetry*, *prose*, *math*, *sport*.

Úroveň page type vetví korpus na subkategórie podľa typu stránky, ktorým programovacím jazykom bola stránka vytvorená. Obsahuje hladinu subkategórií najpoužívanejších jazykov slúžiacich na vytváranie webových stránok napr. *html*, *xhtml*, *jsp*. Každý typ jazyka používa rozdielnu syntax abecedy slov a tým pádom častejšie používanie špecifickej množiny slov na danej stránke. Častejšie používanie jednej množiny slov ovplyvňuje výsledky sledovaných parametrov kompresných programov (*CR, TC, TD*).

Poslednou úrovňou ktorá je zhodná pre všetky kategórie korpusu je úroveň rozdelenia dát v listoch stromu podľa veľkosti súborov. Veľkosti súborov sú vždy rozdelené do troch kategórií :

- *malé súbory* – kategória obsahuje súbory veľkosti KB
- *stredne veľké súbory* – kategória vymedzená pre súbory veľkostí MB
- *veľké súbory* – obsahom kategórie sú súbory veľkostí GB

Diagram 6 - úroveň web content

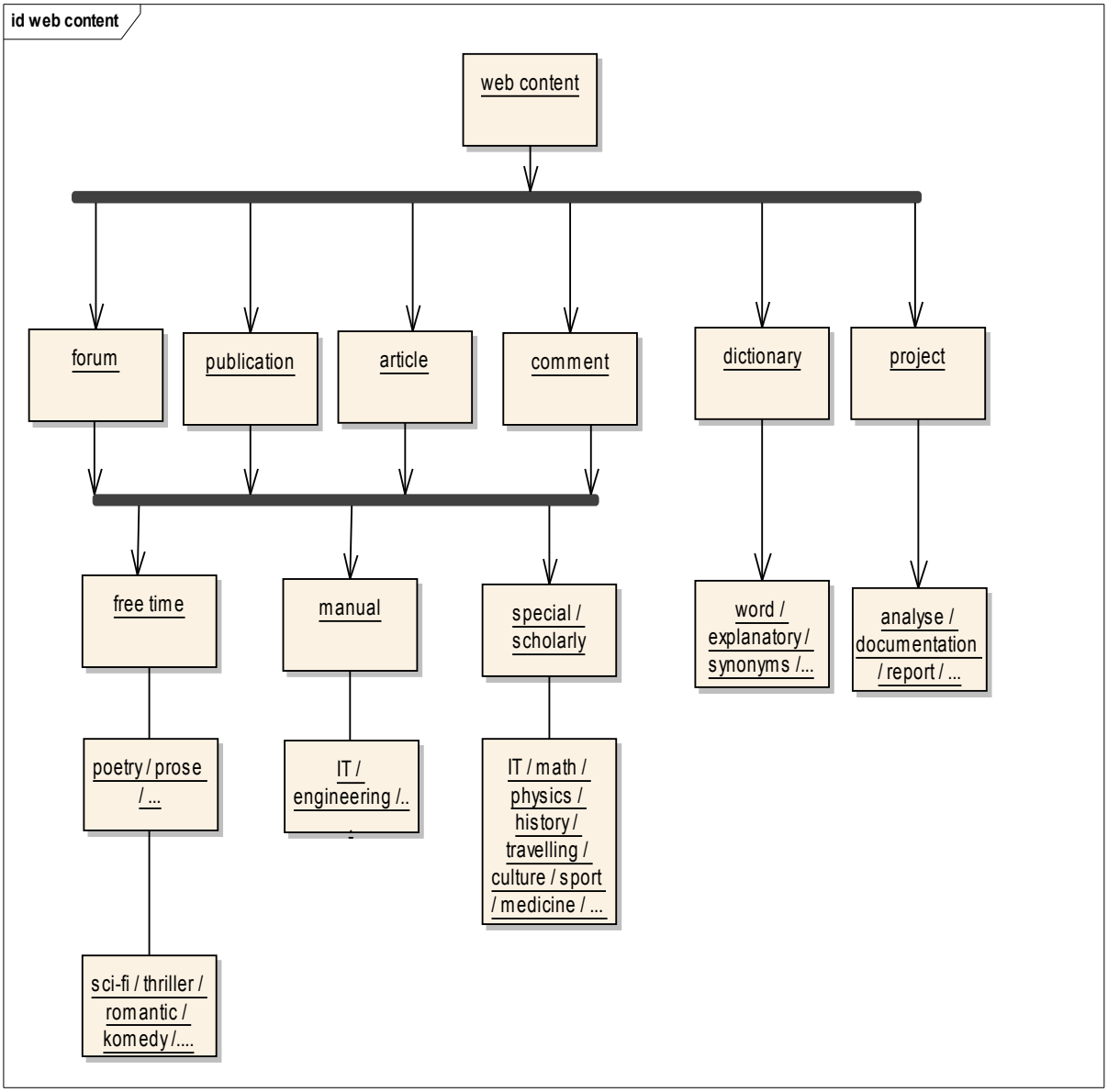
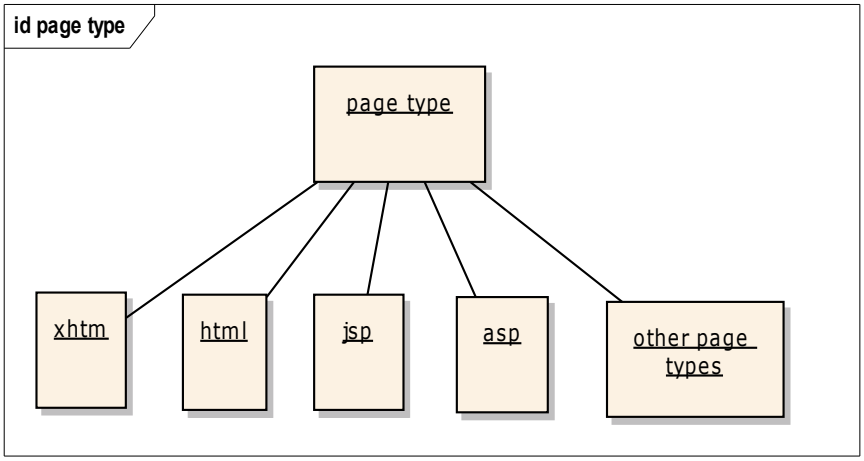


Diagram 7 - úroveň page type



2.1.4 Kategória programing

Programing kategória ktorá zahŕňa súbory zdrojových kódov a kompilovaných kódov najpoužívanejších vývojárskych/programátorských a skriptovacích jazykov. Najvyššia úroveň tejto kategórie vetví súbory do štyroch subkategórií rozdelených podľa funkcie súboru pre ďalšie spracovanie na skriptovacie jazyky, zdrojové kódy programovacích jazykov, kompilované výsledky zo zdrojových kódov a na podporné súbory, ktoré sa používajú pri vývoji systémov a aplikácií. Štruktúra takto definovaných subkategórií rieši sémantickú a syntaktickú rozdielnosť dát, ktorá ovplyvňuje výsledky sledovaných parametre kompresií s týmito vstupnými dátami (*CR*, *TC*, *TD*).

Subkategória *script languages* definuje skupinu najrozšírenejších skriptovacích jazykov používaných pre rozširovanie funkcií aplikácií a programov. Do tejto množiny patria skripty napr. *java script, perl, psql*.

Najvýznamnejšia subkategória z pohľadu testov kompresných programov je *source code* subkategória, ktorá zoskupuje súbory so zdrojovými kódmi najrozšírenejších programovacích a vývojárskych jazykov súčasnosti. Patria sem napr. *java, c++, c, visual basic, cobol.* Špecifické programovacie jazyky využívajú okrem hlavného zdrojového súboru, pomocné zdrojové súbory. Z tohto dôvodu je kategória pre daný programovací jazyk rozšírená o úroveň rozdeľujúcu súbory podľa typu využitia v danom jazyku, ak pre daný jazyk takéto pomocné súbory existujú, napr. pre jazyk *C* sú týmto rozšírením header súbory *h.*

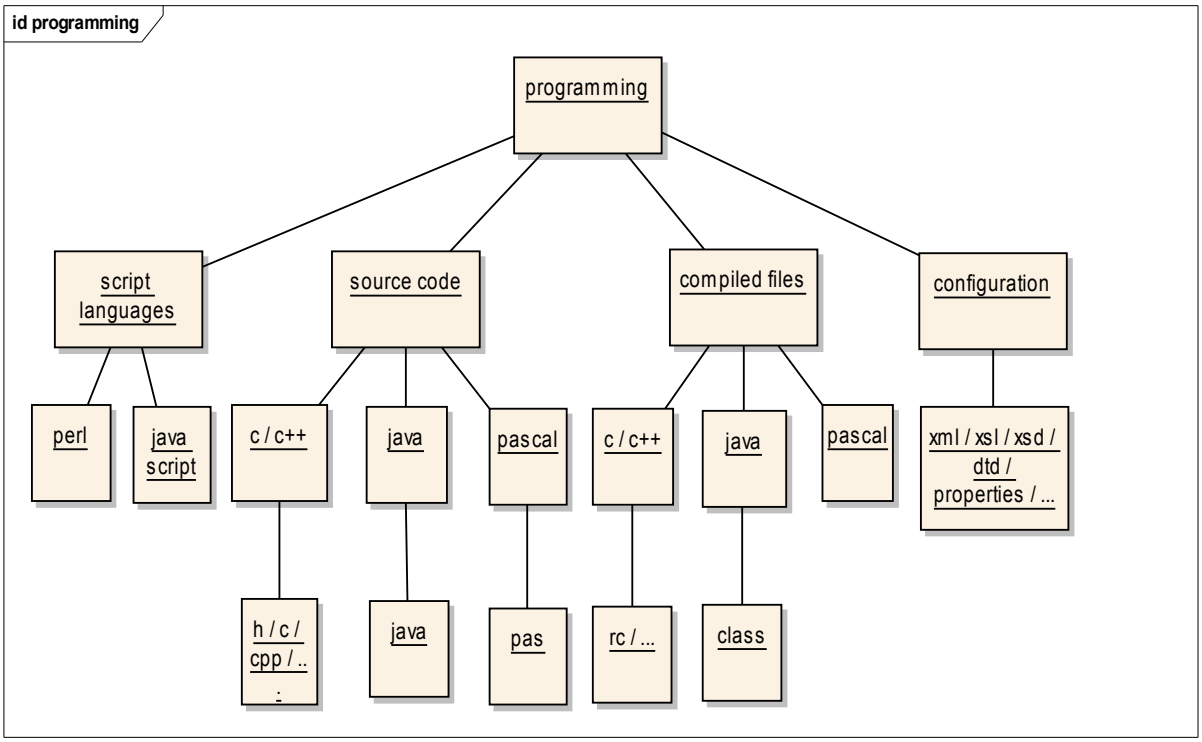
Keďže jednotlivé zdrojové kódy sú kompilované prekladačom do výstupných súborov, ktorých obsah má špecifickú štruktúru pre každý programovací jazyk, výstupné kompilované súbory tvoria vlastnú subkategóriu *compiled* files kategórie *programing*, napr. *class, rc.*

Subkategória *configuration* definuje skupinu pomocných súborov nutných pre správny chod aplikácií a programov. Najpoužívanejšími formátmi konfiguračných súborov sú formáty *xml* a formát *properties* súborov. Ďalej do tejto subkategórie patria súbory pre prevod dát zo zdrojových formátov do výstupných zformátovaných sád - *XSL* (eXtensible Stylesheet Language Transformations) a súbory popisujúce štruktúru xml dokumentov – *XSD* (XML Schema) a *DTD* (Document Type Definition). .

Posledná úroveň ktorá je zhodná pre všetky kategórie korpusu, nie je tak významná pre kategóriu *programing*, pretože väčšina súborov je obsiahnutá v subkategórii *small size.* Subkategórie veľkostí súborov :

- *malé súbory* – kategória obsahuje súbory veľkosti KB
- *stredne veľké súbory* – kategória vymedzená pre súbory veľkostí MB
- *veľké súbory* – obsahom kategórie sú súbory veľkostí GB

Diagram 8 - úroveň programing

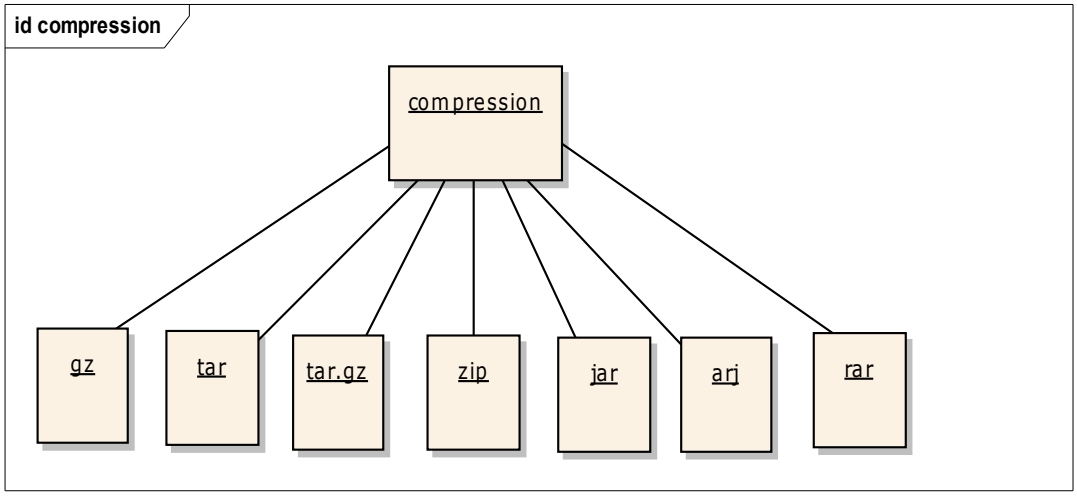


2.1.5 Kategória compression

Compression kategória zoskupuje súbory, ktoré sú výsledkom najpoužívanejších kompresných programov. Štruktúra kategórie je veľmi jednoduchá, obsahuje dve úrovne subkategórií. Prvú úroveň tvoria subkategórie typov kompresných programov napr. *gzip, zip, rar, tar*. Druhou a najnižšou úrovňou je úroveň veľkostí súborov:

- *malé súbory* – kategória obsahuje súbory veľkosti KB
- *stredne veľké súbory* – kategória vymedzená pre súbory veľkostí MB
- *veľké súbory* – obsahom kategórie sú súbory veľkostí GB

Diagram 9- kategória compression

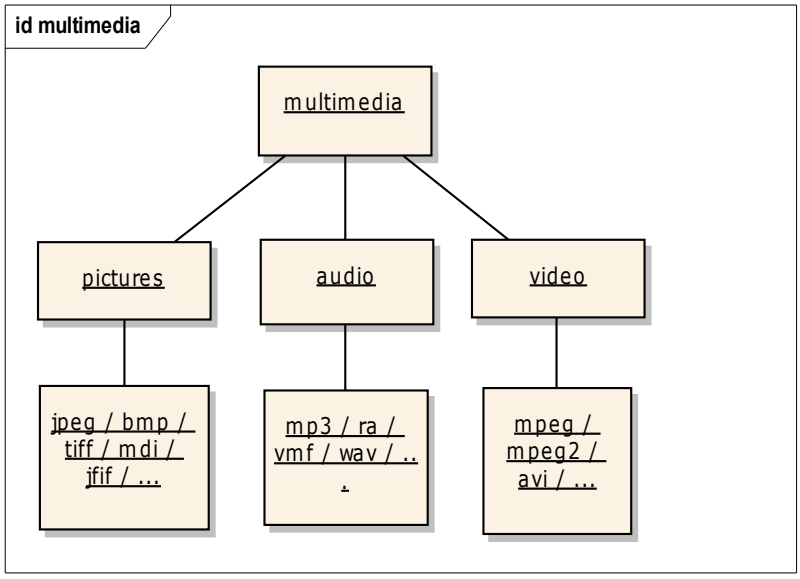


2.1.6 Kategória multimedia

Multimedia kategória obsahuje skupiny multimediálnych komprimovaných súborov. Štruktúra kategórie obsahuje tri úrovne subkategórií. Prvá úroveň vetví skupiny súborov na *pictures*, *audio* a *video* formáty. Druhá úroveň obsahuje subkategórie najpoužívanejších obrazových a zvukových kompresných formátov napr. *jpeg*, *tiff*, *mpeg, mpeg2, avi, mp3, ra.* Treťou a najnižšou úrovňou je úroveň veľkostí súborov:

- *malé súbory* – kategória obsahuje súbory veľkosti KB
- *stredne veľké súbory* – kategória vymedzená pre súbory veľkostí MB
- *veľké súbory* – obsahom kategórie sú súbory veľkostí GB

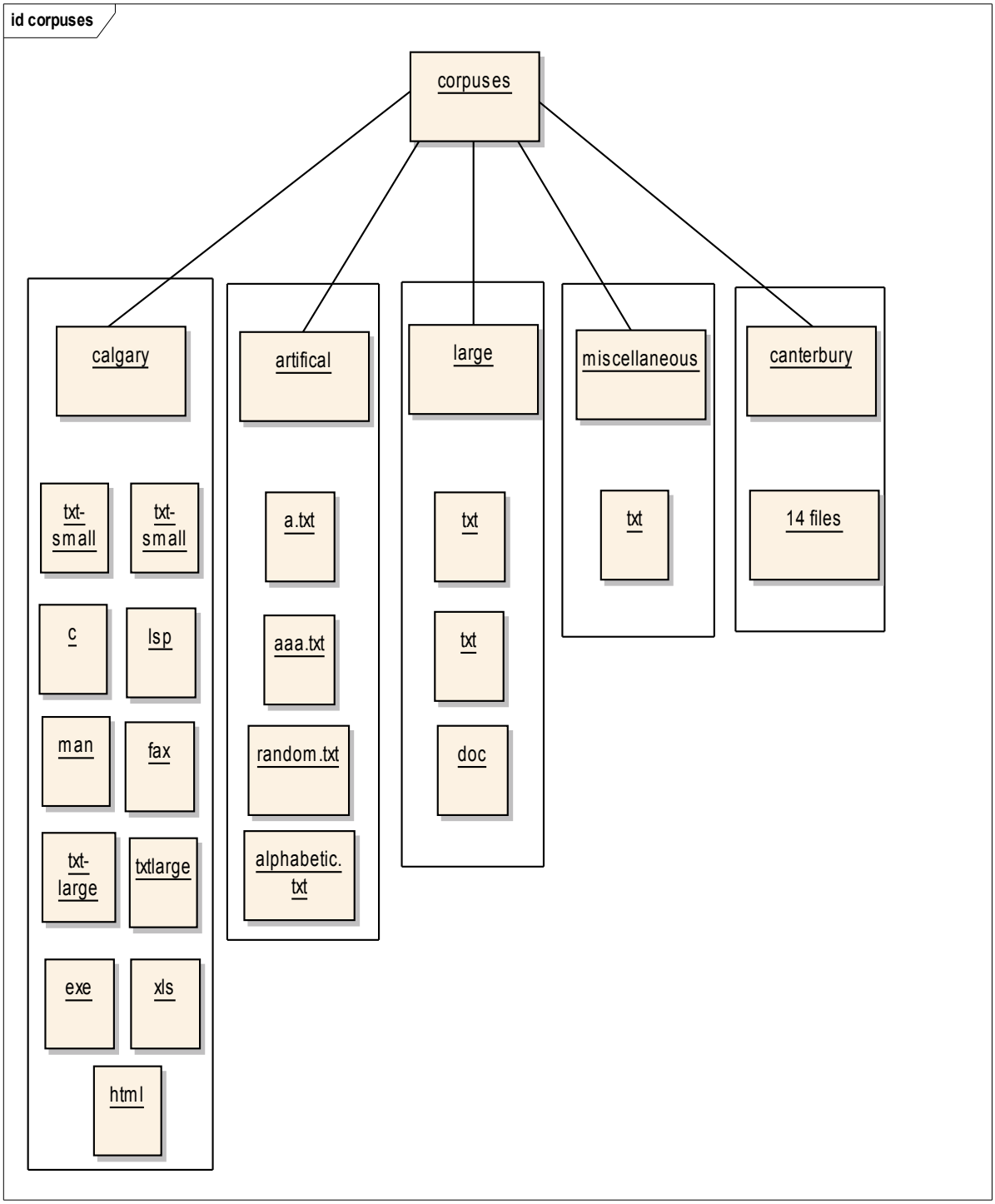
Diagram 10 - kategória multimedia



2.1.7 Kategória corpuses

Posledná kategória umožňuje testovanie kompresných programov za pomoci doposiaľ vyvinutých korpusov. Viz dokumentácia ku korpusom [1] a [2], najvyššia úroveň subkategórií je rozčlenená na *calgary corpus, artifical corpus, large corpus, miscellaneous corpus, cantebury corpus.* Nepriamim dôvodom zahrnutia tejto kategórie do štruktúry korpusu je možnosť porovnania výsledkov testov kompresných algoritmov nad mnoho-súborovým korpusom a doposiaľ vyvinutými korpusami.

Diagram 11 - kategória corpuses



Kapitola 3

Corpus systém

Dôležitou požiadavkou tejto práce je možnosť testovať parametre kompresných programov, a pomocou nich porovnať efektívnosť a účinnosť jednotlivých kompresných algoritmov. K dátam, nad ktorými sú programy testované slúži vytvorený korpus z kapitoly 2. *Corpus systém* musí taktiež obsahovať funkčnosť na zobrazenie výsledkov otestovaných programov a podľa poslednej požiadavky tejto práce, corpus systém musí zobraziť štatistické vyhodnotenie a porovnanie jednotlivých algoritmov.

Keďže kompresné programy testujú vlastní autori týchto programov, *Corpus systém* musí byť dostupný širokému počtu autorov. Z dôvodu verejného prístupu k testovacej funkcii *Corpus systému* a nerozširovania typu dát uložených v korpuse, je najvýhodnejším riešením umožniť prístup užívateľom do systému pomocou webovej aplikácie.

Webová aplikácia spravujúca účty autorov a ich kompresných programov vyžaduje administrátora, ktorý má k dispozícii GUI na správu týchto účtov. Ďalšou úlohou administrátora je správa vlastného korpusu a dát v ňom obsiahnutých. Veľmi účinným riešením je spojiť správu účtov a korpusu adminom do jednej funkčnosti systému, ktorej GUI by bolo taktiež prístupné pomocou webovej aplikácie.

3.1. Požiadavky systému

Základné požiadavky na funkčnosť systému:

- *webová aplikácia s dvoma rozdielnymi rozhraniami – užívateľským a administrátorským*
- *aplikácia implementujúca korpus navrhnutý v kapitole 2*
- *aplikácia spravujúca užívateľské účty autorov kompresných programov, ktorí sú zaregistrovaní do systému*
- *aplikácia testujúca užívateľské kompresné programy a uchovávajúca výsledky týchto testov pre ďalšie spracovanie*
- *aplikácia generujúca štatistické reporty z výsledkov testov užívateľských kompresných programov*
- *implementácia aplikácia nie je závislá od operačného systému*

3.2. Design a implementácia systému

Najvýhodnejším riešením jazyka použitého pre vývoj webovej časti *Corpus systému* je *Java v1.5* [3], ktorá je implementačne nezávislá od operačného systému, čím je splnený posledný bod z požiadaviek na systém. Pre vytvorenie webových stránok – administrátorského a užívateľského GUI systému, ktoré musia zniesť nápor maximálne rádovo desiatok užívateľov v aktuálnom čase, je vhodné použitie *Java* technológie *JSP (Java Server Pages)* [4]. S výberom technológie pre vývoj stránok aplikácie, súvisí výber aplikačného servera na implementáciu webovej časti systému. Opäť aplikačný server musí byť implementačne nezávislý od operačného systému servera, preto vhodnou voľbou je voľne dostupný aplikačný server *Tomcat v5.5* [5].

Ďalšou dôležitou požiadavkou na *Corpus systém* je uchovávanie dát, spojených so správou korpusu, užívateľských účtov a výsledkov testov kompresných programov. Riešením je použitie SQL databáze, ktorá je implementačne nezávislá od operačného systému, voľne dostupná, spoľahlivá, efektívna pri dotazoch nad rádovo tisíckami užívateľských účtov, desaťtisíckami záznamov spojených s korpusom a desaťtisíckami záznamov s dátami o výsledkoch testov kompresných algoritmov. Z dôsledku spomenutých dôvodov je vhodným riešením Postgres SQL databáza.

Druhá podmienka požiadaviek na systém vyžaduje implementáciu korpusu popísaného v kapitole 2. Keďže korpus obsahuje tisíce až desaťtisíce súborov, veľkostí rádovo KB, MB až GB, dáta korpusu musia byť uložené na rovnakom *file systéme* servera, na ktorom je implementovaná webová časť systému, z dôvodu eliminovania časových nárokov na prenos dát medzi serverom na ktorom sú dáta korpusu fyzicky umiestnené a serverom na ktorom bežia testy kompresných algoritmov s využitím súborov korpusu.

Zo spomenutých technologických postupov použitých na základné funkčné komponenty *Cospus systému*, vyplýva štruktúra systému, ktorou je 4 – vrstvová aplikácia:

- *web GUI (administrátorské a užívateľské)*
- *Java main framework (funkčné jadro systému)*
- *databáza*
- *file systém*

Diagram 12 - štruktúra Corpus systému

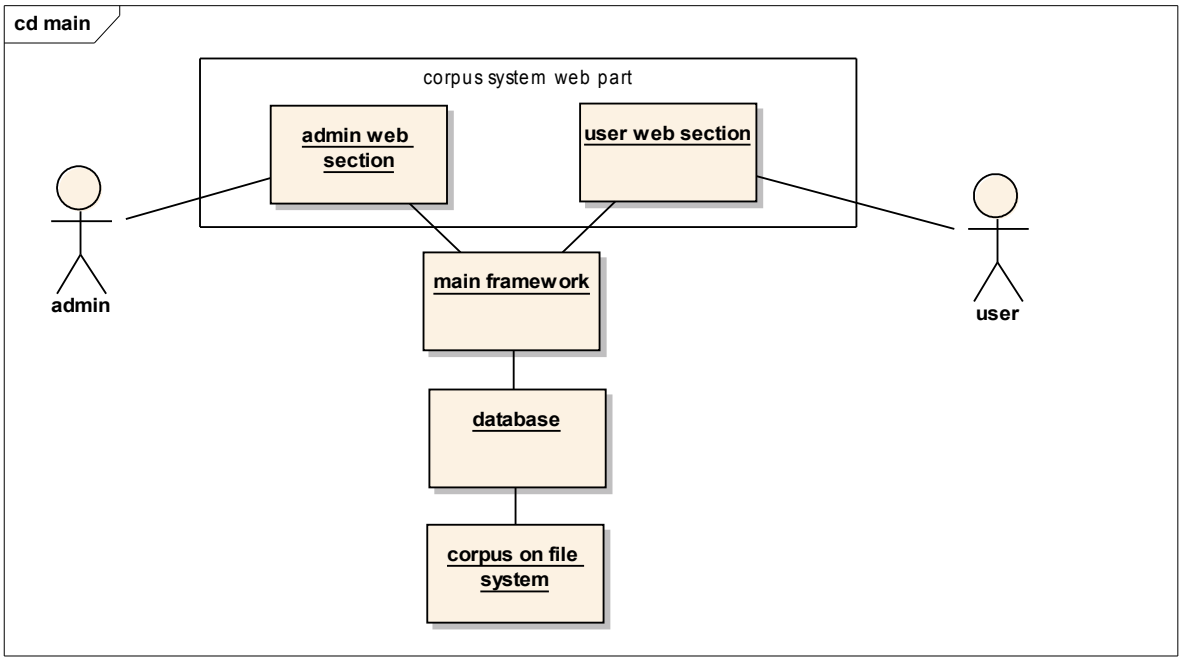
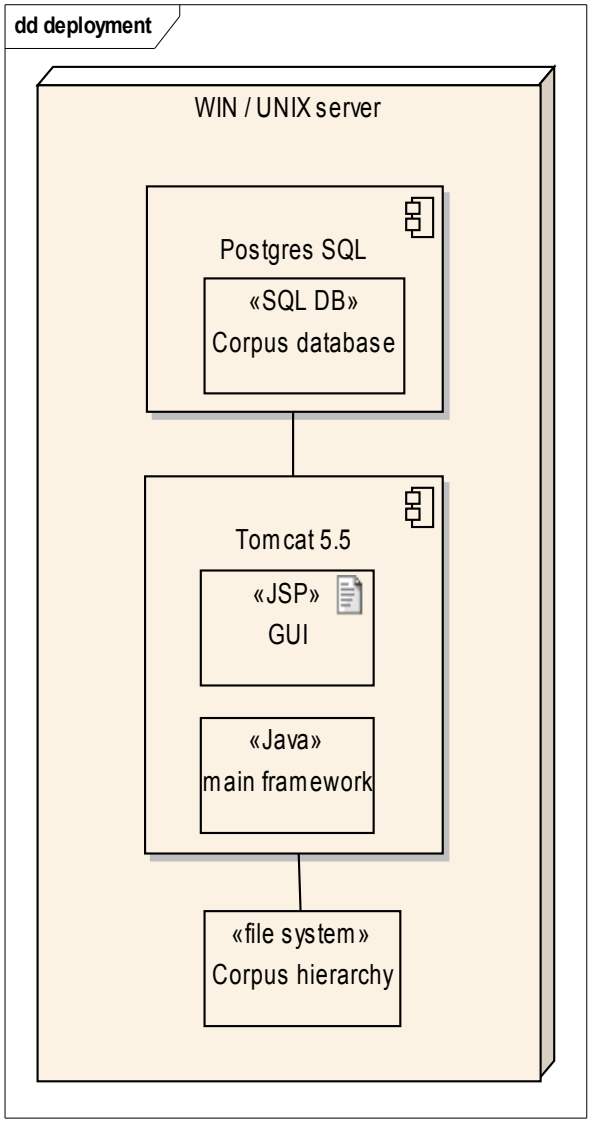


Diagram 13 - Corpus systém deployment



3.2.1 Webová časť Corpus systému

Webová časť systému tvorí najvyššiu vrstvu systému, ktorá je viditeľná pre užívateľov tzv. GUI. GUI systému je rozdelený podľa užívateľských práv umožňujúcich prístup k funkciám na dve základne časti – GUI pre administrátora (admina) a GUI pre autorov kompresných programov (užívateľov). Obidve časti sú navzájom nezávislé. Medzi komponenty ktoré sú zdieľané obidvoma rozhraniami patrí *main framework,* databáza a súbory korpusu uložené na *file systéme* daného servera.

Medzi dôležité interné komponenty oboch webových častí aplikácie patria komponenty:

- *JSP súbory – pomocou ktorých sú generované HTML stránky GUI v závislosti od odoslaného dotazu užívateľa na server*

- *konfiguračné súbory, ktoré sa delia na dva typy - zdieľaná oboma GUI a špecifické pre jednotlivé GUI.*

Zvyšné vrstvy 4 – vrstvovej aplikácie *Corpus systému* patria medzi externé komponenty, ktoré sú nutné ku korektnému spusteniu aplikácie.

Obidve webové časti systému môžu byť prístupné iba s platnými užívateľskými právami. V systéme je implementovaná *Realm* [5] technológia aplikačného servera *Apache Tomcat,* pre povolenie prístupu k jednotlivým zdrojom aplikácie. Technológia *Realm* môže implementovať viacej spôsobov validácie práv užívateľov. V *Corpus systéme* je použitý typ *JDBCRealm*, čiže autentifikácia užívateľov prebieha pomocou informácií o užívateľoch uložených v relačnej databáze, ktorá je navrhnutá pre systém (kapitola 3.2.6) a je prístupná pomocou *JDBC* [6] ovladačov. Autentifikácia *Corpus systém* rozlišuje dva typy prístupových práv (rolí) – *corpus_admin* a *corpus_user*. Užívateľ s prístupovým typom *corpus_admin* má prístup len do administrátorske webovej časti, *corpus_user* užívateľ má prístup len do užívateľskej webovej časti aplikácie.

V obidvoch webových aplikáciách je implementovaná rovnaká štruktúra pre presmerovanie medzi jednotlivými stránkami aplikácie. Väčšinu fyzických stránok reprezentujú príslušná virtuálne akcie, ktoré definujú funkcie prístupné na danej stránke a nasledujúce možnosti presmerovania stránky na ďalšiu stránku po úspešnom alebo neúspešnom vykonaní funkcie. Funkčnosť jednotlivých akcií implementujú *Java* objekty z *main framework* vrstvy systému (viz kapitola 3.2.6). Ak nie je stránka reprezentovaná žiadnou akciou, na tejto stránke nie je možné vykonať žiadnu aktivitu, funkciu. Mapovanie jednotlivých stránok na akcie definuje konfiguračný *XML* dokument *mvc-config.xml,* ktorého štruktúra je popísaná v kapitole 3.2.6 v časti *Action mapping.*

Ďalšou dôležitou funkciou aplikácie z pohľadu admina je možnosť zistiť hlavné dôvody neočakávaných zlyhaní systému v ktoromkoľvek stave procesu alebo v ktorejkoľvek funkčnej jednotke systému. Obidve časti webovej aplikácie používajú totožný systém logovania chýb – apache tomcat technológia *log4j* [7]. Konkrétne je použitý typ logovania *RollingFileAppender*, ktorého vlastnosťou je schopnosť logovať do súboru s definovaným umiestnením na *file systéme*, pričom po dosiahnutí stanovenej veľkosti súboru, je tento súbor indexovaný najnižším číslom a systém pokračuje v logovaní do nového súboru. Konfiguračné parametre pre administrátorskú a užívateľskú aplikáciu sa nachádzajú v súboroch *log4jAdmin.properties, log4jUser.properties a log4jRegistration.properties.*

3.2.2 Administrátorské webová aplikácia

Požiadavky:

- *správa užívateľských účtov a s ňou spojená registrácia nových užívateľov*
- *správa Corpus hierarchie a jej dát na file systéme*
- *správa administrátorskej a užívateľskej webovej aplikácie*

- *zobrazenie výsledkov testov kompresných programov*
- *generovanie štatistických porovnaní testov do rozdielnych formátov reportov*
- *obmedzenie prístupu do aplikácie len pre administrátorské účty*

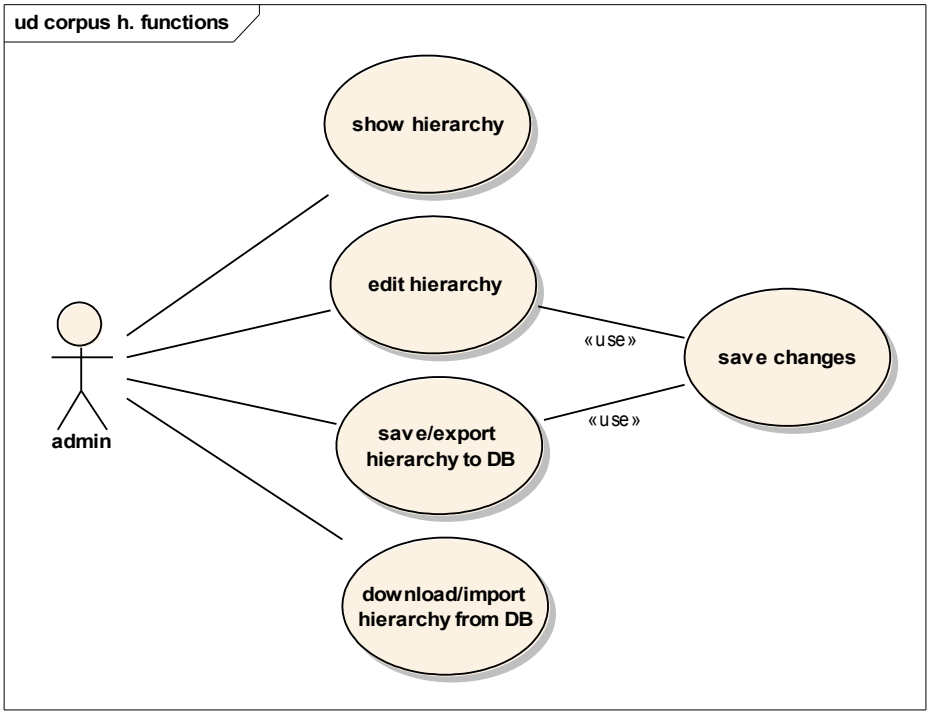
Navrhnutá štruktúra administrátorskej webovej aplikácie a jej GUI pre admina vychádza z požiadaviek funkcií potrebných pre administrátorskú správu kompletného systému. Funkcie admina sú rozdelené do 7 základných skupín:

- *funkcie pre správu virtuálnej štruktúry korpusu (corpus hierarchy)*
- *funkcie pre správu dát korpusu (file / data manager)*
- *funkcie pre správu užívateľských účtov (users)*
- *funkcie pre prezeranie skončených testov kompresných programov (user-test results)*
- *funkcie pre generovanie reportov z výsledkov užívateľských testov, založených na štatistických porovnaniach dát testov (statistic data comparison)*
- *funkcie pre potvrdenie alebo zamietnutie žiadosti o registráciu nových užívateľov (registrations)*

Stránky skupiny *corpus hierarchy* pracujú s hierarchickou štruktúrou korpusu popísanej v kapitole 2 – s tkz. virtuálnou štruktúrou korpusu. Štruktúra je reprezentovaná *XML* dokumentom, ktorý najlepšie vystihuje stromovú štruktúru korpusu v dátovej forme potrebnej pre implementáciu vývojovým jazyka *Java.* Popis štruktúry je v kapitole 3.2.5 časť *Corpus hierarch.*

Virtuálna štruktúra korpusu je uložená v databáze v *XML* formáte v tabuľke *T02_CORPUS_HIERARCHY*. Admin má možnosť modifikovať poslednú aktívnu verziu štruktúry, ktorá je automaticky stiahnutá z databáze a následne zobrazená adminovi. Každá novo vytvorená verzia je uložená do databáze, pričom staršie verzie korpusu taktiež zostávajú uchované v databáze, nie sú prepísané novou verziou. Potvrdením novej verzie, sa táto verzia stáva aktívnou pre celý systém, tým pádom všetky nasledujúce testy testujú kompresné programy podľa tejto aktívnej verzie korpusu. Z dôvodu prehľadnosti štruktúry korpusu, *DATA* elementy neobsahujú kompletné súbory, ale len *ID* referencie, pomocou ktorých sa mapujú konkrétne súbory z fyzického umiestnenia na *file systéme* do korpus štruktúry. Mapovanie ID referencií na konkrétne súbory obsahuje tabuľka *T03_CORPUS_HIERARCHY_SOURCE* v databáze *corpus_server* (viz kapitola 3.2.6).

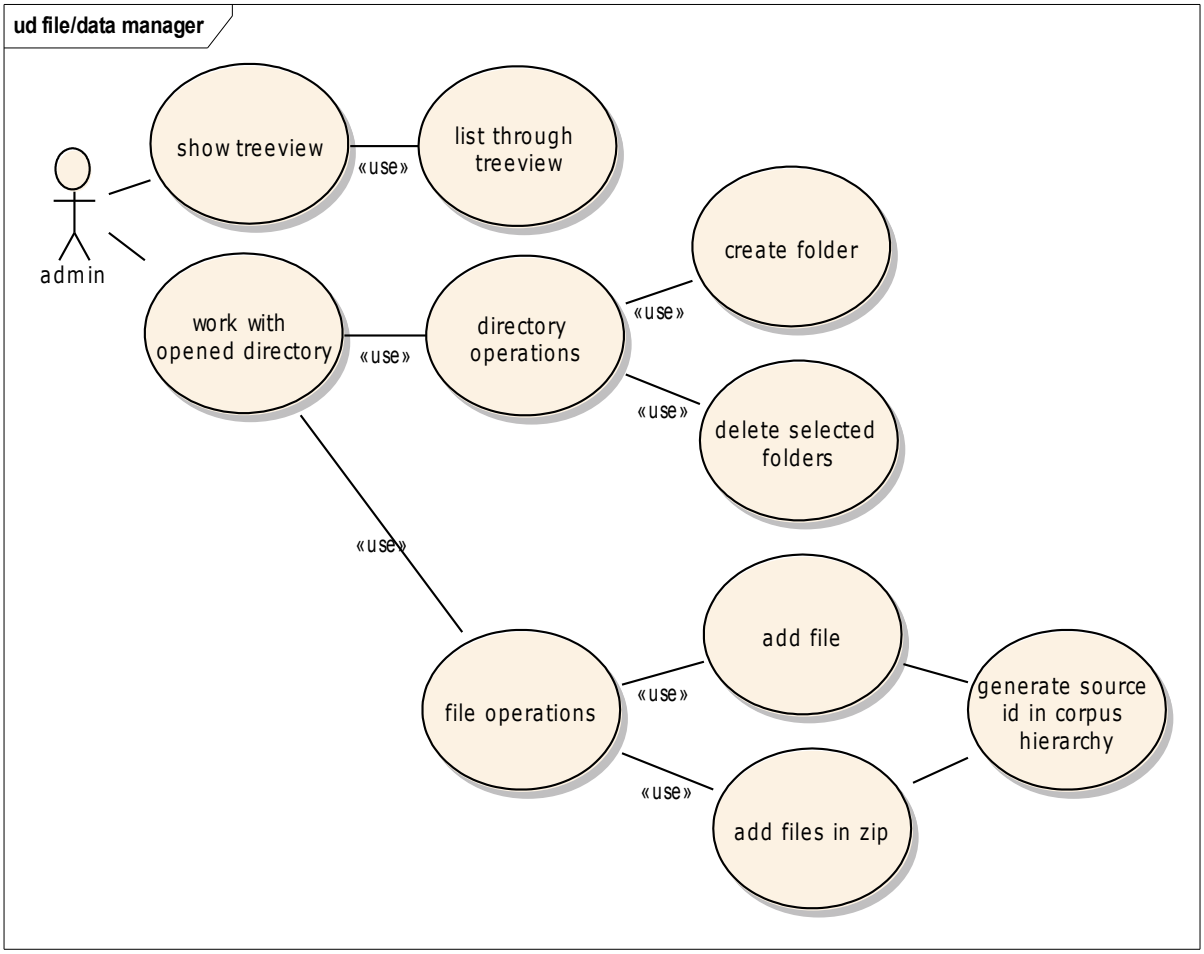
Diagram 14 - funkcie corpus hierarchy



Stránky skupiny *file/data manager* pracujú s fyzickými súbormi uloženými na *file systéme*, tkz. fyzická reprezentácia korpusu. Na týchto súboroch sú testované jednotlivé kompresné programy ich autorov. Admin má k dispozícii základné operácie so súbormi – uploadovanie, mazanie súborov, vytváranie a mazanie adresárov. Aby admin nemusel pri každej zmene virtuálnej štruktúry korpusu premiestňovať, nahrávať alebo mazať súbory fyzickej reprezentácie prepojené s korpusom, štruktúra fyzických súborov na *file systéme* nemusí zachovávať formu štruktúry virtuálneho korpusu. O správne pridelenie súboru do konkrétneho listu korpusu sa stará spomínaná mapovacia tabuľka databáze, ktorá namapuje *ID* referenciu na konkrétny súbor na *file systéme.* Z tohto dôvodu, ak admin odstráni *DATA* element s *name="file_01"* a *source_id="id_01"* z virtuálnej štruktúry, kde umiestnenie *file_01* na *file systéme* je *"/corpus_hierarchy/subdir/file_01"*, nemusí fyzicky odstraňovať súbor, pretože mapovanie na tento súbor už vo virtuálnom korpuse neexistuje. Tým pádom odpadá prípadné opätovné nahrávanie súboru na *file systém*, ak by sa admin rozhodol, tento súbor znova použiť. Pre spomínaný prípad admin jednoducho znova vytvorí element *DATA* vo virtuálnej štruktúre s mapovaním na umiestnenie súboru *file_01.* Ďalšou uľahčujúcou funkciou skupiny *file/data manager* je automatické vytvorenie časti virtuálneho korpusu, pri pridaní väčšieho množstva súborov do fyzickej reprezentácie korpusu.

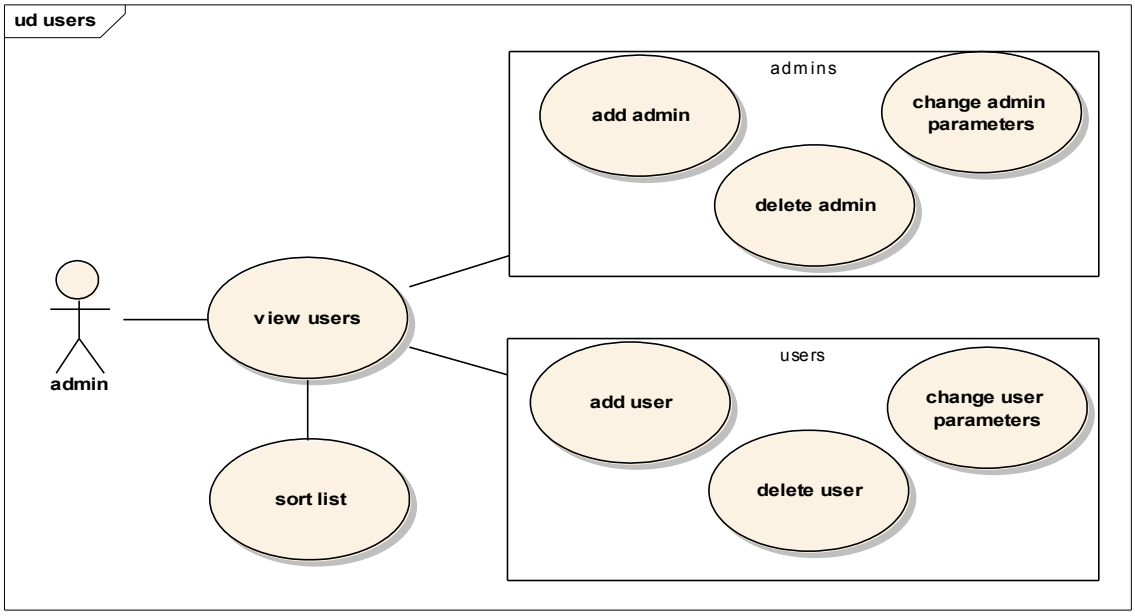
Z popisu skupín funkcií *corpus hierarchy* a *file/data manager* vyplýva, že obe skupiny sú veľmi úzko medzi sebou previazané a definujú kompletnú štruktúru korpusu s dátami, ktorá je potrebná pre testovanie kompresných algoritmov vytvorených užívateľmi.

Diagram 15 - funkcie file/data manager



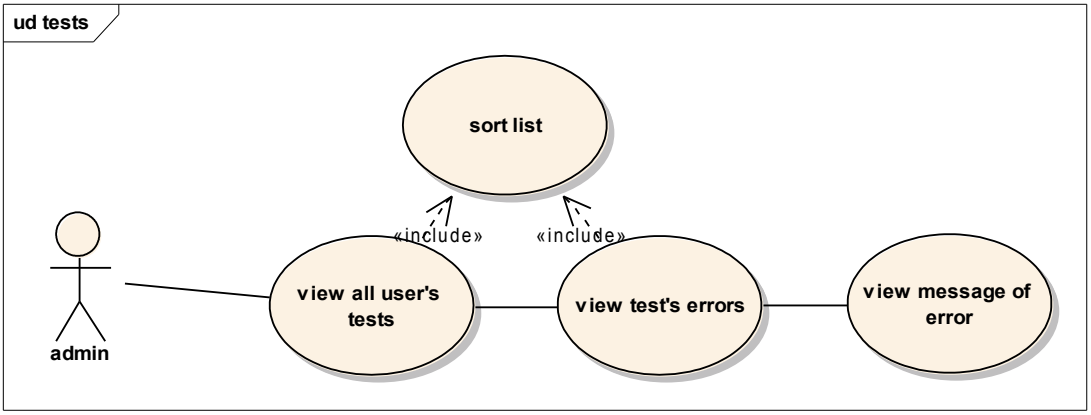
Stránky skupiny *users* implementujú funkcie pre správu užívateľských účtov. Amin má možnosť meniť prihlasovacie parametre účtov *login*, *password* a skupinu prístupových práv, pod ktorou sa bude užívateľ s daným účtom prihlasovať do systému, a tým pádom admin určuje či tento užívateľ je adminom systému alebo bežným autorom kompresných algoritmov. Dáta definujúce jednotlivé účty sa uchovávajú v tabuľkách *T00_USER, T08_ACCESS_ROLE, T12_USER_INFO* databáze navrhnutej pre systém (viz kapitola 3.2.6).

Diagram 16 - funkcie users



Stránky skupiny *user-test results* zobrazujú zoznam všetkých uskutočnených testov kompresných algoritmov, úspešne dokončených aj neúspešných, od všetkých autorov. Všetky potrebné dáta popisujúce testy a ich výsledky sú uložené v databáze (viz kapitola 3.2.6) v tabuľkách *T06_TEST* a *T07_TEST_RESULT*.

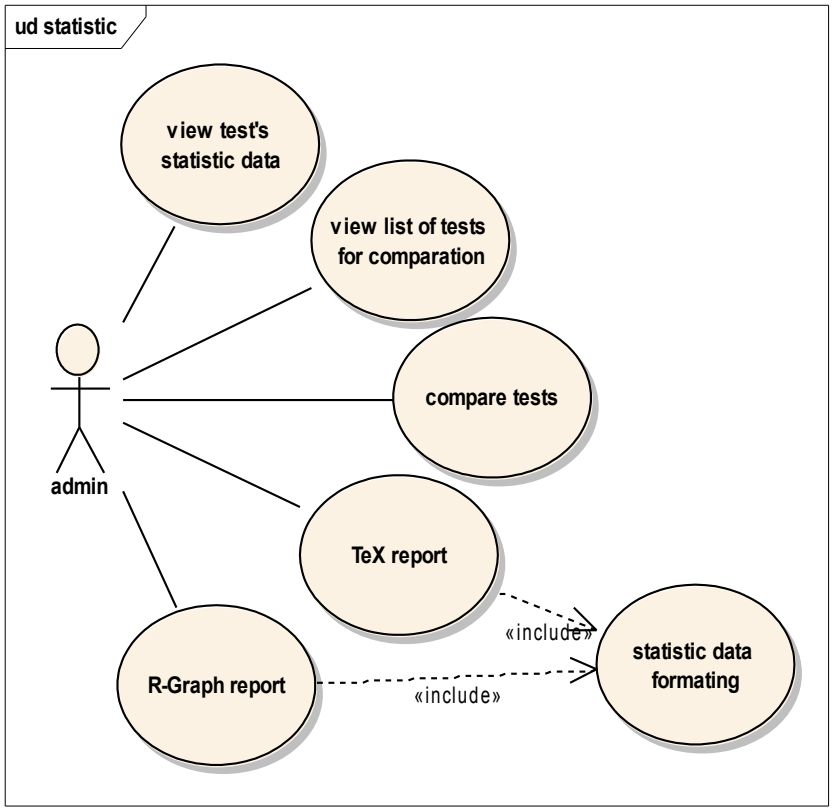
Diagram 17 - funkcie user-test results



Stránky skupiny *statistic data comparison* implementujú funkcie pre štatistické porovnanie výsledkov úspešných testov a funkcie na následné vytvorenie reportov z týchto výsledkov. Štatisticky porovnané môžu byť len testy, ktoré boli uskutočnené nad rovnakou verziou *corpus_hierarchy*. Keďže každá verzia korpusu môže obsahovať rozdielne dáta, porovnávanie výsledkov testov nad rozdielnymi vstupnými dátami je bezúčelné. Systém implementuje Mediánovú štatistickú metódu pre porovnanie testov (dôvody použitia mediánovej metódy sú vysvetlené v kapitole 4).

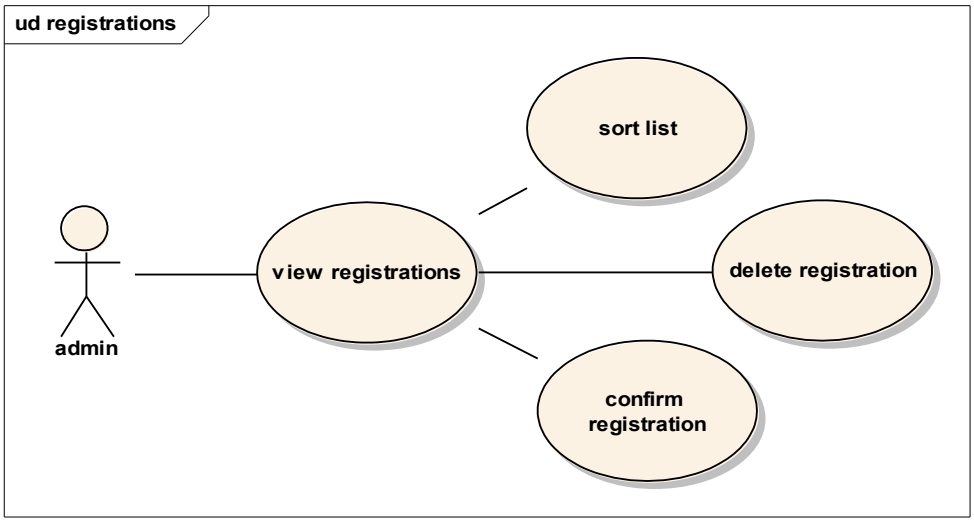
Na navrhovanie formátov reportov z výsledkov testov, slúži špeciálne GUI. Výber testov ktoré majú byť použité do reportu sa realizuje pomocou definovania užívateľských parametrov zo štruktúry *parameter groups* (detailnejší popis štruktúry je v kapitole 3.2.1.2). Hodnoty parametrov z *parameter groups* sa definujú pri inicializácii testu, pred vlastným spustením testu. Tým pádom každý realizovaný test je spojený s určitou definovanou skupinou vlastných hodnôt parametrov. Na vybranie požadovaných dát stačí zadať hodnoty parametrov do filtra reportu. Systém tento filter aplikuje na všetky dostupné testy a zobrazí len tie testy, ktoré spĺňajú podmienky filtra. Pomocou tejto funkcionality je admin schopný vytvoriť si vlastný feedback podľa vlastných požiadaviek. Podporované typy formátov vygenerovaného reportu sú TeX [8] alebo R-Graph [9].

Diagram 18 - funkcie statistic data comparison



Registrations je poslednou skupinou funkcií administrátorskej webovej aplikácie. V nej má admin možnosť zamietnuť alebo schváliť žiadosti užívateľov o prístup do systému. Žiadosti vyplnené rozhraním registrácií (viz kapitola 3.2.1.3) sú uchované v databáze(viz kapitola 3.2.6) v tabuľke *T11_REGISTRATION*. Po potvrdení registrácie je novému autorovi vytvorený užívateľský účet s informáciami, ktoré autor vyplnil v registračnom formulári. Následne je záznam odstránený zo zoznamu žiadostí o registráciu. Ak admin zamietne registráciu, záznam je odstránený zo zoznamu bez ďalších následných aktivít.

Diagram 19 - funkcie registrations



3.2.3 Užívateľská webová aplikácia

Požiadavky:

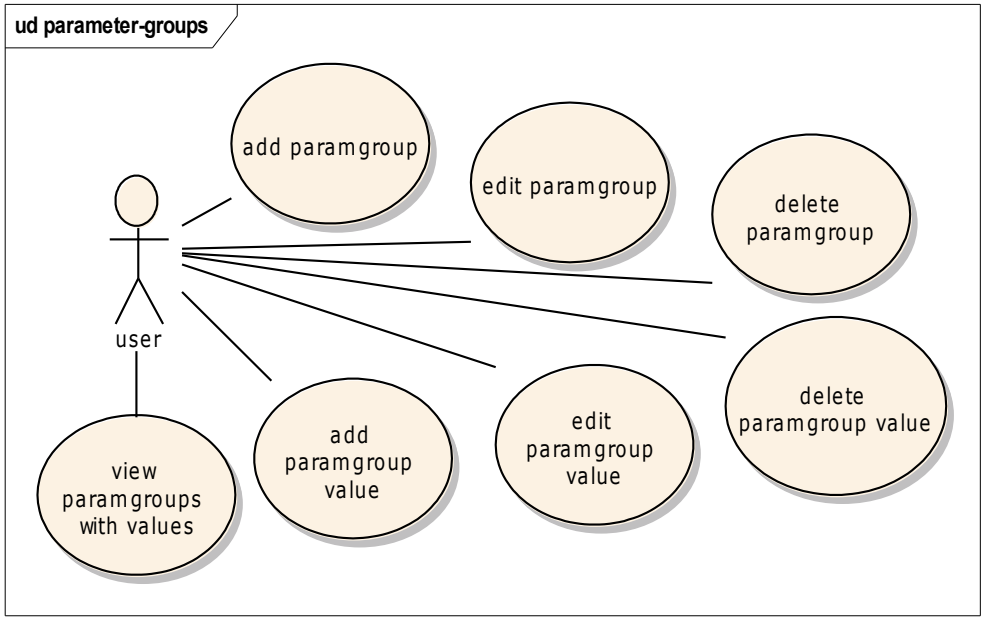
- *testovanie kompresných programov*
- *zobrazenie výsledkov testov kompresných programov*
- *generovanie štatistických porovnaní testov do rozdielnych formátov reportov*
- *obmedzenie prístupu do aplikácie len pre užívateľské účty*

Navrhnutá štruktúra užívateľskej webovej aplikácie a jej GUI pre registrovaného autora kompresných programov vychádza zo základných funkčných požiadaviek na testovanie algoritmov. Funkcie užívateľa sú rozdelené do 4 skupín:

- *funkcie pre vytvorenie nového testu kompresného programu (create new test)*
- *funkcie pre prezeranie ukončených testov kompresných programov (user-test results)*
- *funkcie pre generovanie reportov z výsledkov užívateľových testov, založených na štatistických porovnaniach dát testov (statistic data comparison)*
- *funkcie pre vytvorenie skupín parametrov, ktoré môžu použité pre popis jednotlivých testov (parameter groups)*

Stránky skupiny *parameter groups* slúžia užívateľovi na definovanie vlastných typov parametrov a zoznamov hodnôt príslušných ku každému parametru, ktoré môže užívateľ následne použiť na popis určitého testu, a tým pádom pri vytváraní reportu zo štatistických dát, môže vymedziť skupinu testov, ktoré budú použité pre tento report. Všetky definované skupiny parametrov majú rovnakú štruktúru tkz. štruktúru *parameter groups.* Táto štruktúra je implementovaná vo viacerých častiach webovej aplikácie. Štruktúra je reprezentovaná pomocou *Java* objektu *ParamgroupComponent* vo vrstve systému *main framwork* (viz kapitola 3.2.5) a príslušnými dátami uloženými v databáze (názvy parametrov, hodnoty parametrov, vzťahy medzi jednotlivými parametrami a ich hodnotami). Každý užívateľ môže využívať iba svoje vytvorené parametre s hodnotami, preto je každá štruktúra *parameter groups* v databáze jednoznačne spojená s užívateľom, ktorý ju vytvoril. V databáze uchovávajú dáta pre jednotlivé štruktúry *parameter group* tabuľky *T13_USER_PARAMGROUP, T14_PARAMGROUP, T15_PARAMGROUP_VALUE, T16_PARAMGROUP_PARAMGROUP_VALUE.*

Diagram 20 - funkcie parameter groups



Stránky skupiny *create new test* implementujú funkcie potrebné na vytvorenie nového testu, jeho spustenie a zobrazenie výsledkov testu. Keďže každý test môže byť spojený s príslušným setom preddefinovaných parametrov, jednou z inicializačných funkcií pri vytváraní testu je práve vybranie týchto parametrov a ich hodnôt z vytvorenej štruktúry *parameter groups.* Aby mohol byť vôbec test realizovaný, musí byť na server nahraný príslušný kompresný program autora so všetkými potrebnými knižnicami a konfiguračnými súbormi. Keďže každý program je navrhnutý iným autorom, neexistujú žiadne štandardné konvencie pre používanie *command-line* parametrov programov a tým pádom definovania vstupov a výstupov kompresií resp. dekompresií. Preto systém implementuje vlastnú štruktúru validácie *command-line* parametrov. Táto štruktúra je založená na informáciách o použití *command-line* parametrov programu pre kompresiu a dekompresiu priamo od autora. Kľúčovými výrazmi validácie *command*-line parametrov autor určuje požitie alebo ignorovanie jednotlivých parametrov pri realizácii kompresie resp. dekompresie.

Kľúčové výrazy kompresie:

- *archive_file – autor určuje či kompresný program vyžaduje použitie výstupného súboru kompresie*

- *compressing_file – autor určuje polohu parametru definujúceho názov vstupného (komprimovaného) súboru medzi zvyšnými command-line parametrami*

Kľúčové výrazy dekompresie:

- *archive_file – autor určuje polohu parametru definujúceho názov vstupného (zkomprimovaného) súboru medzi zvyšnými command-line parametrami*

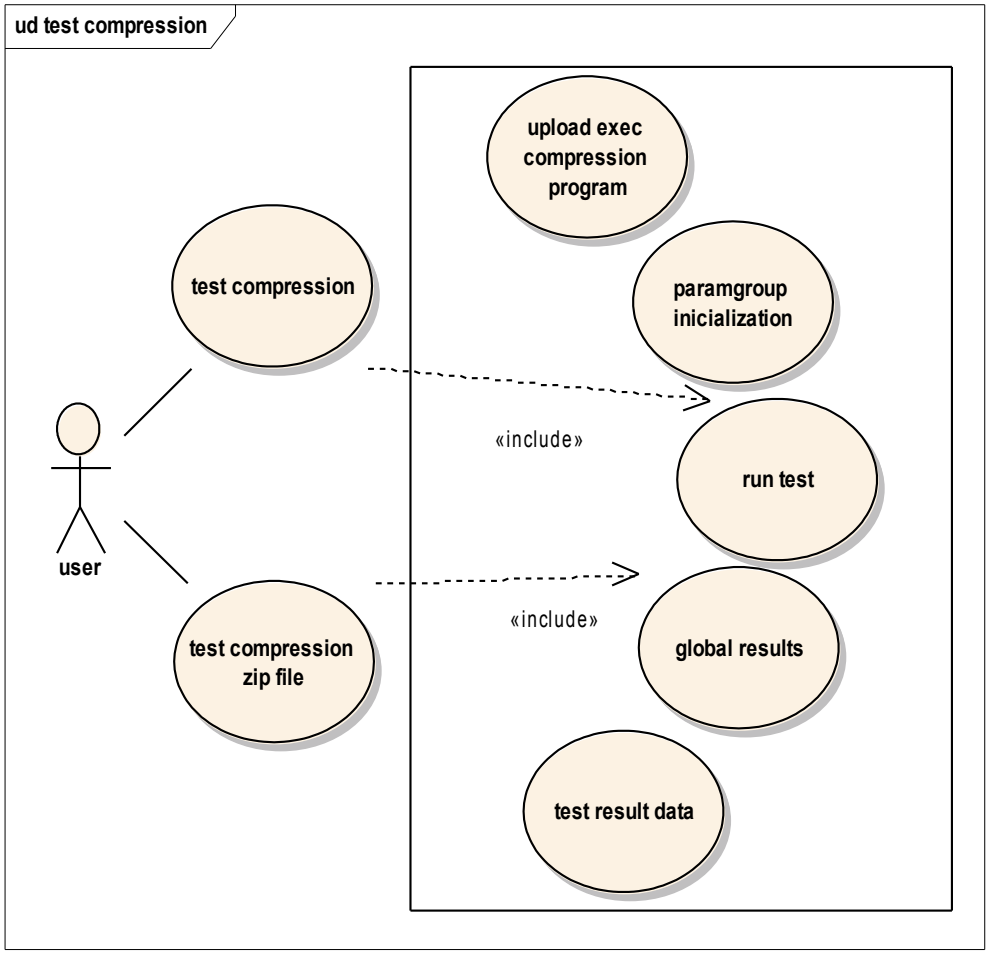
- *decomressed_file – autor určuje či dekompresný program vyžaduje použitie výstupného súboru dekompresie*

Systém podľa použitých kľúčových výrazov identifikuje vhodný *Java* objekt z *main framework* vrstvy, ktorý implementuje spôsob a postup vyhodnocovania konkrétneho testu (viz kapitola 3.2.5).

Posledným krokom inicializácie testu je výber požadovanej subkategórie korpusu, nad ktorou má byť test realizovaný. Systém použije pre zobrazenie aktuálny aktívny korpus z tabuľky databáze *T02_CORPUS_HIERARCHY*.

Kompresné alebo dekompresné programy môžu narušiť bezpečnosť a stabilitu operačného systému tým, že sa pokúsia zapisovať do nepovolených miest vo *file sytéme* servera. Z dôvodu nenarušenia bezpečnostných zásad operačného systému, *Corpus systém* využíva bezpečnostné prvky implementované aplikačným serverom *Tomcat*, založených na technológii *Java Security Manager* [10]. Bezpečnostné prvky zabraňujú kompresným a dekompresným programom používať iné adresáre ako pracovné adresáre pridelené systému *Corpus*. Rovnaké obmedzenia platia pre prístup do databáze, ktorá je blokovaná pre prístup kompresným a dekompresným programom. Pokiaľ vlastné spustenie neznámych programov od autorov nenaruší bezpečnostnú ochranu a nevyskytnú sa ďalšie neočakávané chyby pri vykonávaní testu, test bude úspešne dokončený, pričom výsledky testu budú zapísané do tabuľky *T07_TEST_RESULT* databáze, informácie popisujúce inicializačné nastavenia testu, priebeh testu a úspešné resp. neúspešné dokončenie testu budú zapísané do tabuľky *T06_TEST* databáze (popis štruktúry databáze je v kapitole 3.2.6).

Diagram 21 - funkcie create new test



Stránky skupiny *user-test results* implementujú rovnaké funkcie pre zobrazenie výsledkov autorových testov ako stránky zhodnej skupiny v administrátorskej webovej aplikácii (viz kapitola 3.2.1.1). Rozdiel v užívateľskej funkčnosti je v obmedzení zobrazovaných testov len na užívateľom vytvorené testy v porovnaní so zobrazovanými testami všetkých zaregistrovaných užívateľov v administrátorskej verzii.

Stránky skupiny *statistic data comparison* implementujú rovnaké funkcie pre štatistické porovnanie testov a vytváranie reportov z testov ako stránky zhodnej skupiny v administrátorskej webovej aplikácii (viz kapitola 3.2.1.1). Rozdiel v užívateľskej funkčnosti je v obmedzení práce len na testy, ktoré boli vytvorené daným užívateľom, na rozdiel od neobmedzenej práce s testami všetkých užívateľov, ktorú ma k dispozícii admin v administrátorskej verzii.

3.2.4 Registračný formulár

Registračný formulár pre autorov kompresných programov je webová aplikácia pridružená k hlavnej užívateľskej webovej aplikácii. Na rozdiel od administrátorskej a užívateľskej aplikácie, registračný formulár nepoužíva žiadne autentifikačné funkcie pre prístup užívateľa k nástrojom formuláru, pretože jeho hlavnou úlohou je zaslať žiadosť o registráciu nového užívateľa adminovi, čiže registráciu doposiaľ neregistrovaného autora.

Systém odosiela dáta o žiadosti do tabuľky *T11*_REGISTRATION databáze, a taktiež zasiela na definované e-mailové adresy adminov upozornenie o vytvorení novej žiadosti o registráciu.

3.2.5 Vrstva main framework

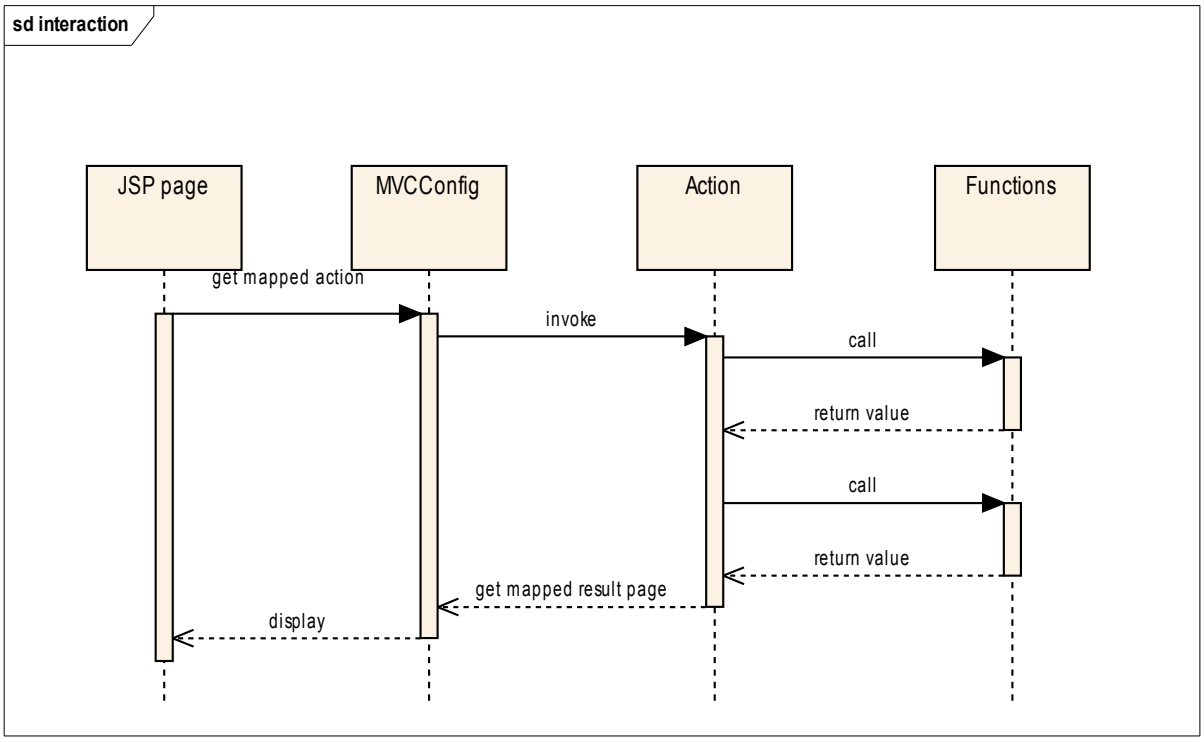
Main framework predstavuje druhú najvyššiu vrstvu v kompletnej štruktúre *Corpus systému,* tvorí funkčné jadro systému, v ktorom sú pomocou jazyka *Java* a pomocných konfiguračných súborov vytvorené všetky najdôležitejšie štruktúry systému, implementované v najvyššej vrstve prístupnej pre užívateľov – vo webovej aplikácii. *Main framework vrstva* spája najvyššiu vrstvu s dvoma najnižšími vrstvami reprezentujúce dáta systému – databáza a *file systém*. Aby mohli byť tieto dáta zobrazené užívateľovi, musia byť transformované do prijateľnej formy. K tomuto účelu systém využíva špeciálne navrhnuté štruktúry, pričom hlavnými požiadavkami, okrem vlastnosti uľahčiť vývoj jednotlivých nástrojov systému, ktoré musia všetky štruktúry spĺňať sú abstraktnosť a nezávislosť od iných objektov, vlastne definujú možnosť prípadného rozšírenia jednotlivých štruktúr (objektov) o ďalšie funkcie, potrebné pri riešení nových problémov pomocou *Corpus systému* v budúcnosti.

- *action mapping – štruktúra navrhnutá pre navigáciu medzi stránkami webovej aplikácie*
- *corpus hierarchy – štruktúra navrhnutá pre kooperáciu virtuálnej štruktúry korpusu, fyzickej reprezentácie korpusu a mapovacích dát z databáze*
- *file treeview – štruktúra reprezentujúca usporiadanie fyzických súborov korpusu na file systéme*
- *test – štruktúra navrhnutá pre vykonávanie jednotlivých kompresných programov*
- *parameter groups – štruktúra reprezentujúca jednotlivé užívateľské skupiny parametrov s príslušnými zoznamami hodnôt*
- *sort – štruktúra navrhnutá pre porovnávanie testov podľa rôznych štatistických metód*
- *page components – štruktúry pre zobrazovanie nástrojov webovej aplikácie na jednotlivých stránkach aplikácie*
- *locks – štruktúra zámkov použitých v aplikácii*

Návrh *Action mapping* štruktúry uľahčuje zložité presmerovanie medzi stránkami aplikácie po úspešnom alebo neúspešnom vykonaní niektorej z funkcií nástrojov systému. Jej abstraktné riešenie počíta s možnosťou rozšírenia systému o ľubovoľný počet nových stránok bez významného zásahu do jadra systému a so zanedbateľným dopadom na ďalšie objekty *main framu,* databáze a webovej aplikácie. Každá stránka webovej aplikácie ktorá implementuje funkčné nástroje, je reprezentovaná *Java* objektom akcie. Mapovanie stránok na ich reprezentáciu pomocou *Java* objektu je definované v konfiguračnom *XML* súbore *mvc-config.xml*. Štruktúra dokumentu *mvc-config* je popísaná v kapitole 3.2.1. Akcie ďalej určujú, ktoré nasledujúce stránky sa majú zobraziť po dokončení aktuálnej akcie, čiže každý proces funkčnej komponenty na stránke, musí byť definovaný v konkrétnom objekte akcie. Akcie vlastne predstavujú prvý krok z pohľadu *main frameworku* pri prechode z vizuálneho

zobrazenia funkčných komponent na stránkach k vykonávaniu týchto funkcií v jazyku *Java.* Napr. validačné funkcie vstupných polí na stránkach sú definované v objekte akcie, ktorá danú stránku reprezentuje. Podľa typu návratovej hodnoty validačných funkcií sa akcia rozhodne, ktorou stránku zobrazí ako nasledovnú, ako reakciu na výsledok validácie (napr. pri neúspešnej validácii zobrazí chybovú stránku).

Diagram 22 - action mapping



Štruktúru akcií tvorí abstraktná trieda *AbstractAction*, ktorá implementuje interface *MVCActionInterface*. Z dôvodu korektnej implementácie akcií, každá akcia musí byť potomkom tejto abstraktnej triedy. Jednotlivé inštancie akcií sa vytvoria pri počiatočnej inicializácií aplikácie parsovaním *XML* dokumentu *mvc-config.* Po vytvorení štruktúry akcií a ich mapovaní na jednotlivé stránky, je celá štruktúra uložená do pamäti, čím je odstránená opätovná náročná časová réžia pre jednotlivé mapovania stránka – akcia, ktorá je nutná pri parsovaní *XML* dokumentu *mvc-config.*

Štruktúra konfiguračného dokumentu mapovania stránka – akcia pomocou *DTD* jazyka*:*

```
<!ELEMENT mvc-config (basic, action-mappings)>
<!ELEMENT basic EMPTY>
<!ELEMENT action-mappings (action+)>
<!ELEMENT action (forward+)>
<!ELEMENT forward (#PCDATA)>
<!ATTLIST basic
     url CDATA #REQUIRED
>
<!ATTLIST action
     name CDATA #REQUIRED
     class CDATA #REQUIRED
>
<!ATTLIST forward
     name CDATA #REQUIRED
     path CDATA #REQUIRED
     redirect (true) #IMPLIED
     over CDATA #IMPLIED
>
<!ATTLIST waiting
     name CDATA #REQUIRED
     on CDATA #IMPLIED
     off CDATA #IMPLIED
>
```

- *mvc-config je root element dokumentu*
- *action-mappings element zoskupuje všetky typy akcií*
- *action element reprezentuje mapovanie stránky na akciu*
- *parameter name elementu action definuje názov akcie použitej na JSP stránke*
- *parameter class elementu action definuje objekt v Jave reprezentujúci danú akciu*
- *forward element reprezentuje presmerovanie aktuálnej stránky na nasledujúcustránku v závislosti od vykonanej funkcie a valídneho výsledku tejto funkcie*
- *parameter path elementu forward definuje názov akcie (stránky), ktorá sa zobrazí po dokončení funkcie aktuálnej akcie*
- *parameter name elementu forward definuje návratovú hodnotu jednej z funkcií prístupnej aktuálnou akciou*

Vlastný *XML* dokument s použitím základných elementov:

```
<mvc-config>
     <basic url=""/>
     <action-mappings>
          <action name="actionName" class="className">
               <forward name="functionResultName"
                        path="nextActionName"
                        redirect="true"/>
          </action>
     </action-mappings>
</mvc-config>
```

Štruktúra *Corpus hierarchy* je navrhnutá pre uľahčenie spracovania korpusu (z kapitoly 2) v *main frameworku* a pre efektívnu implementáciu operácií nad korpusom. Medzi tieto operácie patria hlavne operácie na modifikáciu štruktúry korpusu, operácie na prepojenie virtuálnej štruktúry korpusu s konkrétnymi dátami a operácie s korpusom potrebné pri realizácii kompresných testov. Najvýhodnejšou variantov je reprezentácia korpusu v *XML* jazyku – tkz. virtuálna reprezentácia korpusu. Popis *XML* štruktúry v jazyku *DTD* je následovný:

```
<?xml version="1.0" encoding="windows-1250"?>
<!ELEMENT CORPUS_HIERARCHY (CATEGORY*, DATA*)>
<!ELEMENT CATEGORY (CATEGORY*, DATA*)>
<!ELEMENT DATA (#PCDATA)>
<!ATTLIST CATEGORY
     name CDATA #REQUIRED
>
<!ATTLIST DATA
     name CDATA #REQUIRED
     source_id CDATA #REQUIRED
>
```

- *CORPUS_HIERARCHY je root element dokumentu*
- *CATEGORY element predstavuje kategóriu korpusu, uzol stromovej štruktúry*
- *Parameter name elementu CATEGORY definuje názov kategórie*
- *DATA element reprezentuje súbor korpusu, list stromovej štruktúry*
- *Parameter name elementu DATA definuje názov súboru, fyzických dát spojených s daným listom stromovej štruktúry korpusu*
- *Parameter source_id elementu DATA reprezentuje ID záznamu v tabuľke mapovania súbor – list korpusu*

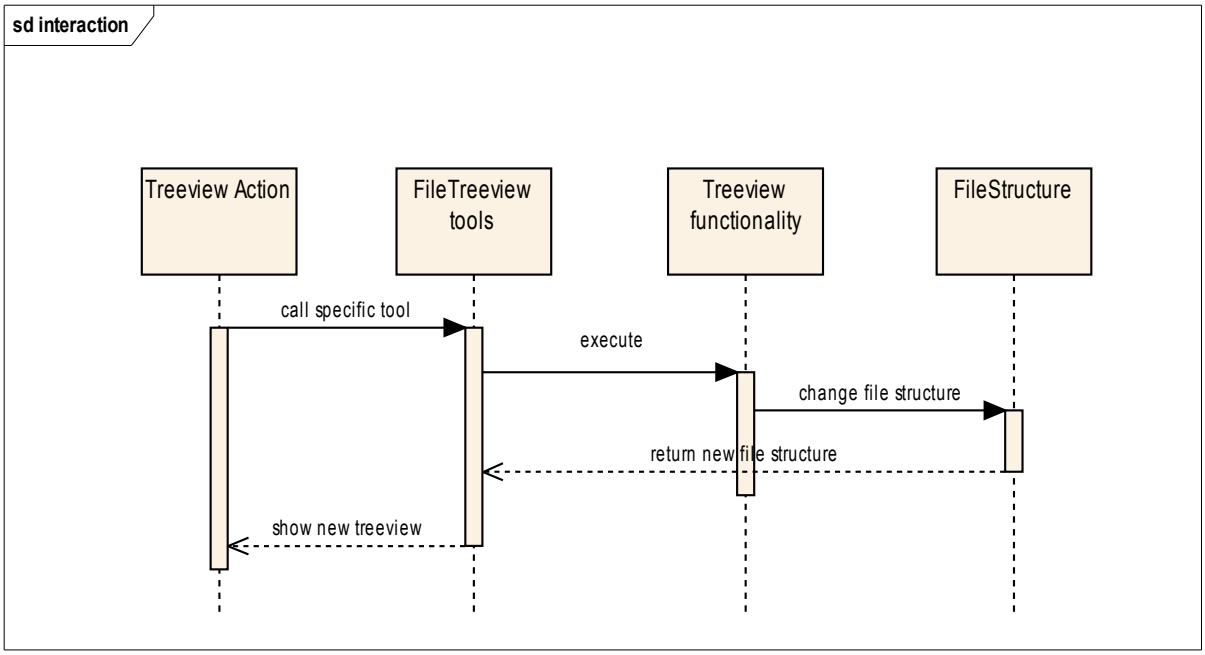
Vlastný *XML* dokument s použitím základných elementov:

```
<CORPUS_HIERARCHY>
    <CATEGORY name=”categoryName”>
        <DATA name=“fileName“ source_id=“referenceID“/>
    </CATEGORY>
</CORPUS_HIERARCHY>
```

Všetky dostupné operácie s korpusom sú definované v *Java* objekte *CorpusHierarchyUtil.* Táto trieda rozhoduje, ktoré funkcie je potrebné vyvolať pre korektnú realizáciu danej operácie. Funkcie korpus operácií sú rozdelené do 3 základných typov – databázové funkcie definuje trieda *CorpusHierarchyDb* a *CorpusHierachySource*, funkcie zobrazujúce korpus na stránkach definuje trieda *DrawCategoryList*, zvyšné funkcie realizujúce napr. validáciu, upload, download virtuálnej korpus štruktúry sú definované priamo v triede *CorpusHierarchyUtil.* Špecifická operácia korpusu, mapovanie virtuálnej štruktúry na konkrétne dáta korpusu, je implementovaná v tabuľke *T03_CORPUS_HIERARCHY_SOURCE,* kde sú spojené ID referencie (hodnota parametra *source_id*) s konkrétnym súborom na *file systéme* (pole *source_path* v SQL tabuľke). Každý list korpusú môže byť spojený maximálne s jedným súborom na *file systéme.* Každý súbor na *file systéme* môže byť namapovaný na 0 – N listov korpusu. Z dôvodu implementácie operácie mapovania v databáze sú funkcie tejto operácie podmnožinou databázových operácií, definuje ich *Java* objekt *CorpusHierachySource.*

Ďalšiu významnú štruktúru použitú vo vrstve *main framework* reprezentuje štruktúra *file treeview*. Jej implementácia uľahčuje správu dát na *file systéme* a ich priame pridávanie do virtuálnej štruktúry korpusu. Podobne ako štruktúra *action mapping*, *file treeview* obsahuje objekty spojujúci vizuálne zobrazenie nástrojov s ich funkčnou definíciou. Z tohto dôvodu je štruktúra reprezentovaná objektami *FileTreeview*, ktorý zobrazuje nástroje pre správu adresárovej štruktúry dát korpusu na stránke, *Treeview* ktorý obsahuje funkcie realizujúce jednotlivé nástroje a objekt *FileStructure*, ktorý priamo reprezentuje aktuálnu adresárovú štruktúru dát.

Diagram 23 - file treeview



Z pohľadu požiadaviek na systém je najdôležitejšou implementovanou štruktúrou *test*, v ktorej sú definované všetky funkcie potrebné pri realizácii jednotlivých testov kompresných algoritmov. Celá štruktúra sa nachádza v balíku *corpus.web.user.util.test.* Organizácia objektov v balíku je rozdelená na 2 skupiny typov objektov, prvú tvoria objekty, ktorých funkcie slúžia na upload autorových programov na server, druhú reprezentujú objekty, ktorých funkcie spracujú argumenty kompresných (dekompresných) programov. Systém v aktuálnej verzii podporuje upload dvoma spôsobmi - upload samostatného kompresného programu bez knižníc a konfiguračných súborov (funkcie definované pomocou abstraktnej triedy *AbstractTest*) a upload kompresného programu s knižnicami a konfiguračnými súbormi v balíčku vo vnútri *zip* súboru (funkcie definované v abstraktnej triede *AbstractTestZip*). Pridanie podpory nových formátov uploadovaných súborov do systému je jednoduché na implementáciu, z dôvodu modulárnosti štruktúry *test*. Stačí pridať novú abstraktnú triedu, ktorá bude obsahovať funkcie pre upload súboru, prípadne funkcie na rozbalenie súborov vo vnútri uplodovaného súboru. Taktiež skupinu objektov validujúcu argumenty kompresných programov je možné rozšíriť o viacej podporujúcich formátov. V aktuálnej verzii sú podporované formáty argumentov:

- *kompresný program vytvorí zo vstupného súboru nový výstupný súbor, vyžaduje použitie argumentov input_file a output_file*
- *kompresný program prepíše vstupný súbor výstupným súborom, vyžaduje použitie argumentov input_file*
- *dekompresný program vytvorí zo vstupného súboru nový výstupný súbor, vyžaduje použitie argumentov input_file a output_file*
- *dekompresný program prepíše vstupný súbor výstupným súborom, vyžaduje použitie argumentov input_file*

Typ použitého formátu argumentov pre kompresný program určuje akým spôsobom má byť vykonávaný samotný test. Keďže nemôžu byť zmenené zdrojové dáta korpusu, v prípade programov, ktoré prepíšu vstupný súbor výstupným, musí systém špeciálne vytvoriť kópie zdrojových dát korpusu s ktorými bude program pracovať. Vyhodnotenie priebehu testu a uloženie výsledkov do databáze je rovnaké pre všetky typy postupov testovania programov.

Štruktúra *parameter groups* je použitá na vytváranie vlastných užívateľom definovaných parametrov, ktoré rozširujú popis testov. Štruktúra je rozdelená do dvoch úrovní, kde prvú úroveň tvoria názvy parametrov a druhú hodnoty parametrov patriacich konkrétnemu parametru. Z týchto dvoch typov úrovní vyplýva 2 objektový základ štruktúry – zoznam parametrov, kde každý objekt parametru vlastní zoznam hodnôt. Základ štruktúry je definovaný v objekte *ParamgroupComponent.* Systém realizuje rôzne operácie nad *parameter groups* štruktúrou, napr. operácie zobrazenia, operácie modifikácie parametrov a ich hodnôt, operácie potrebné pri vytváraní štatistických reportov. Keďže všetky dáta, ktoré popisujú skupiny parametrov s hodnotami sú uložené v databáze v tabuľkách *T13_USER_PARAMGROUP, T14_PARAMGROUP, T15_PARAMGROUP_VALUE, T16_PARAMGROUP_PARAMGROUP_VALUE, T17_TEST_PARAMGROUP* (viz kapitola 3.2.6), do množiny operácii nad štruktúrou *parameter groups* patria aj databázové operácie. Každá operácia je definovaná funkciami v objektoch jazyka *Java.* O vizuálne zobrazenie skupín parametrov a ich nástrojov na stránkach sa starajú objekty *DrawParamgroupLIst, ParamgroupOperation*. Databázové funkcie obsahujú objekty *UserParamgroupDb, TestParamgroupDb, ParamgroupParamgroupValueDb,* . Validačné funkcie sú definované v objekte *ParamgroupsUtil.* Spomenuté funkcie štruktúry patria do pomocných funkcií, ktoré dopĺňajú hlavný dôvod vytvorenia štruktúry *parameter groups,* ktorým je výber požadovaných testov do štatistického reportu. Konkrétny parameter s konkrétnou hodnotou definovaný pre konkrétny riadok tabuľky reportu, zoskupuje výsledky všetkých testov, ktoré boli vytvorené s touto hodnotou parametra. Parameter a jeho hodnota definovaný pre konkrétny stĺpec tabuľky takisto zoskupuje výsledky všetkých testov, obsahujúcich danú hodnotu parametra. Hodnoty v bunkách tabuľky reprezentujú priemernú hodnotu všetkých testov spĺňajúcich obmedzujúce podmienky v danej bunke.

Príklad.

Tabuľka testov s parametrami, ktoré ich popisujú:

test	param **kompresia** hodnota **gzip**	param **kompresia** hodnota **bzip**	param **jazyk** hodnota **EN**	param **jazyk** hodnota **CZ**
Test1	X	X	X	X
Test2	X			X
Test3	X			X

Zadefinovanie formátu výstupnej tabuľky v štatistickom reporte:

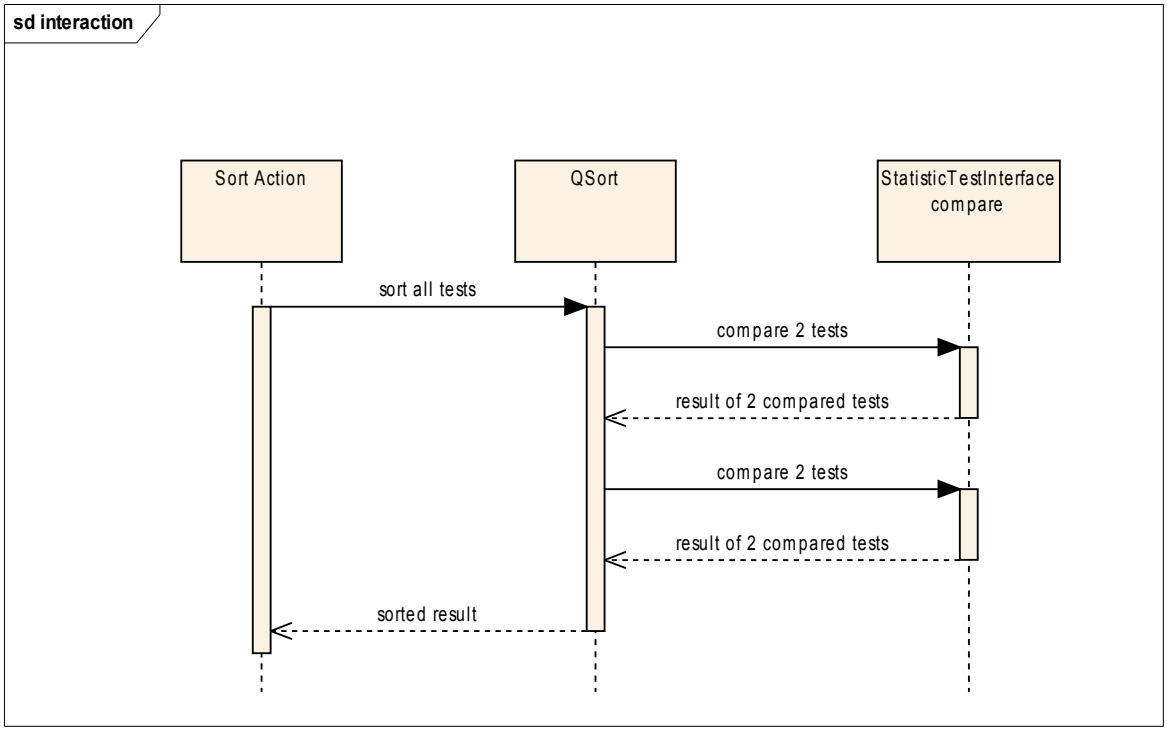
- *1. riadok bude obmedzený na parameter A s hodnotou a1*
- *2. riadok bude obmedzený na parameter A s hodnotou a2*
- *1. stĺpec bude obmedzený parametrom B s hodnotou b1*
- *2. stĺpec bude obmedzený parametrom C s hodnotou c1*

Výsledná tabuľka v štatistickom reporte:

	jazyk – EN	**jazyk – CZ**
kompresia – gzip	Test1	Test1, Test2, Test3
kompresia – bzip	Test1	Test1

Štruktúra navrhnutá pre porovnávanie jednotlivých testov rôznymi štatistickými metódami sa volá *sort.* Jej abstraktné riešenie umožňuje v ďalších verziách implementáciu nových typov štatistických metód, podľa príslušných požiadaviek. V aktuálnej verzii *Corpus systému* je implementovaný *dvojvýberový mediánový test* zo skupiny neparametrických metód [11] (viz kapitola 4). Každý štatistický test je reprezentovaný *Java* objektom, ktorý implementuje *StatisticTestInterface.* Jednotlivé objekty testov porovnávajú vždy výsledky práve 2 testov. Pre zoradenie testov od najefektívnejšieho po najmenej efektívny je implementovaný triediaci algoritmus *QSort,* v *Java* objekte QSort. QSort objekt realizuje porovnávanie jednotlivých testov pomocou volania konkrétneho objektu štatistického testu napr. *MedianTest*.

Diagram 24 - sort



3.2.6 Databáza

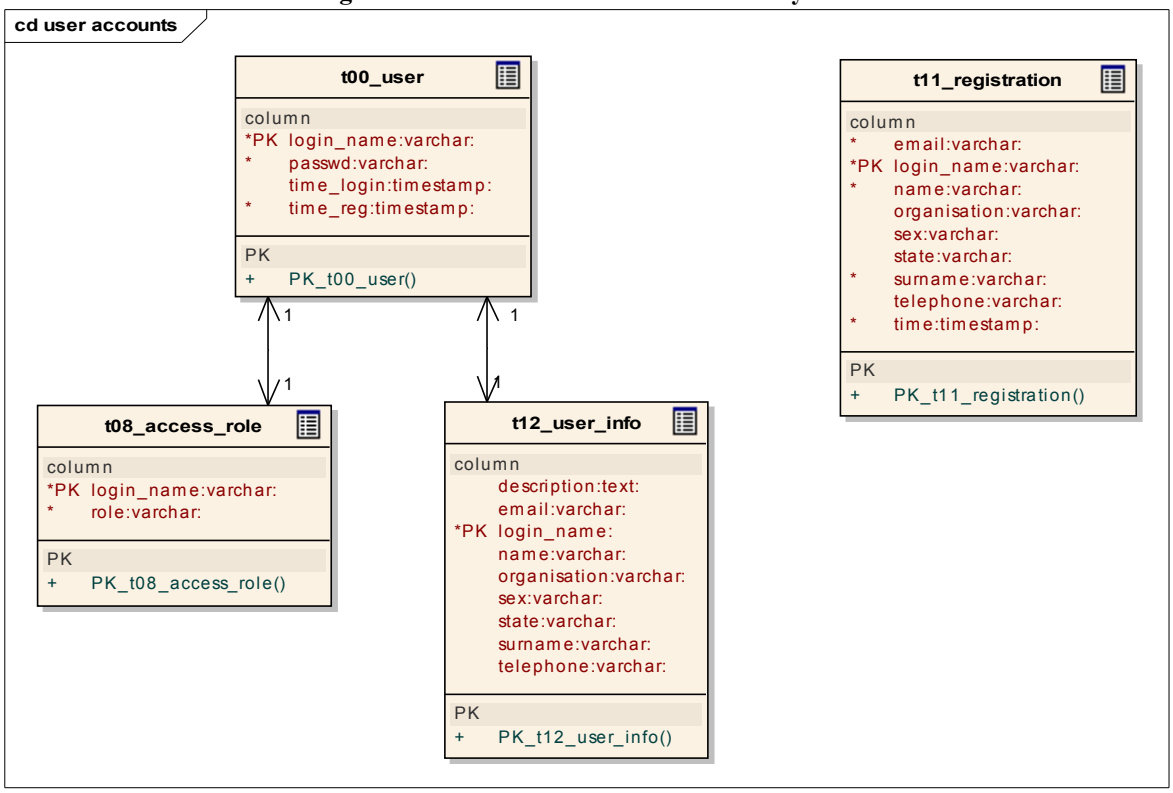
Tretia vrstva *Corpus systému* je zameraná na uchovávanie dát v SQL databáze. Z funkcií systému vyplývajú nasledujúce požiadavky na typ dát uložených v databáze:

- *dáta potrebné pre správu užívateľských účtov*
- *dáta potrebné pre správu korpus hierarchie*
- *dáta jednotlivých testov a ich výsledkov*
- *dáta skupín užívateľských parametrov*

Tabuľky databáze uchovávajúce dáta užívateľských účtov majú jednoduchú štruktúru, v ktorej najvýznamnejšiu úlohu majú dáta definujúce skupinu prístupových práv užívateľa do systému, ktorou môže byť buď administrátorský alebo užívateľský prístup, vyjadrený hodnotou *role (corpus_admin, corpus_user)*. Tabuľky s užívateľskými dátami sú:

- *T00_USER – obsahuje prihlasovacie informácie potrebné pre prihlásenie do systému daným užívateľom, a doplňujúce informácie o dátume posledného prihlásia daného užívateľa do systému, taktiež dátum registrácie*
- *T08_ACCESS_ROLE – definuje vzťah medzi užívateľom a jeho prístupovými právami*
- *T12_USER_INFO – obsahuje doplňujúce informácie o konkrétnom užívateľovi*
- *T11_REGISTRATION – obsahuje informácie z formuláru žiadosti o registráciu nového užívateľa do systému*

Diagram 25 - štruktúra tabuliek užívateľských dát

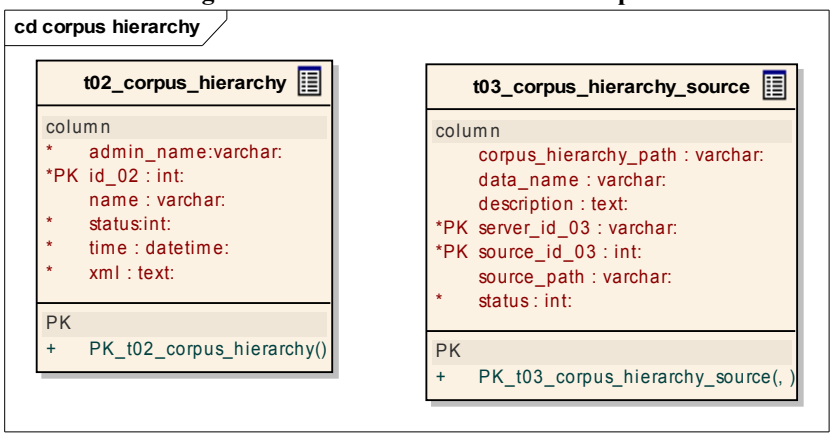


Tabuľky databáze uchovávajúce dáta potrebné pre správu korpusu, musia byť schopné uložiť virtuálnu štruktúru korpusu, ktorý je v *XML* formáte. Na uloženie *XML* štruktúry slúži dátový typ *TEXT*. Do tejto skupiny tabuliek databáze patria nasledujúce tabuľky:

- *T02_CORPUS_HIERARCHY – uchováva jednotlivé vytvorené verzie virtuálnej štruktúry korpusu, pričom obsahuje doplňujúce dáta o adminovi, ktorý vytvoril alebo modifikoval poslednú verziu korpusu, dátum a čas poslednej verzie*

- *T03_CORPUS_HIERARCHY_SOURCE – definuje mapovanie listov stromovej štruktúry korpusu na konkrétny súbor umiestnený na file systéme*

Diagram 26 - štruktúra tabuliek dát korpusu



Tabuľky databáze uchovávajúce inicializačné dáta jednotlivých testov, samotné kompresné programy a výsledky testov, musia byť schopné uložiť súbor reprezentujúci kompresný program autora, alebo zip súbor obsahujúci kompresný program so všetkými potrebnými pomocnými súbormi. Tieto súbory sú uložené pomocou typu CLOB/BLOB. Medzi tabuľky ktoré uchovávajú dáta jednotlivých testov patria:

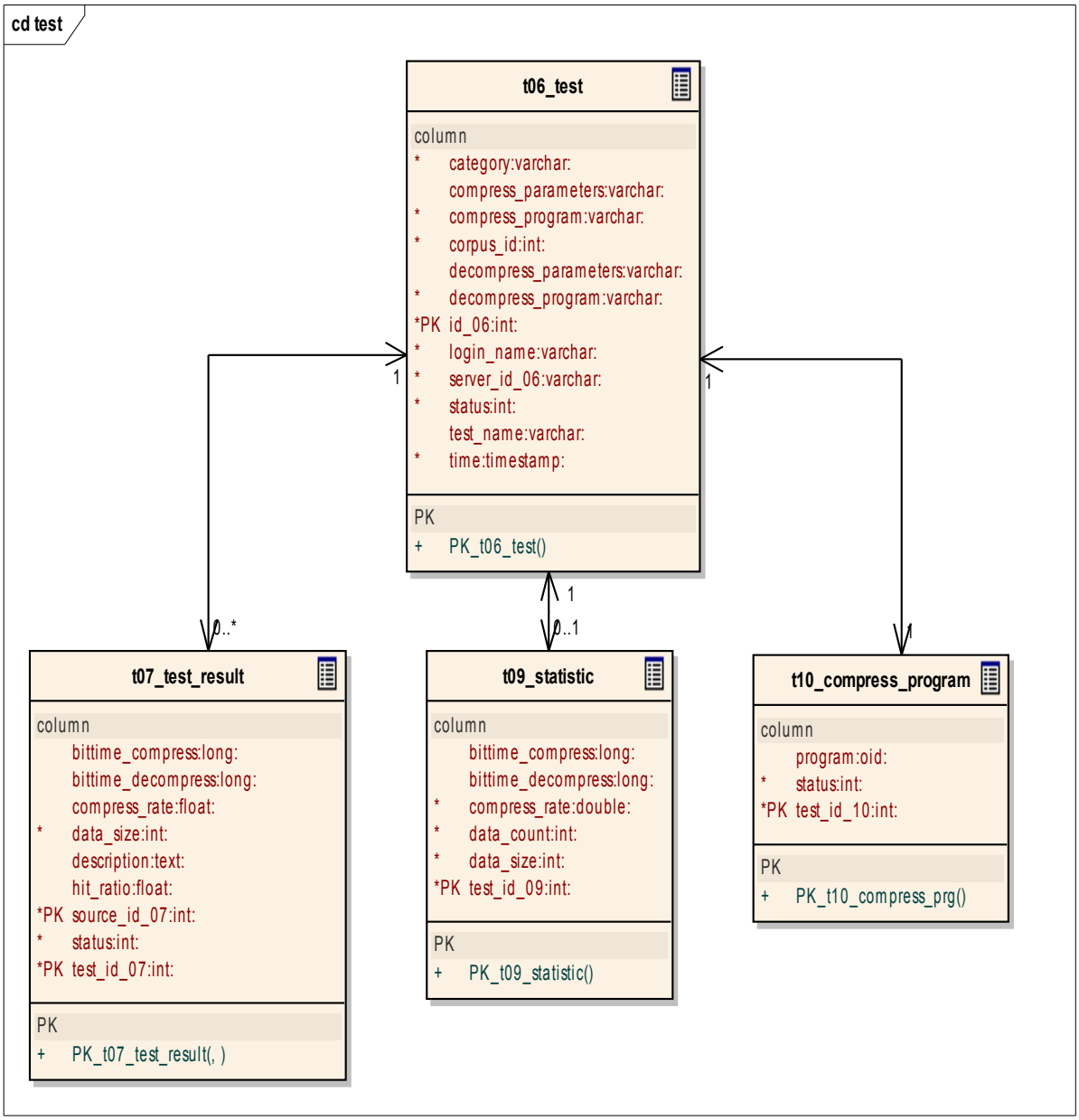
- *T06_TEST – uchováva inicializačné dáta testu, užívateľa ktorý vytvoril daný test, dátum a čas kedy bol test realizovaný*

- *T07_TEST_RESULT – uchováva výsledky kompresie (dekompresie) pre každý súbor korpusu, nad ktorým bol test vykonaný, čiže obsahuje rýchlosti kompresie a dekompresie, kompresný pomer a status úspešnosti testu*

- *T09_STATISTIC – uchováva výsledky kompresie (dekompresie) pre celú časť korpusu, nad ktorou bol test vykonaný, čiže uchováva priemerné hodnoty kompresie, dekompresie a kompresného pomeru vzhľadom na počet otestovaných súborov*

- *T10_COMPRESS_PROGRAM – uchováva súbor kompresného programu, alebo zip súbor obsahujúci vlastný kompresný program a zvyšné pomocné knižnice potrebné pre správny beh programu*

Diagram 27 - štruktúra tabuliek s dátami testov



Tabuľky uchovávajúce dáta skupín užívateľmi definovaných parametrov majú jednoduchú štruktúru, keďže uchovávajú iba názvy všetkých vytvorených parametrov s ich hodnotami a vzťahy spájajúce užívateľa s jeho definovanými parametrami, a ďalej každý parameter s jeho zoznamom príslušných hodnôt. Nasledujúce tabuľky slúžia pre uchovanie dát *parameter groups:*

- *T13_USER_PARAMGROUP – definuje vzťah spájajúci konkrétneho užívateľa so svojimi skupinami parametrov*
- *T14_PARAMGROUP – uchováva názvy všetkých skupín parametrov, vytvorených užívateľmi*

- *T15_PARAMGROUP_VALUE – uchováva názvy všetkých hodnôt vytvorených užívateľmi*

- *T16_PARAMGROUP_PARAMGROUP_VALUE – definuje vzťah spájajúci konkrétnu skupinu parametrov so svojimi hodnotami*

- *T17_TEST_PARAMGROUP – definuje vzťah spájajúci konkrétny test so skupinou parametrov a hodnotou tohto parametra, ktorý bol použitý pri inicializácii testu*

Diagram 28 - štruktúra tabuliek s dátami skupín parametrov

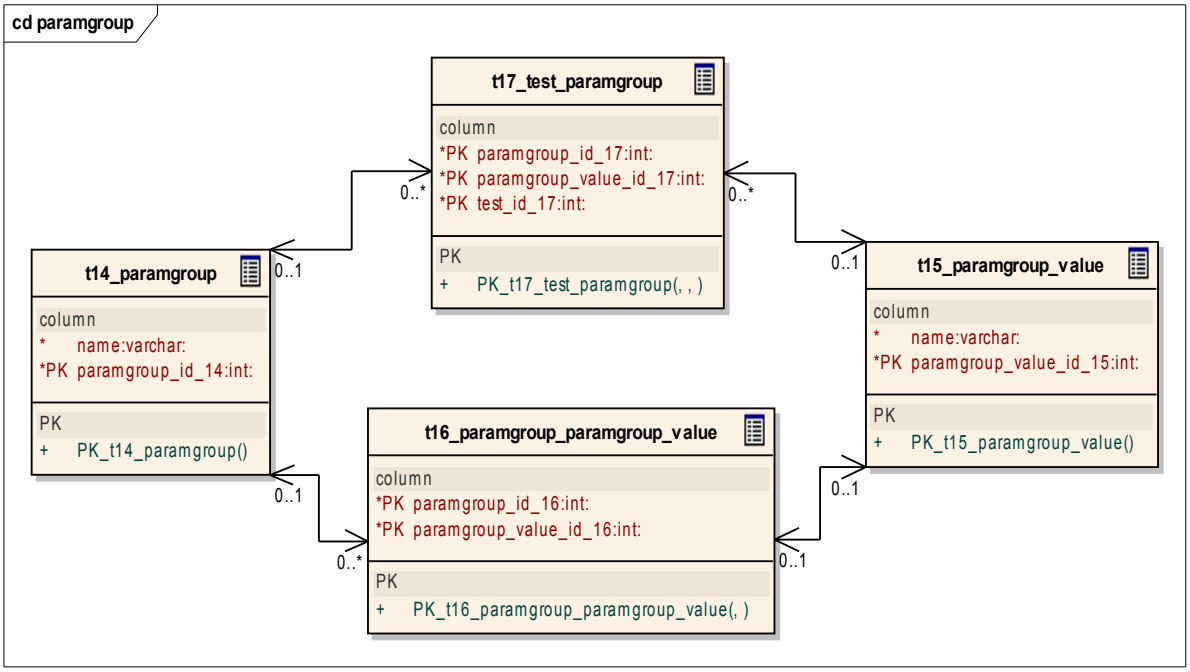


Diagram 29 - vzťah user - paramgroup

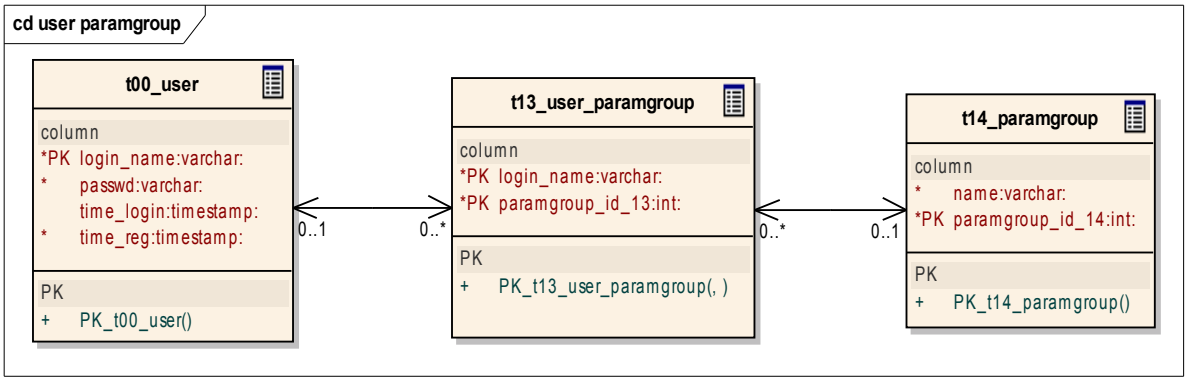
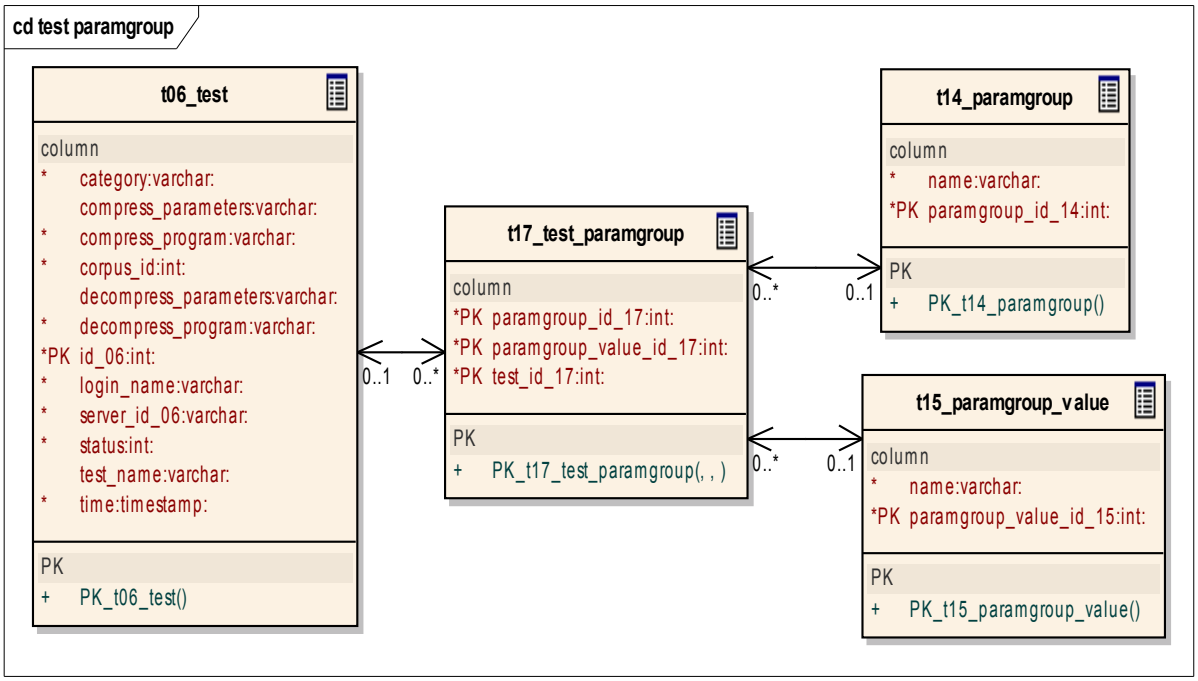


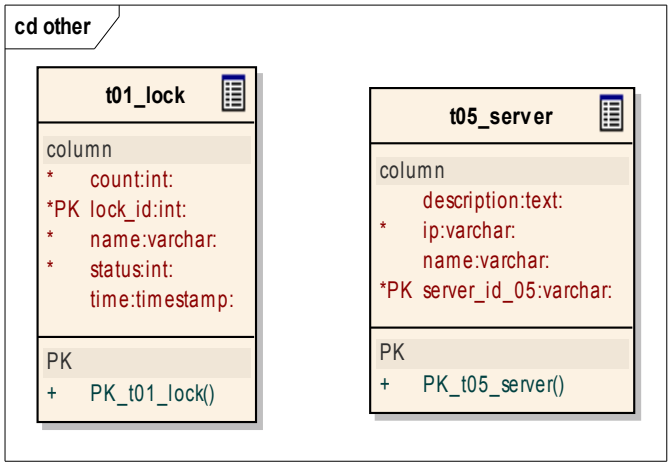
Diagram 30 - vzťah test - paramgroup



Posledným tabuľkami databáze sú nasledujúce dve pomocné tabuľky:

- *T01_LOCK – uchováva aktuálne stavy aktívnych zámkov použitých v systéme Corpus pre obmedzenie prístupu k dátam korpusu, k virtuálnej štruktúre korpusu a k dátam prebiehajúcich testov*
- *T05_SERVER – uchováva informácie o serveroch, na ktorých sú umiestnené fyzické dáta korpusu*

Diagram 31 - štruktúra pomocných tabuliek



3.2.7 Dáta korpusu na file systéme

Posledná vrstva *Corpus systému* obsahuje fyzické dáta korpusu, ktoré sú umiestnené na *file systéme* servera, na ktorom je nainštalovaná webová aplikácia systému. Dôvodom rovnakého umiestnenia aplikácie a dát korpusu je odstránenie nezanedbateľných časových nárokov pri prípadnom kopírovaní dát (súborov) korpusu do dočasných adresárov, nutných pri realizácii testov niektorých kompresných programov. Virtuálna štruktúra korpusu je mapovaná na fyzické súbory korpusu pomocou databáze, tabuľky *T03_CORPUS_HIERARCHY_SOURCE* (viz kapitola 3.2.1.1). Štruktúra rozmiestnenia súborov korpusu vo vnútri *root* adresára pre dáta korpusu nie je závislá od jeho virtuálnej štruktúry, pretože všetky potrebné informácie na získanie dát pre konkrétny koncový uzol stromovej štruktúry korpusu sú uchované v spomínanej tabuľke mapovania.

3.2.8 Analýza, návrh a vývoj aplikácie

Pri analýze a návrhu nových rozsiahlejších systémov je veľmi efektívne použitie modelovacieho jazyka systémov *UML* (Unified Modeling Language) [12]. Analýza a návrh v tomto jazyku pomáhajú odstrániť nežiaduce nedorozumenia medzi zadávateľom požiadaviek a realizátorom projektu už v počiatkoch vývoja systému, a tým pádom odstraňuje časovo náročné korektúry chýb v neskorších častiach vývoja systému, alebo dokonca až po jeho dokončení. Z tohto dôvodu bol jazyk *UML* použitý pri analýze požiadaviek a návrhu *Corpus systému.*

Keďže vývoj systému bol realizovaný aj s možnosťou jeho ďalšieho rozšírenia o niektoré komponenty a funkcionality v budúcnosti, počas prebiehajúceho vývoja bolo veľmi potrebné vyvárať verzie kompletných funkčných častí systému, ku ktorým by bola možnosť vždy sa vrátiť, pri vyskytnutí neriešiteľných problémov vo vývoji niektorej z funkcionalít systému. Z tohto dôvodu bol pri vývoji *Corpus systému* použitý systém pre správu verzií projektu *CVS* (Concurrent Version System) [13].

Kapitola 4

Štatistická metóda

Dôležitou časťou práce je správny výber štatistickej metódy, ktorá je použitá na porovnávanie výsledkov jednotlivých testov kompresných algoritmov, pričom výsledky testov reprezentujú dáta, zozbierané pri vykonávaní kompresie a dekompresie nad konkrétnymi súbormi – fyzickými dátami korpusu. Do štatistických výsledkov sú zahrnuté len úspešné testy, tzn. testy, pri ktorých vykonávaní nedošlo k žiadnej funkčnej alebo technickej chybe a zároveň výsledný súbor dekompresie bol zhodný so vstupným súborom kompresie.

Kompresný pomer, rýchlosť kompresie a rýchlosť dekompresie sú tri sledované hodnoty, ktoré sú zaznamenané pre každý súbor nad ktorým je vykonávaný daný test. Kompresný pomer predstavuje hodnotu podielu zdrojového súboru pred kompresiou a jeho komprimovanou verziou. Čas kompresie je hodnota predstavujúca počet bytov pôvodného súboru komprimovaných za 1s. Čas dekompresie je taktiež hodnota, ktorá predstavuje počet bytov pôvodného súboru komprimovaných za 1s. Štatisticky je možné porovnať testy podľa všetkých troch veličín.

4.1 Medián test

Výsledné hodnoty sledovaných veličín testu kompresného programu nad danou časťou korpusu sú ovplyvnené najmä veľkosťou a typom súboru, nad ktorým je test realizovaný. Keďže súbory v jednotlivých kategóriách korpusu nie sú striktne rozčlenené podľa presných veľkostí súborov, výsledné hodnoty testu sú pre každý súbor rozdielne a nespĺňajú predpoklady štatistických metód o rozdelení dát. Z tohto dôvodu nie je možné aplikovať parametrické metódy. Možnou aplikovateľnou triedou štatistických metód, ktoré nepredpokladajú konkrétne rozdelenie a vyhovujú spracovaniu dát získaných z výsledkov testov, je trieda neparametrických metód. Výsledné dáta testov taktiež spĺňajú ďalšiu podmienku, ktorá je nutná pre použitie neparametrických metód – určovanie na základe veľkého rozsahu výberu, ktorý je docielený množstvom dát v korpuse rádovo tisícov až desaťtisícov.

Skupina neparametrických metód obsahuje viacero typov testov, ktoré sú vhodné pre použitie nad konkrétnou skupinou dát. Pre porovnanie dvoch testov je potrebné porovnať dáta z dvoch nezávislých výberov, čiže výsledné dáta z prvého a výsledné dáta z druhého testu. Do skupiny neparametrických metód a dvojvýberových testov patria napr. testy *Wilcoxonov test, Van der Waerdenov test, Mediánovy test, Kolmogorov-Smirnov test.*

Wilcoxonov test nie je úplne najvhodnejší pre použitie v korpuse, pretože reaguje na všetky rozdiely v porovnávaných dátach. Pre korpus je prijateľnejší test, ktorý vynecháva extrémne veľké alebo extrémne malé hodnoty. Tieto extrémne hodnoty sa môžu niekedy vyskytnúť pri vykonávaní testu na serveri pri aktuálnom preťažení kapacít servera, alebo iných neočakávaných udalostiach. Preto je vhodné použiť *mediánový test*, založený na hypotéze *Van der Waerdenov testu. Mediánový test* je vhodný v prípade cenzurovaných výberov, kedy sa o niektorých extrémne veľkých alebo extrémne malých hodnotách vie len to, že sú väčšie alebo menšie ako zvolená hranica.

Z definície *Van der Waerdenova testu:*

Nech $X_1,...,X_m$ je náhodný výber zo spojitého rozdelenia s hustotou $f(x)$ a nech $Y_1,...,Y_n$ je na ňom nezávislý náhodný výber s hustotou $f(x - \Delta)$. Je treba testovať hypotézu $H0: \Delta = 0$ proti alternatíve $H1: \Delta \neq 0$. Všetkých $m + n$ veličín $X_1,...,X_m$, $Y_1,...,Y_n$ (združený výber) usporiadame podľa veľkostí od najmenšej po najväčšiu hodnotu. V ňom majú veličiny $X_1,...,X_m$ poradie $R_1,...,R_m$ a veličiny $Y_1,...,Y_n$ majú poradie $R_{m+1},...,R_{m+n}$. Aplikáciou veličín na potreby korpusu dostávame nasledovné:

- *$X_1,...,X_m$ predstavujú výsledne hodnoty sledovanej veličiny prvého porovnávaného testu, pre jednotlivé súbory, nad ktorými bol test realizovaný*
- *$Y_1,...,Y_n$ predstavujú výsledne hodnoty sledovanej veličiny druhého porovnávaného testu, pre jednotlivé súbory, nad ktorými bol test realizovaný*

Pre *mediánový test* použitím teórie lokálne najsilnejších testov vyplýva štatistika

$$S' = \sum_{i=1}^{m} \left\{ sign\left[R_i - \frac{1}{2}(m + n + 1) \right] \right\}$$

Z dôvodov symetrie rozdelenia sa používa testovaná štatistika

$$S = \sum_{i=1}^{m} \frac{1}{2} \left\{ sign\left[R_i - \frac{1}{2}(m + n + 1) \right] + 1 \right\} = \frac{1}{2}(S' + m)$$

Štatistika S je rovná počtu tých veličín prvého výberu, ktoré sú väčšie ako medián združeného výberu, ak je $m + n$ nepárne číslo a medián patrí do prvého výberu, je tento počet zväčšený o ½.

Pri platnosti H_0 je rozdelenie štatistiky S symetrické okolo jej strednej hodnoty $ES = m/2$ a rozptyl je daný vzorcom

$$\operatorname{var} S = \frac{mn}{4(m + n - 1)}, \quad \text{pre } m + n \text{ parne}$$

$$\operatorname{var} S = \frac{mn}{4(m + n)}, \quad \text{pre } m + n \text{ neparne}$$

Pri $m, n \to \infty$ má pri platnosti H_0 veličina

$$U_m = \frac{S - \mathrm{E}S}{\sqrt{\operatorname{var} S}}$$

asymptotické rozdelenie N(0, 1). Ak $|U_m| \geq \mu(\alpha/2)$, zamietne sa H_0 na hladine blížiacej sa α. Pre porovnávanie testov kompresných programov je zvolená hladina 5%, z tabuliek kritických hodnôt, porovnávaná kritická hodnota pre $\alpha = 0{,}05$ je $1{,}96$.

Kapitola 5

Príklad testu kompresného programu

Nasledujúci príklad zobrazuje kroky procesu testovania autorovho kompresného programu nad danou časťou korpusu.

XML definícia korpusu:

```
<CATEGORY name="CZ">
     <CATEGORY name="text">
          <CATEGORY name="KB">
               <DATA name="000001.TXT" source_id="0" />
               <DATA name="000002.TXT" source_id="1" />
               <DATA name="000003.TXT" source_id="2" />
               <DATA name="000004.TXT" source_id="3" />
               <DATA name="000005.TXT" source_id="4" />
               .
               .
               .
               <DATA name="000068.TXT" source_id="67" />
               <DATA name="000069.TXT" source_id="68" />
          <CATEGORY>
          <CATEGORY name="MB"/>
          <CATEGORY name="GB"/>
     </CATEGORY>
</CATEGORY>
```

Test bude realizovaný nad celou subkategóriou korpusu s názvom *CZ*, subkategória reprezentuje súbory s obsahom textu v českom jazyku. Jej subkategóriou je *text*, ktorá reprezentuje súbory v jednoduchom *plain text* formáte. Poslednou hladinou subkategórií v štruktúre sú subkategórie rozdeľujúce súbory do skupín podľa ich veľkostí. V tomto prípade jediná skupina *KB* obsahuje dáta.

Pre názorné zobrazenie možností aplikácie, budú použité kompresné programy *gzip* a *bzip2*. Test bude realizovaný najskôr na programe *gzip*.

1. krokom príkladu, je zadefinovanie inicializačných parametrov testu, medzi ktoré patrí definícia užívateľových parametrov (v zobrazenom prípade testu sa nevyužijú pre generovanie reportu), definícia použitej časti korpusu (subkategória *CZ*), definícia argumentov kompresného resp. dekompresného programu (*gzip.exe COMPRESSING_FILE_NAME* resp. *gzip.exe –d ARCHIVE_FILE_NAME*).

2. krokom je vlastná realizácia testu. Po úspešnom vykonaní testu je možnosť porovnania testu s výsledkami iných testov aplikovaním štatistickej metódy neparametrického dvojvýberového *medián* testu.

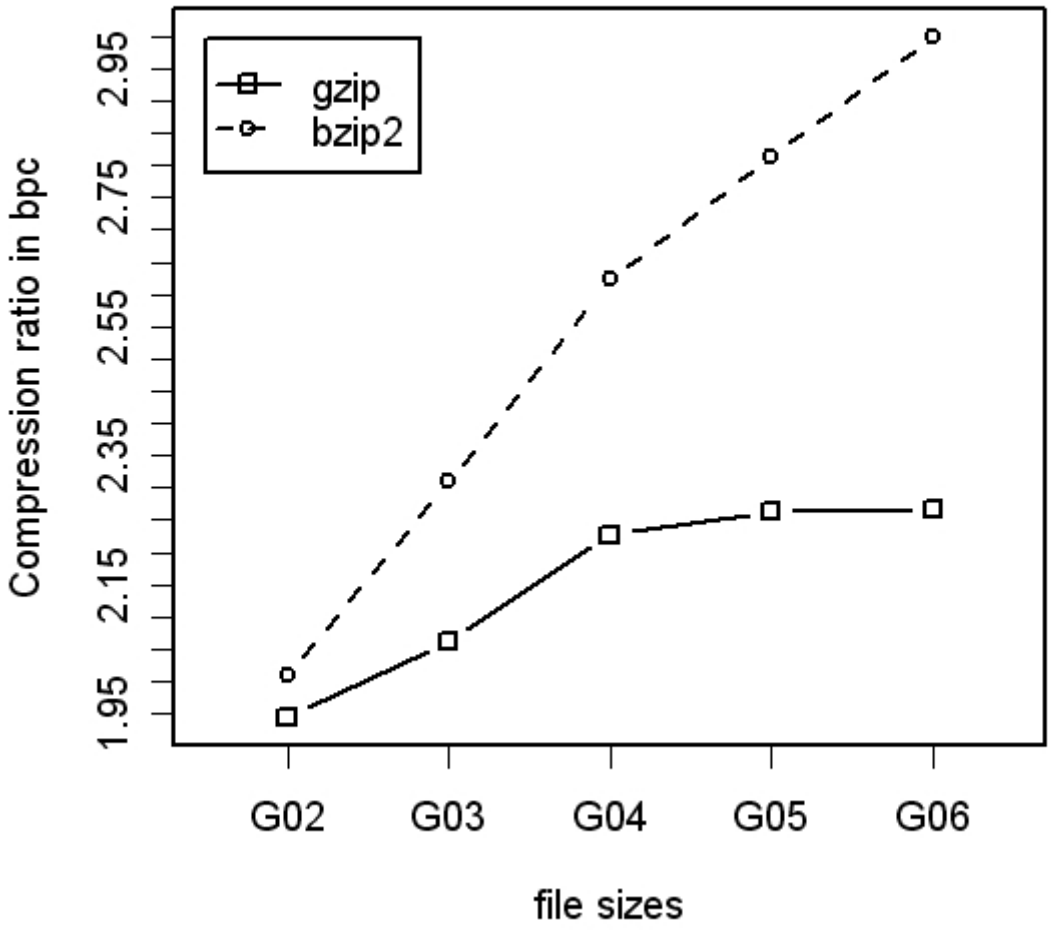
3. krokom je aplikácia krokov 1. a 2. na druhý kompresný program *bzip2*. Po úspešnom vykonaní testov dostávame nasledujúce priemerné hodnoty výsledkov testov:

program	**kompresný pomer**	**rýchlosť kompresie**	**rýchlosť dekompresie**
gzip	2,212	3533 kB	4021 kB
bzip2	2,664	1861 kB	1145 kB

4. krokom je vygenerovanie *TeX* reportu zo získaných hodnôt testu. Autor má možnosť si stiahnuť zdrojový kód výslednej tabuľky v *TeX* formáte, ktorý môže použiť do textu svojich publikácií. V zobrazenom príklade by boli použité nasledujúce hodnoty kompresných pomerov do výslednej tabuľky reportu:

	1kB - 10kB	**10kB - 50kB**	**50kB - 200kB**	**200kb - 500kB**	**500kB - 2MB**
gzip	1,944	2,062	2,227	2,263	2,267
bzip2	2,010	2,309	2,623	2,813	3,000

5. krokom príkladu je vygenerovanie grafu v *R-Graph* formáte. Pričom opäť má autor možnosť stiahnuť zdrojový kód výsledného grafu. Pre získané hodnoty v príklade bude mať výsledný graf tvar, kde skupina G02 reprezentuje veľkosti súborov 1kB-10kB, skupina G03 súbory 10kB-50kB, G04 súbory 50kB-200kB, G05 súbory 200kB-500kB a skupina G06 súbory s veľkosťami 500kB-2MB:


gzip
bzip2
Compression ratio in bpc
1.95
2.15
2.35
2.55
2.75
2.95
G02
G03
G04
G05
G06
file sizes

Kapitola 6

Možnosti rozšírenia práce

Prvotná implementácia *Corpus Systému* je navrhnutá modulárne, s možnosťou rozšírenia funkcií, implementovaných najmä vo webovej aplikácii systému, bez významných zásahov do štruktúry systému. Jednotlivé významné štruktúry použité vo webovej aplikácii, sú reprezentované interfacom, ktorý by prípadná rozšírená časť danej štruktúry implementovala. Možným prínosom pre webovú časť systému by bolo pridanie alebo dopracovanie nasledujúcich častí:

- *dopracovanie stávajúceho GUI pre administrátorskú a užívateľskú webovú časť systému*

- *vytvorenie klientskej aplikácie pre administrátorské funkcie systému a jej GUI pomocou Java technológie SWING, ktorá by uľahčila najmä administrátorskú správu korpusu*

- *implementácia ďalších výstupných formátov pre generované reporty z výsledkov kompresných testov*

Rozšírenie práce z pohľadu štatistického vyhodnocovania testov kompresných programov, je v implementácii ďalších štatistických metód dvojvýberových neparametrických testov napr. *Van der Waerdenov test, Kolmogorovov-Smirnovov test* pre porovnanie výsledkov testov v rámci viacerých štatistických metód.

Kapitola 7

Záver

Hlavným cieľom tejto práce bolo vytvoriť systém schopný testovať kompresné programy (algoritmy) na ich efektívnosť, a schopný porovnávať programy medzi sebou a tým pádom určiť ktorý kompresný program je najúčinnejší. K dosiahnutiu tohto cieľa bola potrebná implementácia nového návrhu korpusu riešiaceho problémy, ktoré sa objavili pri používaní *Calgary a Cantenbury* korpusov. Aby sa dosiahla vysoká dôveryhodnosť porovnávania kompresných programov, bola pre ich porovnanie implementovaná štatistická metóda, ktorá jednoznačne určuje, či pre dva porovnávané testy boli splnená hypotéza na danej hladine.

Ako každý projekt, aj *Corpus systém* musí byť testovaný v praxi na v reálnom zaťažení a využívajúci produkčné dáta, aby boli odstránené existujúce chyby v čo najväčšej miere. Táto práca sa sústredila na počiatočné riešenie testovania kompresných programov a generovania výsledkov týchto testov. Implementácia riešení pomocou webovej aplikácie neponúka autorom kompresných programov len možnosť testovania účinnosti vlastných algoritmov, ale aj nástroje pre tvorbu vlastných špecifických reportov z požadovaných dát. Ďalšou výhodou je možnosť generovania reportov do formátu TeX a R-Graph, ktoré autor môže použiť v textoch vlastných publikácií.

Stávajúci návrh riešenia systému počíta s možnými úpravami v budúcnosti podľa požiadaviek užívateľov. Najmä návrh komponent užívateľských prostredí musí prejsť obdobím reálneho používania, aby mohol byť prípadne systém doplnený o ďalšie užitočné funkcie, ktoré nie sú implementované v aktuálnej verzii z dôvodov, časovej náročnosti ich implementácie. Napríklad funkcionalitou systému, ktorá je vhodná pre ďalšie rozšírenie, je generovanie reportov do rôznych výstupných formátov.

Systém je otvorený ďalšiemu vývoju v danom smere. Po celú dobu vývoja systému bol braný ohľad na voľbu otvorených technológií, ktoré nie sú závislé na zvolenej platforme alebo konkrétnom produkte.

Príloha

Obsah CD

K tejto práci je priložené CD, ktoré obsahuje zdrojové kódy *Corpus systému,* vyvíjane v jazyku *Java.* Zdrojové kódy sa nachádzajú v adresári *src.*

V adresári *install* sa nachádza inštalačný súbor webovej aplikácie *Corpus systému*. Pomocou inštalačného súboru je možné jednoducho nainštalovať celú aplikáciu aj s pomocnými programami, potrebnými pre správny beh systému.

Adresár *doc* obsahuje podrobné užívateľské príručky k inštalácii webovej aplikácie, manuál k používaniu administrátorskej časti webovej aplikácie a manuál k používaniu užívateľskej webovej aplikácii.

Zoznam literatúry

[1] Calgery corpus
http://links.uwaterloo.ca/calgary.corpus.html

[2] Canterbury corpus
http://www.cosc.canterbury.ac.nz/corpus

[3] Java Reference documentation
http://java.sun.com/reference/docs/

[4] JavaServer Pages Technology
https://java.sun.com/products/jsp/

[5] Apache Tomcat 5.5
http://tomcat.apache.org/download-55.cgi

[6] Java SE Technologies
http://java.sun.com/javase/technologies/database/

[7] Apache log4j Technology
http://logging.apache.org/log4j/1.2/index.html

[8] TeX documentation
http://tex.loria.fr/english/general.html

[9] R-Graph documentation
http://www.r-project.org/
http://www.ats.ucla.edu/stat/Splus/library/lecture_graphing_r.htm
http://www.personality-project.org/r/
http://www.rpad.org/Rpad/R-refcard.pdf

[10] Java Security Manager class
http://java.sun.com/j2se/1.4.2/docs/api/java/lang/SecurityManager.html

[11] Jiří AndělZáklady matematické statistiky. *Univerzita Karlova v Praze Matematicko-fyzikální fakulta, Praha 2002*

[12] UML Specifications
http://www.jeckle.de/uml_spec.htm

[13] CVS Documentation
http://ximbiot.com/cvs/cvshome/docs/

[14] Jan Lánský, Michal Žemlička: Compression of Small Text Files Using Syllables. *Technical report no. 2006/1. Department of Software Engeneering, Faculty of Mathematics and Physics, Charles University, Prague January 2006*
http://kocour.ms.mff.cuni.cz/~zemlicka/pdf/tr2006_1.pdf

[15] Apache Ant
http://ant.apache.org/

[16] Nullsoft scriptable install system
http://nsis.sourceforge.net/Main_Page