UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Ernest Pinter

Priprava razdeljevalnih postopkov s pomočjo sistema za upravljanje različic

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Mentor: prof. dr. Blaž Zupan

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00167/2011
Datum: 06.10.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ERNEST PINTER**

Naslov: **PRIPRAVA RAZDELJEVALNIH POSTOPKOV S POMOČJO SISTEMA ZA UPRAVLJANJE RAZLIČIC**

CONCURRENT VERSION SYSTEMS FOR SUPPORT OF SOFTWARE DISTRIBUTION

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V nalogi preučite lastnosti sodobnih programskih sistemom za upravljanje različic. Preučite njihovo potencialno vlogo v razdeljevalnih postopkih. Zasnujte arhitekturo sistema, ki uporablja programsko opremo za upravljanje različic v namene izbora programskih datotek za prenos na ciljna okolja strank. Sistem implementirajte in ga preizkusite na primerih uporabe.

Mentor:

prof. dr. Blaž Zupan

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Ernest Pinter, z vpisno številko 63990302, sem avtor/-ica diplomskega dela z naslovom:

Priprava razdeljevalnih postopkov s pomočjo sistema za upravljanje različic

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Blaža Zupana
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov.,angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 1.decembra 2011 Podpis avtorja/-ice:

Zahvala

Zahvaljujem se mentorju prof. dr. Blažu Zupanu za usmerjanje in pomoč pri izdelavi diplomskega dela, nadrejenemu v podjetju HRC informacijski inženiring d.o.o. Iztoku Koštomaju za mentorstvo v področju sistemov za nadzor različic in idejah za izvedbo razdeljevalnih postopkov. Največja zahvala pa gre družini, ženi Moniki za potrpežljivost in podporo v času pisanja diplomskega dela ter hčerama Hani in Zoji ker sta bili večkrat prikrajšani za mojo družbo.

Kazalo

Seznam uporabljenih kratic in simbolov

CVS Concurrent Versions System

RCS Revision Control System

ACL Access Control List

SQL Structured Query Language

PL/SQL Oraclova proceduralna razširitev SQL jezika

NSIS Nullsoft Scriptable Install System (http://nsis.sourceforge.net)

Slovenske različice angleških pojmov iz CVS

skladišče repository

peskovnik sandbox

objava commit

prevzem checkout

uvoz import

osvežitev update

navzkrižje conflict

oznaka tag

veja branch

glavna veja head branch, trunk

Povzetek

V pričujoči diplomski nalogi obravnavamo sisteme za upravljanje z različicami in pristope k razvoju razdeljevalnih postopkov. Osredotočimo se na popularna sistema CVS in Subversion. Naša predpostavka je, da so poleg vodenja različic tovrstni sistemi uporabni tudi za druga opravila. Eno od teh je ugotavljanje primernosti posamezne različice progamske datoteke za prenos na ciljna okolja strank. V nalogi pregledamo tipične zahteve teh, tako imenovanih razdeljevalnih postopkov. Na osnovi naše analize primer razdeljevalnega postopka tudi implementiramo in ga preskusimo v praksi.

Ključne besede:

CVS, Subversion, RCS, Sistemi za upravljanje z različicami, Oracle, SQL.

Abstract

In this thesis we study concurrent version systems and approaches to support software distribution. We focus on two popular systems, CVS and Subversion. It is assumed that aside from version management, these systems can also be used for other tasks. One of these is to determine which version of individual program files are appropriate for the installation on target customer environments. The thesis examines typical requirements of the so-called distribution procedures. Based on the analysis, the example of distribution procedure is also implemented and tested in practice.

Key words:

CVS, Subversion, RCS, Concurrent version systems, Oracle, SQL

Poglavje 1

Uvod

V sodobnem času razvoja programske opreme si le tega brez sistema za upravljanje različic ne znamo več predstavljati. Lahko tudi trdimo, da se večjega projekta brez tega ne da kvalitetno izvesti. Obstaja cela vrsta orodij, ki nam tovrstno upravljanje omogočajo in so si med sabo deloma podobni, deloma pa se razlikujejo v arhitekturi ali v načinu uporabe. V drugem poglavju pričujoče diplomske naloge sta predstavljena dva priljubljena odprtokodna predstavnika takšnih sistemov CVS in Subversion s poudarkom na razlikah med njima.

V podjetju HRC informacijski inženiring d.o.o. razvijamo in vzdržujemo celovit informacijski sistem za podporo poslovanju bank imenovan Hibis. Za upravljanje z različicami programske kode uporabljamo sistem CVS. V procesu razvoja in dostave programske opreme smo naleteli na težavo, za rešitev katere smo izkoristili sistem za upravljanje različic. Zaradi velikega obsega informacijskega sistema in velike količine programskih datotek iz katerih je ta sestavljen, je bilo potrebno poiskati čimbolj učikovito pot dostave izdaj programske opreme do strank. S pomočjo sistema za upravljanje različic znamo določiti, katere različice posameznih datotek in modulov se nahajajo pri stranki in na podlagi tega lahko poiščemo datoteke, ki so se od zadnjega prenosa spremenile. Tem postopkom priprave datotek za prenos pravimo razdeljevalni postopki oziroma distribucija. V tretjem poglavju je sistem razdeljevanja podrobneje opisan na primeru našega informacijskega sistema.

Cilj diplomske naloge je bila prenova programske opreme za razdeljevalne postopke. Ta bo nadomestila ukazne skripte, ki so bili v uporabi do sedaj. Sestavljena je iz strežniškega in odjemalskega dela ter napisana v programskem jeziku Java, ki se je v tem primeru izkazal kot najprimernejši zaradi uporabe na različnih platformah (Windows in Linux). Delovanje novega sistema je opisano v četrtem poglavju poglavju, zanimivejši izseki programske kode strežniškega

dela pa v prilogi. Drugi cilj pa je poiskati odgovor na vprašanje ali lahko za opisane razdeljevalne postopke v enaki meri izkoristimo CVS in Subversion.

Poglavje 2

Sistemi za upravljanje različic programske opreme

Sistem za upravljanje različic je proces beleženja in nadzorovanja sprememb v projektu. Iz vidika razvoja programske opreme (kar bomo obravnavali v nadaljevanju) to pomeni nadzor nad izvorno kodo projektov. Prednosti uporabe sistemov za upravljanje različic lahko ponazorimo s seznamom opravil, ki jih tipičen sistem podpira:

- Vpogled v starejše različice projekta ali posamezne datoteke.
- Prikaz sprememb med dvema fazama projekta ali dvema različicama datoteke.
- Istočasno lahko na projektu dela več razvijalcev brez tveganja z izgubo narejenega dela, podprto je tudi sočasno delo oz. popravljanje iste datoteke.
- Omogoča označevanje in vejanje kar je mogoče uporabiti za vzporedni razvoj, bodisi za popravljanje starejših različic ali pa razvoj novih funkcionalnosti. Spremembe je mogoče združevati iz ene veje na drugo brez ponovnega programiranja.
- Vsako različico je mogoče opremiti s komentarjem, imenom razvijalca, časom objave, kar odpre možnosti različnih analiz razvoja s strani projektnega vodje.

Jedro sistema ponavadi predstavlja strežniški del, na katerem se nahaja skladišče. Tu se na tak ali drugačen način hrani vsa zgodovina sprememb na

določenem projektu, eno skladišče lahko hrani več projektov ali pa je vsak projekt shranjen v svojem skladišču. Sliko skladišča si s pomočjo odjemalca razvijalec prenese v lokalni peskovnik, kjer lahko pregleduje izvorno kodo ali izvaja spremembe, ki se po objavi prenesejo nazaj v skladišče.

Tovrstni sistemi niso koristni samo za upravljanje z izvorno kodo, lahko se uporabijo za shranjevanje dokumentov, sistemsko administracijo (nastavitvene datoteke sistema) oz. karkoli, kar zadeva spreminjanje datotek, kjer je možna potreba po vpogledu v starejše različice.

2.1 Primeri sistemov za upravljanje različic

CVS

Sistem CVS se je razvil iz predhodnika RCS. Prva različica je izšla leta 1990. Glede na predhodnika, ki je omogočal vodenje različic za posamezne datoteke, CVS omogoča tudi upravljanje s projekti. Med drugim uporablja centralno skladišče, ki je lahko oddaljeno in omogoča konsistentno hkratno delo več uporabnikov, ki je sledljivo. CVS je kot prvi sistem za upravljanje različic vpeljal veje (več o tem v poglavju 2.3), funkcionalnost so kasneje povzeli tudi ostali sistemi. Za CVS veljajo vse v prejšnjem poglavju omenjene značilnosti sistemov za upravljanje različic, poleg tega pa omogoča tudi nadzor uporabe s pomočjo prožilcev - skript ki preverjajo, dopolnjujejo in/ali dodatno beležijo dogodke v sistemu. Na primer, CVS omogoča nadzor ustreznosti komentarja ob potrditvi, nadzor ustreznosti imena oznake ali veje in podobno. Omogoča tudi zaklepanje datotek na strežniku za izključno urejanje, kar se izkaže kot uporabno pri dvojiških datotekah, kjer sprememb ni mogoče združevati.

Subversion

Projekt Subversion je leta 2000 vzpostavilo podjetje Collabnet z namenom, da razvijejo sistem podoben CVS-ju, ki pa bi odpravil nekatere pomanjkljivosti in napake, ki jih ta sistem ima. Sistem je bil razvit od začetka, torej ne na osnovi CVS-a, pač pa je prevzel od njega osnovne ideje pri čemer so se želeli izogniti najočitnejšim težavam, ki jih je po njihovem mnenju CVS imel. Subversion se od CVS-ja bistveno razlikuje v naslednjih točkah:

- Atomarno objavljanje: ob potrjevanju sprememb na strežnik se naredi potrditev na vseh navedenih datotekah ali na nobeni v primeru napake

pri potrjevanju posamezne datoteke. To pomeni, da skladišče ne more priti v nekonsistentno ali neveljavno stanje.

- Vodenje različic: različice se vodijo na nivoju celotnega skladišča in ne za vsako datoteko posebej. Številka različica se poveča ob potrditvi sprememb v skladišče ne gleda na število datotek zajetih v potrditev.

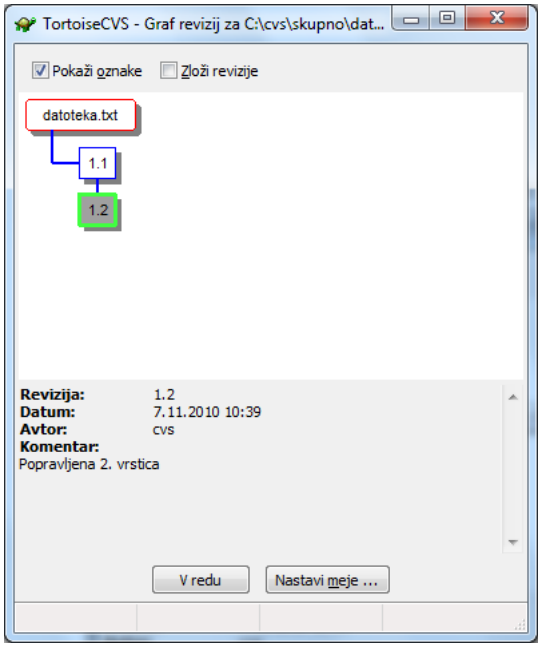
- Postavljanje oznak ali vej je enostavna in časovno poceni operacija, sistem ne ločuje med oznakami, vejami in imeniki (več o tem v poglavju 2.3)

Eden izmed ciljev projekta je bila tudi možnost prehoda iz CVS-ja na Subversion na obstoječih projektih skupaj z vso zgodovino sprememb.

2.2 Uporaba sistema za upravljanje različic

Če se postavimo na stran uporabnika, je začetek uporabe sistema pogojen statusom projekta, ki se vodi ali se bo vodil v sistemu za upravljanje različic. Če se projekt že nahaja v sistemu, uporabnik prične z uporabo s prevzemom projekta iz skladišča v lokalni peskovnik. Če projekta v skladišču še ni, ga mora uporabnik najprej uvoziti, drugi uporabniki pa do tega projekta potem zopet dostopajo s prevzemom.

V fazi razvoja uporabnik popravlja različne datoteke, ko je razvoj zaključen uporabnik le te objavi v skladišče. Drugi uporabniki projekta si morajo spremembe osvežiti v lokalni peskovnik. V primeru, da je uporabnik datoteko, ki jo tudi sam popravlja, osvežil z novo različico iz skladišča se zgodi združevanje. Če se obe spremembi ne prekrivata (ni popravkov v isti vrstici), potem se združevanje napravi samodejno. V primeru popravka v isti vrstici pa pride do navzkrižja. V tem primeru mora uporabnik, ki je naredil osvežitev določiti, katera sprememba vrstice je pravilna, sicer njegova potrditev v skladišče ni možna. Tu je smiselen dogovor z osebo, s katero je prišel v navzkrižje. Na ta način pri znakovnih datotekah ne prihaja do izgube dela v primerih, ko na isti datoteki dela več uporabnikov. Pri dvojiških datotekah pa je položaj drugačen. Sistem sam ne zna združevati datotek zato se v teh primerih priporoča izključen način dela. Uporabnik pred začetkom dela z datoteko prijavi v sistem, da bo datoteko začel urejati. V skladišču na strežniku se to zabeleži in do potrditve spremembe ne more te datoteke spreminjati noben drug uporabnik. To je rešeno na ta način, da skladišče ne dovoli prijave urejanja datoteke, brez tega pa je v lokalnem peskovniku datoteka označena samo za branje.

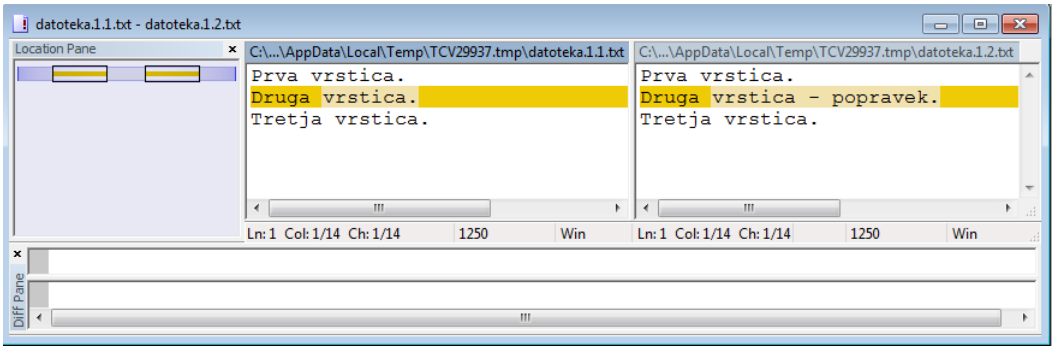


Slika 2.1: Primer grafa v odjemalcu TortoiseCVS.

Poleg izvajanja sprememb ima uporabnik tudi več možnosti pregledov in analiziranja objavljenih različic. Osnovni možnosti sta ukaza za dnevnik (log) in spremembe (diff), ki pravo vrednost pokažeta v primeru grafičnega odjemalca. Ti znajo v večini primerov dnevnik prikazati v obliki grafa, spremembe pa na uporabniku prijazen način. Na sliki 2.1 je primer grafa iz orodja TortoiseCVS (http://www.tortoisecvs.org), ki teče na odjemalcu, na sliki 2.2 pa prikaz sprememb v orodju WinMerge (http://winmerge.org). Ti dve orodji je mogoče nastaviti, da delujeta drug z drugim. Na primer ukaz za prikaz sprememb med dvema različicama v TortoiseCVS izvozi ustrezne različice datotek iz CVS skladišča in jih posreduje orodju WinMerge, ki analizira razlike in jih prikaže. Za Subversion lahko namesto TortoiseCVS uporabimo podobno orodje TortoiseSVN. Vsa našteta orodja so odprtokodna in brezplačno dostopna za okolje Windows. Podobna orodja obstajajo tudi za druge opercijske sisteme.

2.3 Oznake, veje in združevanje različic

Naprednejša uporaba sistema za upravljanje različic zajema tudi uporabo oznak in vej. Čeprav lahko iz sistema v vsakem trenutku izvozimo katerokoli

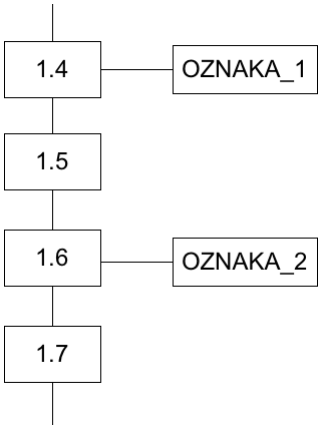


Slika 2.2: Primer prikaza razlik med dvema različicama v programu WinMerge.

različico iz preteklosti, je težko določiti katere različice posameznih datotek tvorijo konsistentno oz. smiselno celoto. Če v trenutku, ko to poznamo, postavimo na ustrezne različice oznako, jih lahko kasneje izvozimo skupaj kot celoto, ne glede na to, koliko različic je na posameznih datotekah bilo dodanih kasneje. Druga možnost uporabe, ki jo uporabljamo tudi mi in na katero se bomo navezovali kasneje, je označevanje različice datoteke, ki je nameščena na določenem ciljnem sistemu. To je mogoče izkoristiti na dva načina, prvi je ta, da imamo informacijo o tem, kaj je nameščeno na določenem sistemu stranke kar v razvojnem okolju, druga prednost pa je možnost, da za prihodnjo namestitev prenesemo samo razlike med trenutno namestitvijo in novimi različicami datotek. Za ta namen se uporabljajo plavajoče oznake. Oznako je po pripravi nove namestitve potrebno prestaviti na novo različico. Sistem je podrobneje razložen v 3. poglavju. Postavljanje oznak je na splošno smiselno in priporočeno [3] ob naslednjih dogodkih:

- ko je na projektu zaključena kakšna kompleksna funkcionalnost in na večjih mejnikih projekta,
- pred izločitvijo kakšne obstoječe funkcionalnosti,
- pred začetki testiranj,
- pred implementacijo sprememb, ki bi lahko pokvarila delujočo kodo,
- po združevanju vej.

Pravila postavljanja oznak je potrebno določiti glede na potrebe projekta, razvojnih orodij in platform ter glede na politiko podjetja ali razvojne ekipe, ki

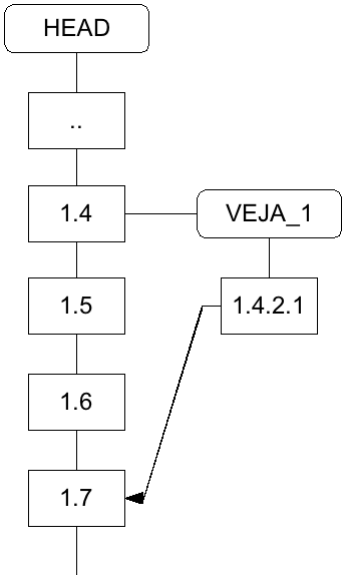


Slika 2.3: Primer oznak pripetih na različice.

projekt razvija. V vsakem primeru pa je priporočena smiselna in sistematična uporaba imen oznak, sicer lahko ob velikem številu le teh prihaja do nesporazumov. Na sliki 2.3 je prikazan graf različic s pripetima dvema oznakama.

Veje lahko obravnavamo kot oznake, na katerih pa je možen tudi vzporeden razvoj. Osnovna črta razvoja se imenuje glavna veja. Tako kot pri oznakah je tudi pri vejah več različnih načinov in več razlogov za uporabo. Vsem pa je skupno to, da je na veji mogoč tudi dodaten razvoj, ki teče vzporedno z glavno vejo ali drugimi vejami, brez posegov v vsebino na le teh. Tipičen primer uporabe veje je izdaja nove različice programske opreme. Na skupek različic datotek, ki predstavlja novo izdajo se postavi veja, kar omogoči dodatna testiranja pred izdajo ali odpravo hroščev po izdaji, obenem pa na glavni veji teče nemoten razvoj novih funkcionalnosti v katere je mogoče združiti popravke hroščev, v kolikor je to potrebno. Vejo je mogoče uporabiti tudi v primeru poizkusnega razvoja. Če se sprememba obnese, se združi v nazaj v glavno vejo, sicer se veja opusti.

Kadar se na projektu uporabljajo veje, nastane potreba po združevanju sprememb iz ene veje v drugo. Primer na sliki 2.4 prikazuje vejo na kateri je bila izvedena sprememba, ki je bila vključena (združena) v glavno vejo, kar je ponavadi ponazorjeno s puščico, ki kaže smer združevanje. Vzemimo, da veja ponazarja izdajo. Na različico 1.4 je bila postavljena veja, na glavni veji so se naprej razvijale nove funkcionalnosti (do različice 1.6). Na izdaji je bila ugotovljena napaka v programu, ki je bila v različici datoteke 1.4.2.1 odpravljena. Združevanje vsebine z veje VEJA_1 popravek vključi v glavno vejo. Pri združevanju z vej veljajo enaka pravila kot so v poglavju 2.2 opisana



Slika 2.4: Primer veje na različici.

za združevanje ob osveževanju (samodejni popravki, navzkrižja). Združevanje je mogoče tudi v nasprotno smer. Vsebinsko bi glede na omenjen primer to pomenilo, da se nova funkcionalnost prenese na vejo, na primer vključi v prejšnjo izdajo).

Za izvajanje popravkov na vejah je v CVS sistemu potreben ponoven prevzem datotek iz skladišča v krajevni peskovnik. Običajno se ob prevzemu prenesene glavna veja, obstaja pa možnost izbire poljubne veje. To se ponavadi lokalno prenese v nov imenik, ki se imenuje enako kot veja. Ko je potrebno narediti popravek na veji, se je potrebno postaviti v ustrezni imenik in urediti datoteko ter jo potrditi v skladišče. Rezultat takšnega sistema je, da je v peskovniku ista datoteka v več različnih imenikih glede na to, koliko vej te datoteke je prevzetih iz skladišča. Če se razvoj na kakšni veji zaključi jo je mogoče sprostiti iz peskovnika, vseeno pa veja z vsemi popravki ostane zabeležena v skladišču.

Subversion pa oznake in veje obravnava na drugačen način. Za sistem ni razlike med oznako, vejo in imenikom. Postavljanje oznake pomeni kopiranje imenika v drug podimenik. Če se v tem novem podimeniku razvoj nadaljuje to pomeni, da se je oznaka spremenila v vejo. Kljub temu ostaja osnovna filozofija enaka. Možna je uporaba enakih tehnik vejanja in združevanja v eno ali drugo smer. Tudi v peskovniku je struktura podobna. V literaturi [2] pri

uporabi Subversiona priporočajo naslednjo imeniško strukturo:

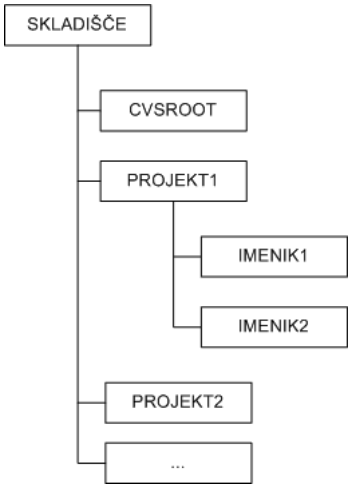
- projekt
  - imenik za glavno vejo razvoja (angl. *trunk*)
  - imenik za kopiranje v oznake (angl. *tags*)
  - imenik za kopiranje v veje (angl. *branches*)

Kopiranje imenikov v oznake ali veje za skladišče ne pomeni podvajanja vsebine, ampak ustvari povezave oz. kazalce na željeno različico. Sistem, ki ga uporablja Subversion ima tudi slabost, da ne podpira možnosti prej omenjenih plavajočih oznak, kar omeji uporabnost označevanja. Prav tako ni mogoče postaviti oznako na posamezno datoteko ampak je najmanjša enota imenik.

Poznamo dve osnovni filozofij postavljanja vej oz. vejanja [3]:

- **V splošnem stabilna koda (ang. *basicaly stable*)**: Glavna veja je tu vzdrževana na ta način, da je v večini primerov pripravljena na izdajo različice. Veje se uporabljajo za razvoj, odpravo napak in poizkusno programsko kodo. V najstrožji obliki prepoveduje združevanje programske kode na glavno vejo preden gre skozi postopke za zagotavljanje kakovosti, v pogostejših praksah pa je dovoljeno združevanje programske kode, ki prestane razvojne testne primere. Prednost tega sistema je, da je mogoče praktično kadarkoli objaviti novo izdajo programske opreme z glavne veje, saj se nove funkcionalnosti in za hrošče dovzetna poizkusna programska koda razvijajo na vejah. Slabost pa je ponavadi ta, da združevanje na glavno vejo dela druga oseba kot je programsko kodo razvijala (npr. odgovorni za zagotavljanje kakovosti) in ki le to najbolje pozna, kar lahko vodi do nepravilnega združevanja in posledično napak v izdanem programu.

- **V splošnem nestabilna koda (ang. *basicaly unstable*)**: Pri tej filozofiji pa glavna veja vsebuje zadnje popravke oz. razvite funkcionalnosti. Veje se uporabljajo objavo izdaj in popravke hroščev, v manj strogi obliki pa se tudi v tem primeru na vejah razvija poizkusna koda. Prednost tega sistema je majhno število potreb združevanj, slabost pa je očitna že v imenu, namreč to da lahko glavna veja v nekaterih primerih vsebuje hroščato, nestabilno ali celo nedelujočo kodo, zato je potreben dodaten nadzor nad programsko kodo in ustrezno časovno načrtovanje vej za objavo novih izdaj programske opreme. Na naših projektih večinoma uporabljamo to vrsto postavljanja vej.



Slika 2.5: Imeniška struktura skladišča.

2.4 Organizacija skladišča

CVS

Skladišče sistema je zgrajeno iz projektnih imenikov in posebnega administrativnega imenika CVSROOT. Projektni imeniki (imenovani tudi moduli) so vrhnji imeniki, ki ponavadi predstavljajo med seboj povezane ali nepovezane projekte. Znotraj teh imenikov se nahajajo projektne datoteke in podimeniki. Projektne datoteke so shranjene v formatu RCS s končnico ",v" (npr. dokument.txt,v je poimenovana rcs datoteka za datoteko, ki se v lokalnem peskovniku imenuje dokument.txt). V datoteki RCS je shranjena celotna zgodovina sprememb skupaj s podatki o oznakah in vejah na učinkovit način, kar pomeni, da se v znakovnih datotekah beležijo samo spremembe glede na prejšnjo različico in ne celotna različice datoteke. Primer imeniške strukture CVS skladišča je prikazan na sliki 2.5.

Na sliki 2.6 je prikazana vsebina datoteke RCS, kot je shranjena na strežniku skupaj z dvema spremembama izvršenima v peskovniku odjemalca in potrjenih v skladišče strežnika (primer na sliki 2.7). Razvidno je, da je v datoteki RCS zadnja različica vedno predstavljena v celoti, prejšnje različice pa kot razlike. Ob zahtevi po starejši različici tako CVS sestavi ustrezno vsebino datoteke in jo vrne odjemalcu. To velja za znakovne datoteke, pri dvojiških se za vsako različico shrani celotna vsebina datoteke.

Vsak projektni imenik in podimenik lahko vsebuje tudi imenika Attic in

```
head 1.2;
access;
symbols;
locks; strict;
comment @# @;

1.2
date 2010.11.07.09.39.03; author cvs; state Exp;
branches;
next 1.1;
commitid G4Kp2lNjiZpACtVu;

1.1
date 2010.11.07.09.38.21; author cvs; state Exp;
branches;
next ;
commitid Awb8mnldLlilCtVu;

desc
@@

1.2
log
@Popravljena 2. vrstica
@
text
@Prva vrstica.
Druga vrstica - popravek.
Tretja vrstica.@

1.1
log
@Datoteka dodana v CVS
@
text
@d2 1
a2 1
Druga vrstica.
@
```

Slika 2.6: Datoteka RCS z imenom datoteka.txt,v.

Prva različica datoteke datoteka.txt:

```
Prva vrstica.
Druga vrstica.
Tretja vrstica.
```

Druga različica, popravek 2. vrstice:

```
Druga vrstica - popravek.
```

Slika 2.7: Vsebina datoteke datoteka.txt.

CVS. V imeniku Attic se nahajajo datoteke, ki ne obstajajo v glavni razvojni veji - lahko so obstajale v preteklosti in so bile zbrisane ali pa obstajajo samo na stranskih vejah. Izbrisane datoteke ostanejo v CVS zaradi beleženja zgodovine, niso pa več vidne v peskovniku na strani odjemalca. V imeniku CVS so zabeleženi metapodatki o lastnostih posameznih datotek, npr. ali je datoteka zaklenjena za uporabo s strani kakšnega uporabnika oz. ali je datoteka opazovana s strani kakšnega uporabnika (Ukazi cvs edit, watch).

Datoteke lahko zaklepa tudi strežnik sam. To se zgodi ob objavi novih različic datotek s strani odjemalca. Zaklene se celoten imenik za čas trajanja objave. Ta tipično traja del sekunde, lahko pa tudi več minut, npr. ob objavi dvojiških datotek preko počasne mrežne povezave. V času zaklepa ne more noben uporabnik posodabljati, označevati ali objavljati datotek v tem imeniku. Zaklep velja samo za trenuten imenik, ne pa tudi za podimenike, ki morebiti v tem imeniku obstajajo. Zaklep imenika je na strežniku predstavljen kot posebna datoteka.

Del strukture skladišča je tudi imenik CVSROOT. V njem so shranjene nastavitvene datoteke in skripte, ki se prožijo ob dogodkih v skladišču (npr. objava, dodajanje oznak, vej,..). Tudi ta imenik je mogoče prevzeti v lokalni peskovnik in tam urejati datoteke, zato je potrebno ustrezno urediti pravice za dostop do imenika.

Subversion

Za razliko od CVS, kjer je skladišče organizirano v podobni strukturi kot je potem vidna v peskovniku, ima SVN drugačno organizacijo. V tabeli 2.1 je predstavljena imeniška struktura skladišča na strežniku SVN.

Različice datotek so, kot je prikazano, shranjene v imenuiku db. Ta lahko

Tabela 2.1: Organizacija skladišča Subversion

Imenik	Opis
conf	Vsebuje nastavitvene datoteke.
dav	Rezerviran za apache modul mod_dav_svn.
db	Shramba datotek za katere se vodijo različice.
format	Datoteka vsebuje trenutno številko različice skladišča.
hooks	Imenik vsebuje skripte prožilce ob dogodkih v skladišču (enakovredno CVSROOT imeniku na sistemu CVS).
locks	Shramba evidence zaklepanja datotek s strani odjemalcev.

nastopi v dveh oblikah, prva je podatkovna zbirka Berkley DB (BDB), druga pa datotečna struktura s kratico FSFS. Slednja uporablja posebno organizacijo datotek na datotečnem sistemu operacijskega sistema. Podobno kot CVS tudi ta shranjuje razlike med različicami v datoteko, le da to ne shranjuje za vsako datoteko posebej ampak za vsako objavo v skladišče (ki pomeni novo različo skladišča) posebej, ime datoteke pa je kar številka različice objave. Uporaba FSFS ima kar nekaj prednosti pred uporabo Berkleyeve podatovne zbirke. Na primer prenosi skladišča na drug strežnik ali datotečni sistem, vzdrževanje manjšega števila sistemsko odvisnih komponent, ... Zato se na novih skladiščih večinoma uporablja ta možnost. V prvih različicah je Subversion privzeto uporabljal BDB, v zadnjih pa je privzeti FSFS. Na strani odjemalca ni razlike pri uporabi sistema ne glede na uporabljen način shranjevanja različic na strežniku.

2.5 CVSNT: nadgradnja sistema CVS

Sistem CVSNT se je pred leti nastal kot veja sistema CVS, za njegov razvoj pa skrbi podjetje March Hare (http://march-hare.com). Sprva je bil prilagojen strežnikom na okolju Windows, kasneje pa so ga prenesli tudi na druge platforme. Projekt je sicer odprtokoden vendar je pri zadnjih izdajah podjetje omejilo dostop do odprtokodne različice. Mogoče je prevzeti izvorno kodo in jo prevesti, ni pa mogoče dobiti brezplačne prevedene različice. Orodje je sedaj del sistem CVS Suite in se prodaja skupaj s podporo. Ključne prednosti pred CVSNT pred CVS sistemom:

- avtentikacija preko domenskega strežnika (neposredno podpira Active

Directory),

- napredna uporaba prožilcev ob dogodkih, podprta je uporaba sistemskih knjižnic (dll, so),
- preimenovanje datotek s strani odjemalca (poizkusna funkcionalnost),
- omogoča podrobno določanje pravic za dostope do posameznih operacij (ACL), kar je mogoče določiti tudi glede na module ali imenike.

Slednja funkcionalnost je tudi glavni razlog, da smo na naših projektih prešli na to različico CVS. Čeprav se tudi možnost preimenovanja datotek na strani odjemalca sliši mikavna, ima zadeva zaenkrat še precej težav. Ukazi, ki tečejo na strežniku namreč takšnih sprememb imena ne zaznajo v celoti, kar otežuje napredne analize in pripravo razdeljevalnih postopkov, ki bodo opisani v nadaljnih poglavjih. Kljub temu, da so bili razdeljevalni postopki razviti za CVSNT sistem, uporabljajo ukaze, ki so združljivi tudi s CVS sistemom, kjer so bili tudi preizkušeni.

2.6 Ostali sistemi

V zadnjem času so predvsem v odprtokodni skupnosti postali popularni razdeljeni sistemi za upravljanje z različicami (DVCS) [1]. Ti za razliko od prej obravnavanih sistemov v osnovi ne uporabljajo centralnega skladišča (sistem odjemalec-strežnik), ampak sistem uporabnik-uporabnik. Uporabniki si izmenjujejo delovne različice, katero vsakdo avtonomno razvija, funkcionalnosti pa se potem združujejo. Ponavadi obstaja na takšnih projektih hierarhija skrbnikov (mreža zaupanja), kjer višji skrbniki odločajo ali bodo določene funkcionalnosti odobrene in združene ali zavrnjene. Ker večina operacij (potrjevanje, pregled zgodovine, ...) poteka lokalno, je sistem zelo hiter v fazi razvoja novih funkcionalnosti projekta, uporabniki pa ne potrebujejo niti omrežne povezave. Iz tega izhaja tudi slabost takšnih sistemov. Začetno kloniranje izvorne kode, ki je ustreznica prevzemu v odjemalec-strežnik sistemih, traja precej dlje, saj je potrebno lokalno prenesti vse spremembe z oznakami in vejami vred. Končni rezultat je torej toliko kopij skladišča, kolikor je aktivnih razvijalcev na projektu. Od tod tudi ime porazdeljeni sistemi. Najbolj znana tovrstna orodja so našteta v tabeli 2.2

Poleg naštetih obstaja tudi več komercialnih orodij za upravljanje z različicami. V osnovi ponujajo podobne funkcionalnosti kot CVS ali Subversion, razlikujejo se v organizacijah skladišča, nudijo večjo podporo pri uvajanju in

Tabela 2.2: Porazdeljeni sistemi za upravljanje z različicami

Orodje	Opis
Git	Avtor Linus Torvalds, uporablja se na projektu za razvoj Linux jedra in ostalih, npr. GNOME, X.org.
Mercurial	Uporablja se na projektih Mozilla, OpenJDK, Netbeans in drugih odprtokodnih projektih povezanih z Javo.
Bazaar	Sponzoriran s strani podjedja Canonical, ki bdi nad operacijskim sistemom Ubuntu, uporablja ga tudi projekt MySQL.

uporabi, večkrat pa nastopajo kot del orodja, ki pokriva širše funkcionalnosti. Svoje rešitve ima tako tudi večina večjih podjetij, ki ponujajo orodja za razvoj programske opreme, na primer Microsoft (Team Foundation Server), IBM (Rational Clear Case) in drugi.

2.7 Primerjava CVS in Subversion

Čeprav je Subversion nastal s sloganom "CVS, le je da boljši", lahko iz našega vidika rečemo "CVS, le da je drugačen". Subversion ima vgrajeno podporo preimenovanju in premikanju datotek iz enega imenika v drugega. To se odraža na enak način kot običajna sprememba vsebine datoteke in je prikazano v zgodovini različic. Omogoča tudi atomarne objave, kar pomeni da se v skladišče objavijo vse zahtevane spremembe ali pa nobena, če za kakšno datoteko iz kakršnegakoli razloga objava ni možna. CVS objavlja vsako datoteko posebej, kar pomeni, da lahko projekt postavi v nekonsistentno stanje. Na primer, če objavljamo popravek dveh datotek, ki sta med seboj odvisni in eno sistem objavi, drugo pa ne, lahko to pomeni, da drug uporabnik, ki datoteko prevzame, projekta ne more več uspešno prevesti. To je še posebej pomembno pri nekaterih programskih jezikih, ki programsko kodo prevajajo v izvršilno datoteko (npr C ali Java). Tam se ena izvršilna datoteka običajno prevede iz večjega števila programskih datotek. Tudi v primeru naše programske kode (SQL in PL/SQL skripte, ki se nameščajo v podatovno bazo) je to lahko težavno. Funkcionalnosti v različnih skriptah so tu lahko povezane, v eni datoteki se doda polje tabeli, v drugi datoteki za programsko kodo pa se to polje uporabi. Subversion učinkoviteje postavlja oznake in veje. Časovno in z vidika sistemskih sredstev je to poceni operacija, vzpostavijo se samo povezave na ustrezne različice. CVS ob dodajanju oznake ali veje oznako umesti v RCS

datoteko, v praksi datoteko najprej prepiše v kopijo in če je operacija uspešna kopijo preimenuje v originalno datoteko. To je lahko dolgotrajna operacija, še posebej pri dvojiških datotekah, ki so praviloma večje poleg tega pa je vsaka različica shranjena v RCS datoteki v celoti.

Poglavje 3

Opis sistema za razdeljevalne postopke

Postopke za izbiro in označevanje datotek, primernih za dostavo na ciljne sisteme strank, v podjetju imenujemo razdeljevalni ali distribucijski postopki. Naloga teh postopkov je, da na osnovi oznak in vej v sistemu za upravljanje različic izberejo datoteke, ki jih je potrebno prenesti in namestiti na izbrani sistem stranke za izbrano izdajo programsko opreme. V tem poglavju bodo ti postopki podrobneje razloženi na primeru našega informacijskega sistema.

3.1 Kratek opis informacijskega sistema

V podjetju razvijamo in ponujamo celovit informacijski sistem za podporo poslovanju bank. Sestavljen je iz naslednjih glavnih modulov oziroma oddelkov z opisom programske podpore:

Krediti. Poslovanje s krediti fizičnih in pravnih oseb ter depoziti pravnih oseb. Modul zajema približno 1000 zaslonskih mask, 1200 izpisov in okoli 2500 SQL in PL/SQL skript v skupni velikosti približno 500 MB.

Šaltersko poslovanje. Poslovanje z občani (računi, depoziti, kartice, ..), ki zajema blagajniško okence in zaledno poslovanje ter upravljanje z računi pravnih oseb v domači valuti. Zajema približno 1100 zaslonskih mask, 1300 izpisov, 3400 SQL in PL/SQL skript v skupni velikosti približno 600 MB.

Plačilni sistemi. Domači in tuji plačilni promet ter vodenje računov pravnih oseb v tujih valutah. Oddelek pokriva tudi bazo poslovnih partnerjev ter

poslovanje z vrednostnimi papirji, akreditivi, ... Zajema približno 2200 zaslonskih mask, 1400 izpisov, 3000 SQL in PL/SQL skript v skupni velikosti približno 730 MB.

Poročanje. Zajem in čiščenje podatkov ter izdelava poročil za različne državne institucije ter interna poročila bank, ter vodenje glavne knjige. Zajema približno 400 zaslonskih mask, 350 izpisov in 2700 SQL in PL/SQL skript v skupni velikosti približno 250 MB.

Sistem teče na podatkovni bazi Oracle (trenutno na različici 10g), uporabniški vmesnik pa je zgrajen s pomočjo Oracle Forms 6i in Oracle Reports 6i. Trenutno sistem uporablja dvonivojsko arhitekturo. V pripravi je prehod na tronivojsko arhitekturo z uporabo Oracle Forms 11g in Oracle Reports 11g ter z aplikacijskim strežnikom Oracle Weblogic 11g.

3.2 Razvojni cikel

Nove različice programske opreme so v opisanem sistemu na voljo dvakrat letno in sicer v prvem tednu meseca maja in v prvem tednu meseca novembra. Na testno okolje se različica prenese mesec dni prej. Stranke nove zahteve za naslednjo različico prijavijo do 15.6. v tekočem letu za namestitev v produkciji v novembru tekočega leta in do 15.12. v tekočem letu za namestitev v produkciji v maju naslednjega leta. Za zakonske in sistemske spremembe ter odpravo ugotovljenih nepravilnosti delovanja programske opreme to ne velja, zanje veljajo zakonski roki, nepravilnosti pa se odpravljajo sproti. Kljub temu, da so določeni točni datumi, ni nujno, da se prenosi na vsa testna ali vsa produkcijska okolja strank izvajajo istočasno. Lahko namreč pride do razlik tudi nekaj tednov ali mesecev. Stranka lahko na lastno željo kakšno različico preskoči, če v njej ni kakšnih obveznih zakonskih sprememb ali naročenih funkcionalnosti.

3.3 Vrste programskih datotek

Programska koda navedenega sistema je napisana v programskem jeziku PL/SQL. V fazi razvoja jo, poleg namestitve v razvojno bazo, shranjujemo tudi v znakovne datoteke. Vsak bazni objekt ima rezerviran svoj tip znakovne datoteke (tabeli 3.1 in 3.2). Dvojiške datoteke grafičnega uporabniškega vmesnika in izpisov so opisane v tabeli 3.3. Tudi tu velja, da je vsaka vnosna/poizvedbena zaslonska maska oz. izpis shranjen v svojo datoteko.

Tabela 3.1: SQL datoteke

Končnica	Opis
*.con	omejitev (constraint)
*.idx	indeks
*.role	pooblastila
*.seq	sekvenca
*.snap	materializiran pogled
*.snaplog	dnevnik materializiranega pogleda
*.tab	tabela
*.typ	objektni tip
*.user	uporabnik s pooblastili
*.vew	pogled (view)
*.user	kreiranje uporabnika s pripadajočimi pooblastili
*.role	kreiranje pooblastil

Tabela 3.2: PL/SQL datoteke

Končnica	Opis
*.pb	telo paketa
*.pf	funkcija
*.pp	procedura
*.ps	specifikacija paketa
*.trg	prožilec na tabeli

Tabela 3.3: Dvojiške datoteke

Končnica	Opis
*.fmb	zaslonska maska (Oracle Forms module)
*.mmb	meni (Oracle Forms menu)
*.rdf	izpis (Oracle Reports module)

3.4 Označevanje prenesenih objektov

Ko je datoteka pripravljena za prenos se v sistemu za nadzor različic označi z ustrezno oznako ciljnega sistema v obliki [kratica stranke]_[oznaka okolja]. Na voljo so naslednje oznake okolij:

- test - testno okolje
- predp - predprodukcijsko okolje
- prod - produkcijsko okolje

Označevanje okolij na posameznih objektih nam prinese dve prednosti. Prva je ta, da v vsakem trenutku vemo, katere različice so prenesene na določena okolja, druga pa je ta, da lahko s pomočjo oznak določimo razlike med zadnjo različico in različico, ki je prenesena na okolje. Slednje nam omogoča izdelavo postopka, s katerim za prenos pripravimo samo datoteke, ki so se spremenile od prejšnjega prenosa posledično ob vsakem prenosu nove različice ali popravka ne prenaša celoten sklop datotek. Skupaj je teh datotek več kot 20.000, v skupni velikosti več kot 2 GB.

3.5 Rezultat razdeljevalnega postopka

Rezultat razdeljevalnega postopka je struktura imenikov, kamor so odložene ustrezne datoteke. Ta struktura je osnova za generiranje namestitvenega programa. Namestitveni program se oblikuje s pomočjo odprtokodnega programa NSIS. Vsebina in delovanje skripte za izdelavo namestitvenega programa ni predmet pričujoče diplomske naloge,omenimo le, da od razdeljevalnega postopka zahteva sledečo strukturo imenikov (primer na sliki 3.1):

- bazni_objekti
  - bazna_shema1
  - bazna_shema2
  - bazna_shema3
  - ...
- nebazni objekti
  - končni_imenik1

- končni_imenik2
- končni_imenik3
- ...

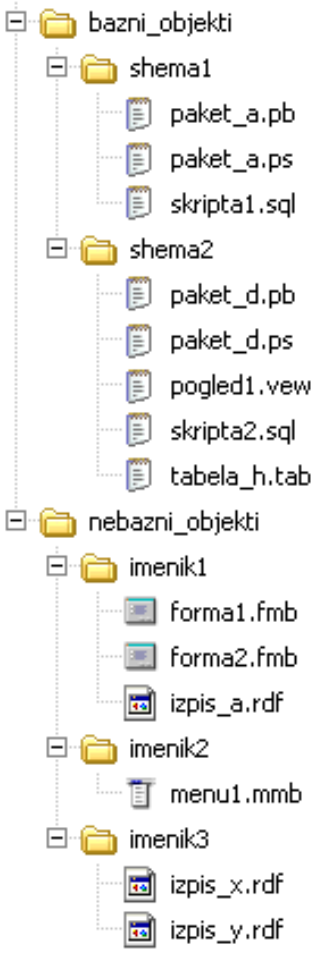
V imeniku bazni_objekti se nahajajo SQL in PL/SQL datoteke. Vsak podimenik je poimenovan z imenom ciljne sheme za namestitev objektov v bazi. V imeniku nebazni_objekti se nahajajo datoteke za zaslonske maske, izpise in menije. Vsak podimenik tako poda ime ciljnega imenika, kamor se prevedeni objekti preslikajo. Namestitvena skripta skrbi za to, da se objekti nameščajo in prevajajo v pravilnem vrstnem redu.

Zgrajeno namestitveno datoteko stranka v svojem okolju izvede sama ali pa namestitev zanjo opravimo mi.

3.6 Nastavitve postopka

Postopek priprave je razdeljen po modulih znotraj podjetja. Vsak modul uporablja nastavitvene datoteke za razdeljevalne postopke, kar omogoči delno izdelavo razdeljevalnega postopka in prinese manj zapletov pripravi namestitve popravkov, ki se lahko na okolja v določenih primerih prenašajo tudi neodvisno. Za vsak sklop sta potrebni dve nastavitve ni datoteki. V prvi so našteta ciljna okolja, ki jih modul podpira. Imena ciljnih okolij so enaka imenom oznak za okolja (opisane v poglavju 3.4). Datoteka vsebuje poljubno število vrstic, vsaka predstavlja ciljno okolje. Vrstica je sestavljena iz štirih kolon. V prvi koloni je ime ciljnega okolja, v drugi veja prenosa, ki določa vejo iz sistema za upravljanje različic, ki se prenaša v ciljno okolje, v tretji koloni je definiran tip okolja (test, produkcija) v četrti pa šifra stranke. Namesto veje okolja v določenih primerih mogoče uporabiti kar oznako drugega ciljnega okolja, kar pomeni, da se po namestitvi različice na enem okolju uskladijo z različicami na drugem. Tak primer je lahko prenos različic, ki so trenutno na testnem okolju v produkcijsko okolje.

Primer (slika 3.2) prikazuje nastavitveno datotko za modul, kjer se na okolja x1_test, x2_test in x3_test prenašajo različice z veje RELEASE_9 (ki označuje 9. različico programske opreme), na okolje x2_prod pa se prenaša starejša različica 8 z veje RELEASE_8. Na naše testne baze hrcx10 in hrcx11 pa se prenašajo zadnje različice potrjenih objektov (veja HEAD). Slednje se ponavadi izvaja samodejno vsako noč in služi za dodatno preverjanje ustreznosti potrjenih različic.



Slika 3.1: Primer datotečne strukture.

```
x1_test   REALEASE_9  test        1234
x1_prod   REALEASE_8  produkcija  1234
x2_test   REALEASE_9  test        2111
x2_prod   REALEASE_9  produkcija  2111
hrcx10    HEAD        test        1111
hrcx11    HEAD        test        1111
```

Slika 3.2: Primer nastavitvene datoteke za ciljna okolja.

```
modul1/direktorij1 bazni_objekti/shema1
modul1/direktorij2 bazni_objekti/shema1
modul1/direktorij1 nebazni_objekti/modul1
```

Slika 3.3: Primer nastavitvene datoteke za imenike modula.

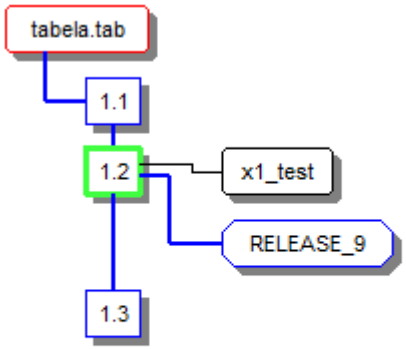
V drugi datoteki so navedeni imeniki, ki se za modul prenašajo na ciljna okolja. Primer je predstavljen na sliki 3.3. V prvi koloni datoteke je naveden imenik izvorne kode iz sistema za upravljanje različic, v drugi koloni pa ciljni imenik za izdelavo namestitvene skripte (po pravilih opisanih v 3.5).

3.7 Postopek priprave

Ob zagonu razdeljevalnega postopka je znan modul, za katerega se postopek izvaja in ciljno okolje. Skripta prebere nastavitve iz ustreznih datotek (poglavje 3.6) in poišče datoteke v navedenih imenikih, ki so kandidati za prenos. To so tiste datoteke, ki imajo zadnjo različico na izbrani veji višjo od različice, kjer je pripeta oznaka ciljnega okolja. Za datoteke, ki temu kriteriju ustrezajo, se zadnje različice na veji izvozijo iz CVS in preslikajo na ciljni imenik za izdelavo namestitvene skripte in označijo kot prenesene na ciljni sistem. Postopek je končan, ko se na ta način analizirajo in obdelajo vsi imeniki navedeni v nastavitveni datoteki. Ponovitev postopka za isti ciljni sistem te datoteke ne prevzame več. Vsak razdeljevaljni postopek izdela tudi razveljavitveno datoteko. Datoteka vsebuje CVS ukaze, ki oznako ciljnega okolja prestavijo nazaj na različico, ki je bila označena pred zagonom razdeljevalnega postopka.

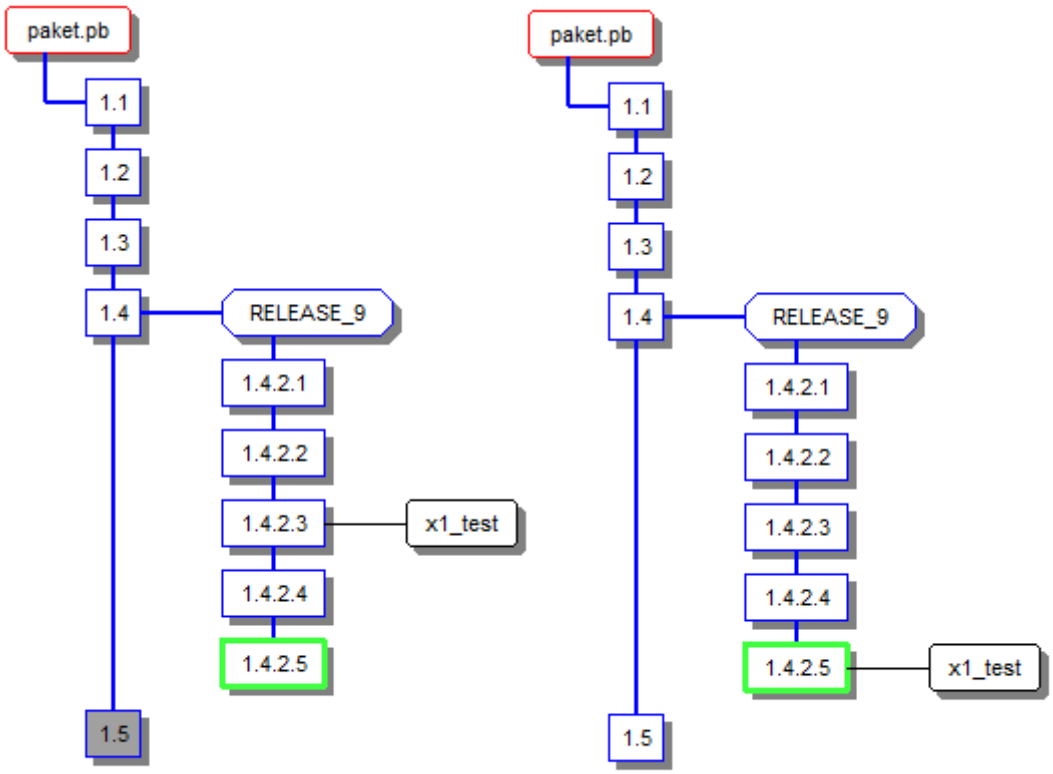
Primer: Na imeniku 'objekti' sta dve datoteki, "paket.pb" in "tabela.tab". Pred razdeljevaljnim postopkom izgleda graf različic tako, kot je prikazan na slikah 3.4 in 3.5 (levi del slike). Požene se razdeljevalni postopek za ciljno okolje x1_test, ki zajame prej navedeni imenik, prenaša pa z veje RELEASE_9. Datoteka "tabela.ta" ima oznako x1_test na zadnji različici te veje zato ni kandidatka za prenos in postopek jo preskoči. Datoteka "paket.p" pa ima oznako ciljnega okolja na različici 1.4.2.3, zadnja različica na veji pa je 1.4.2.5, vsebna te različice se izvozi iz CVS, odloži na ciljni imenik in na zadnjo različico se doda oznaka x1_test. Graf datoteke "paket.pb" po izvedenem postopku je prikazan na sliki 3.5 (desni del slike). V primeru zagona razveljavitvene datoteke se oznaka x1_test na datoteki "paket.pb" prestavi nazaj na različico 1.4.2.3.

Včasih se pojavi potreba po izvozu trenutnih različic, ki so nameščene na



Slika 3.4: Graf različic datoteke tabela.tab pred prenosom.

ciljnem sistem, npr. za ponovna prevajanja izvorne kode ali za arhiviranje kode pri stranki. V tem primeru se postopek pokliče s posebnim stikalom, ki naredi samo izvoz različic, kjer je postavljena navedena oznaka ciljnega sistema (brez primerjav z zadnjimi različicami). Podobno bi sicer bilo mogoče doseči že z vgrajenim CVS ukazom, vendar v tem primeru ne bi bila upoštevana pravila iz nastavitvenih datotek (moduli, ciljni imeniki).



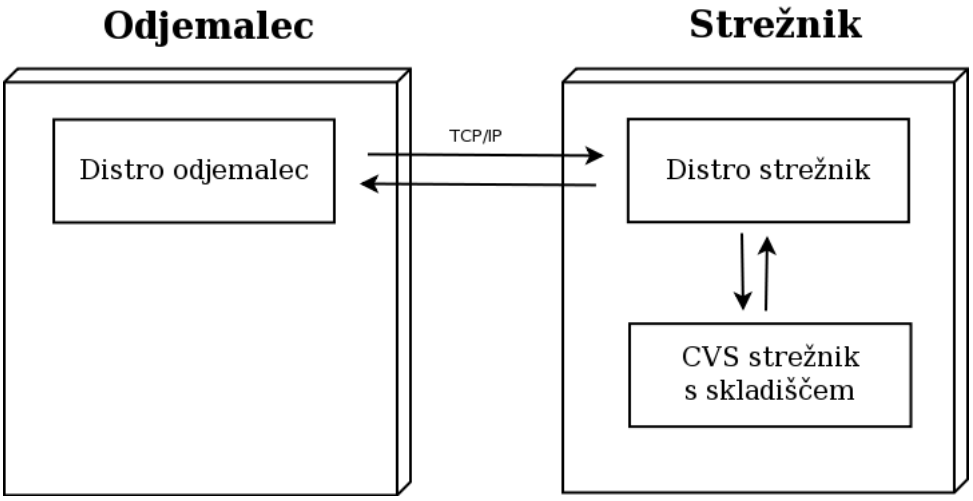
Slika 3.5: Graf različic datoteke paket.pb pred in po prenosu.

Poglavje 4

Razvoj orodja za razdeljevalne postopke

Trenutno imamo za razdeljevalne postopke narejene skripte, ki tečejo na strežniku Linux, kjer se nahaja skladišče CVS. Skripta se je skozi čas spreminjala in dograjevala. V zadnjem času se je pokazalo, da to ni več optimalna pot, njeni uporabi so se pridružili razvijalci večine modulov v podjetju. Za zagon skripte se je potrebno prijaviti na strežnik preko protokola SSH in jo pognati iz ukazne vrstice. Za zagon sta potrebna dva parametra, naziv modula in oznaka ciljnega okolja. Uporabniku je to manj prijazen način dela, težave pa se lahko pojavijo tudi ob napačni uporabi, kot je na primer zagon iz neustreznega imenika. Uporabnik tudi nima pregleda ali morda že tečejo kakšni razdeljevalni postopki na strežniku, ki bi pogojno lahko bili v navzkrižju z njegovim.

V ta namen smo razvili novo različico programske opreme za pripravo objektov namenjenih prenosu na okolje strank. Da bi zagotovili optimalno delovanje izvozov in hkrati prijaznejši uporabniški vmesnik je aplikacija razdeljena na strežnik in odjemalec. Strežniški del teče na istem strežniku kot CVS in vsebuje vso logiko potrebno za pripravo objektov, odjemalec pa je grafični vmesnik, ki strežniku pošilja ukaze ali poizvedbe preko TCP/IP protokola v XML formatu, strežnik pa sporočilo obdela in vrne odgovor (slika 4.1) . Uporabljajo se enake nastavitvene datoteke kot pri starem načinu, opisane v poglavju 3.6. Tako strežnik kot odjemalec sta napisana v programskem jeziku Java in zato primerna za zagon na vseh platformah, za katere obstaja Java odjemalec.



Slika 4.1: Skica arhitekture orodja za razdeljevalne postopke.

4.1 Strežnik

Na strežniku je potrebno namestiti strežniški del aplikacije. Sestavljen je iz Java arhiva, ki se sproži s sledečimi parametri:

- imenik z nastavitvenimi datotekami
- imenik za odlaganje začasnih datotek
- imenik za odlaganje rezultatov razdeljevalnega postpka
- vrata strežnika

Program sprejema zahtevke, katerih obdelave se začnejo izvajati v novih sistemskih procesih (uporablja se nitenje). Tako je strežnik ne glede na dolžino obdelave takoj pripravljen na nov zahtevek.

Vsi zahtevki se pošiljajo v sledečem formatu XML:

```
<Request action=""
         param1=""
         param2=""
         param3=""
         param4="" />
```

V atribut "action" se polnijo oznake zahtevkov, v nadaljevanju imenovane akcija. V atribute param1 do param4 pa se polnijo parametri glede na na akcijo. Opis akcij sledi v nadaljevanju.

4.1.1 Seznam modulov (listModules)

Ko strežnik prejme zahtevek s to akcijo, analizira mapo z nastavitvenimi datotekami in iz nje izlušči seznam modulov, za katere je mogoče izvesti distribucijo. Akcija se kliče brez dodatnih parametrov. Rezultat je XML s seznamom modulov, ki je na voljo odjemalcu za nadaljne obdelave in prikaze. Primer zahtevka je:

```
<Request action="listModules" />
```

Primer odgovora na ta zahtevek pa:

```
<Response>
   <Modul>modul1</Modul>
   <Modul>modul2</Modul>
</Response>
```

4.1.2 Seznam oznak ciljnih sistemov (listTags)

Pri akciji listTags je zahtevan prvi parameter, ki mora biti napolnjen z oznako modula, za katerega strežnik vrne seznam možnih oznak ciljnih sistemov. Seznam se oznak pridobi iz nastavitvene datoteke za oznake (poglavje 3.6, slika 3.2). Primer zahtevka je:

```
<Request action="listTags"
         param1="modul1" />
```

Primer odgovora na ta zahtevek pa:

```
<Response>
   <Tag>x1_test</Tag>
   <Tag>x1_prod</Tag>
   <Tag>x2_test</Tag>
   <Tag>x2_prod</Tag>
   <Tag>hrcx10</Tag>
   <Tag>hrcx11</Tag>
</Response>
```

4.1.3 Seznam načinov zagona razdeljevaljnih postopkov (listOptions)

Akcija ne zahteva parametrov, namenjena pa je prenosu načinov zagona postoopkov do dojemalca. V primeru, da se strežnik dogradi z novim načinom razdeljevalnega postopka ni potrebno dograjevati tudi odjemalca ampak je dovolj da se dopolni polnenje tega seznama. Primer zahtevka je:

```
<Request action="listOptions" />
```

Primer odgovora na ta zahtevek pa:

```
<Response>
   <Option>  | Standarden postopek</Option>
   <Option>-a| Izvoz za oznako</Option>
</Response>
```

4.1.4 Proženje priprave razdeljevalnih postopkov (startDistro)

Ta akcija je jedro strežniškega dela. Poskrbi za razdeljevalni postopek opisan v poglavju 3. V uporabi so štirje parametri, v prvem je naveden modul za katerega se izvaja postopek, v drugem oznaka ciljnega okolja (enako kot pri prejšnji različici postopka), kot tretji parameter pa se strežniku pošlje uporabniško ime s katerim je bil sprožen postopek na odjemalcu. Četrti parameter je rezerviran za dodatna stikala. Strežnik vrne potrditev (status 1) ali zavrne zahtevek z napako (status 2). Zahtevek je zavrnjen v primeru, da za isti modul in isto oznako ciljnega sistema že teče razdeljevalni postopek ali v primeru nepredvidenih napak pri zagonu postopka. Primer zahtevka je:

```
<Request action="startDistro"
         param1="modul1"
         param2="x1_test"
         param3="uporabnik"
         param4="" />
```

Primer odgovora na ta zahtevek pa:

```
<Response action="startDistro"
          status="1"
          statusMsg="Postopek je nastavljen na strežniku" />
```

Po uspešni sprožitvi se na strežniku prične razdeljevalni postopek. Za komunikacijo s CVS strežnikom se uporablja vmesnik, ki proži ukaze preko ukazne vrstice operacijskega sistema. Na vsakem imeniku, ki je določen za prenos, se najprej izvede ukaz:

```
cvs rls -e -r [veja_prenosa] [imenik]
```

Ukaz vrne seznam objektov iz CVS z zadnjimi različicami na veji [veja_prenosa] v imeniku [imenik]. Rezultati ukaza so v obliki primerni za nadaljno obdelavo (za kar poskrbi stikalo -e). Rezultat ukaza za vejo prenosa za isti primer kot je opisan v poglavju 3.7:

```
/paket.pb/1.4.2.5/Wed Apr 13 13:07:07 2011/kv/TRELEASE_9
/tabela.tab/1.2/Thu Apr 7 12:31:58 2011/kv/TRELEASE_9
```

Vsebina se shrani v zgoščevalno tabelo, pri čemer je ključ tabele ime datoteke, vrednost pa različica datoteke. Naslednji ukaz pridobi različice, kjer so trenutno postavljene oznake ciljnega sistema. Ukaz je sestavljen podobno kot prejšnji, s to razliko, da se namesto veje prenosa kot parameter pošlje oznaka ciljnega sistema. Za navedeni primer dobimo sledeči rezultat:

```
/paket.pb/1.4.2.3/Wed Apr 6 09:01:05 2011/kv/Tx1_test
/tabela.tab/1.2/Thu Apr 7 12:31:58 2011/kv/Tx1_test
```

Rezultati tega ukaza se shranijo v drugo zgoščevalno tabelo v enaki obliki.

Nadaljnja obdelava analizira obe tabeli. Če je v drugi tabeli različica nižja kot v prvi ali če datoteka v drugi tabeli ne obstaja (še nikoli ni bila prenesena na ciljni sistem), se različica datoteke iz prve tabele prenese na ciljni imenik in ustrezno označi v CVS.

Operacija za vsako datoteko se zabeleži v dnevnik razdeljevalnega postopka, poleg tega pa se izdela tudi razveljavitvena datoteka, ki omogoči postavitev oznak ciljnega sistema v stanje pred zagonom postopka.

V primeru, da se ne dela običajen razdeljevalni postopek ampak samo izvoz (stikalo v četrtem parametru), se izdela samo seznam različic datotak z oznako ciljnega sistema. Iz tega seznama se vse prenese na ciljne imenike, brez primerjav z zadnjimi različicami.

4.1.5 Seznam postopkov v izvajanju (listActive)

Akcija vrne seznam postopkov, ki se na strežniku trenutno izvajajo. V uporabi je parameter z modulom, za katerega strežnik vrne seznam trenutno aktivnih

postopokov z oznako ciljnega sistema, uporabniškim imenom izvajalca postopka in čas zagona postopka. Seznam se pridobi s pomočjo posebne začasne datoteke, ki jo razdeljevalni postopek naredi na začetku in izbriše na koncu postopka. Primer zahtevka je:

```
<Request action="listActive"
         param1="modul1" />
```

Primer odgovora na ta zahtevek pa:

```
<Response>
   <Distro modul="modul1"
           tag="x1_test"
           osuser="uporabnik1"
           started="20110417131556"
           finished="" />
   <Distro modul="modul1"
           tag="hrcx11"
           osuser="uporabnik2"
           started="20110417131711"
           finished="" />
</Response>
```

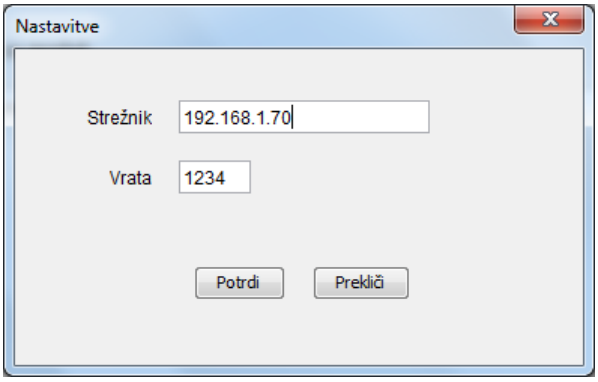
4.1.6 Seznam zaključenih postopkov (listFinished)

Akcija ima podobno vlogo kot listActive, le da vrne seznam zaključenih postopkov. Strukture za zahtevek in odgovor so podobne, s to razliko, da je pri zahtevku mogoče kot parameter poslati tudi oznako ciljnega sistema, pri odgovoru pa je napolnjeno polje s časom zaključka postopka.

4.1.7 Prikaz dnevnika izvajanja postopka (showLog)

Ta akcija vrne dnevnik izvajanja postopka. Parametri so modul, oznaka in čas začetka postopka, ki enoznačno določijo eno izvajanje. Akcija se lahko pokliče na končanih ali aktivnih postopkih, slednje je namenjeno sprotnemu preverjanju stanja postopka. Odgovor izjemoma ni XML datoteka ampak vsebina datoteke dnevnika, kjer se nahaja ime imenika, ime datoteke, predhodna različica in različica prenosa. Primer zahtevka je podan spodaj:

```
<Request action="showLog"
         param1="modul1"
```



Slika 4.2: Nastavitev strežnika v odjemalcu.

```
param2="x1_test"
param3="20110417131556"/>
```

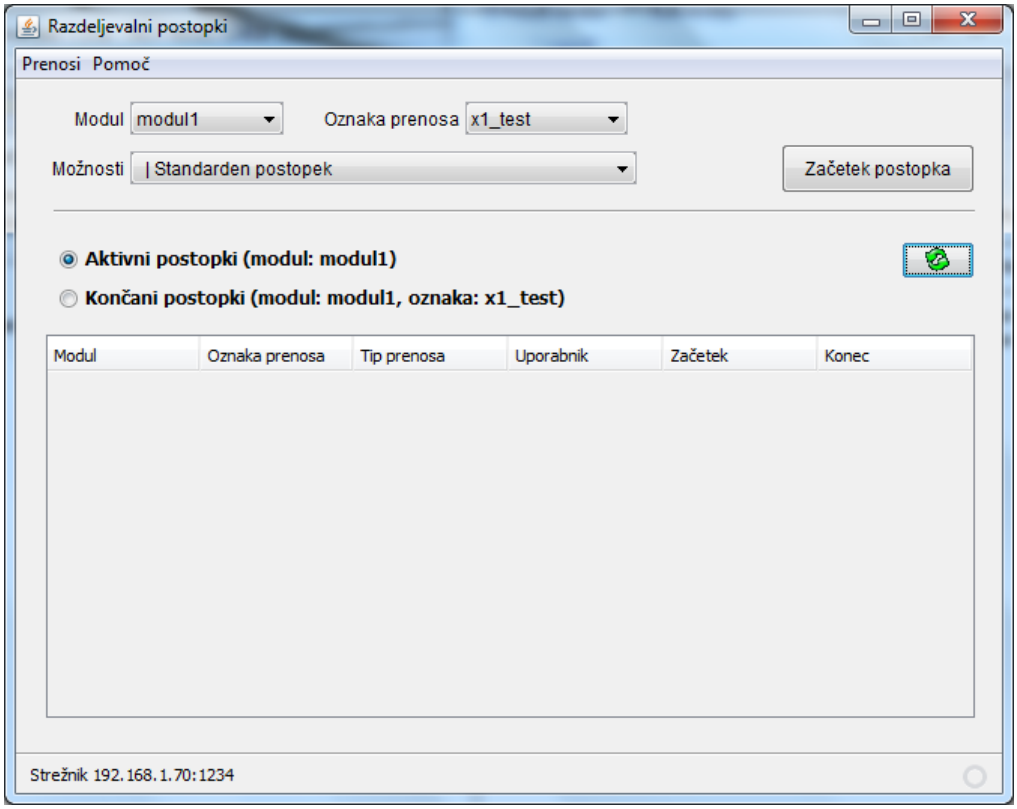
Primer odgovora na ta zahtevek pa je sledeč:

```
skupno/test/dat2.sql 1.18 1.23
skupno/test/paket.pb 1.4.2.5 1.4.2.6
skupno/test/dat3.sql 1.17 1.18
skupno/test/dat6.sql 1.3 1.11
skupno/test/dat5.sql null 1.4
skupno/test/dat1.sql null 1.26
skupno/test/dat8.sql 1.21 1.25
skupno/test/dat7.sql 1.14 1.17
```

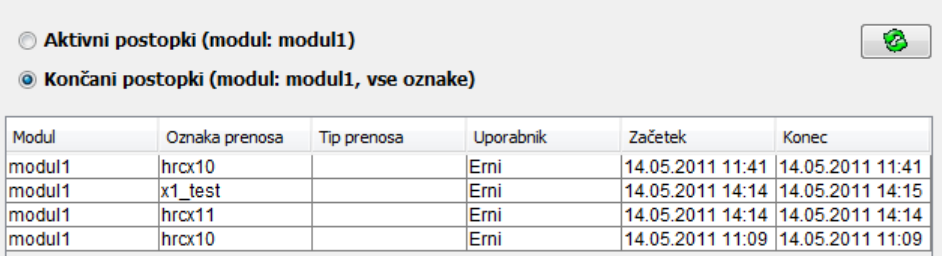
4.2 Odjemalec

Odjemalec je program z grafičnim vmesnikom za pošiljanje ukazov strežniškemu delu programa. Sestavljen je iz glavnega zaslona in dveh pomožnih. Ob prvi prijavi v odjemalca se uporabniku odpre zaslon, kamor je potrebno vpisati podatke o strežniku, IP naslov ali mrežno ime ter vrata strežnika (slika 4.2). Okno z nastavitvami je mogoče odpreti tudi naknadno iz izbirnika. Po potrditvi vnosa se odpre glavni zaslon (slika 4.3). Nastavitve strežnika in vrat se shranijo in jih ob zagonu ni potrebno vsakič znova vnašati.

Komunikacija med odjemalcem in strežnikom poteka v naslednjem zaporedju. Najprej odjemalec pošlje strežniku zahtevek za seznam modulov (4.1.1), ki jih je mogoče izbrati v padajočem seznamu Modul. Ko uporabnik



Slika 4.3: Glavni zaslon odjemalca.

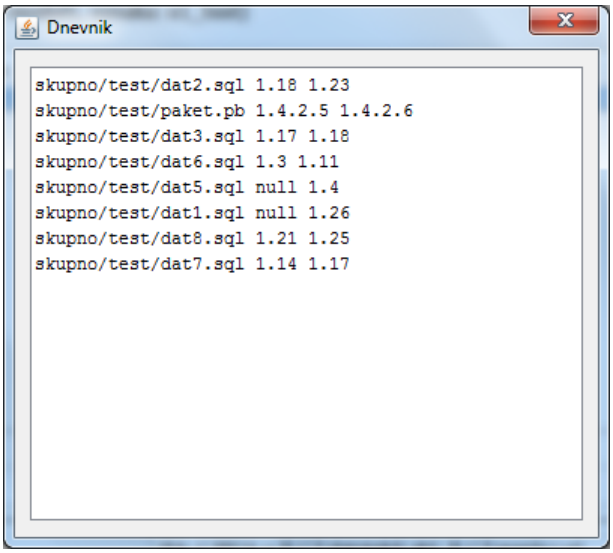


Modul	Oznaka prenosa	Tip prenosa	Uporabnik	Začetek	Konec
modul1	hrcx10		Erni	14.05.2011 11:41	14.05.2011 11:41
modul1	x1_test		Erni	14.05.2011 14:14	14.05.2011 14:15
modul1	hrcx11		Erni	14.05.2011 14:14	14.05.2011 14:14
modul1	hrcx10		Erni	14.05.2011 11:09	14.05.2011 11:09

Slika 4.4: Seznam končanih postopkov za modul1.

izbere modul iz seznama oz. ga kasneje spremeni, se na strežnik pošlje zahtevek za seznam oznak ciljnega sistema (4.1.2). Le te se napolnijo v padajoči seznam Oznaka prenosa. Izbor modula vpliva tudi na seznam postopkov. Glede na izbor se pošlje zahtevek za aktivne ali končane postopke (4.1.5, 4.1.6). Rezultati se izpišejo v tabelo v spodnjem delu ekrana (primer na sliki 4.4). Tabela se osvežuje periodično, kar pomeni da se vsakih nekaj sekund ponovno pošlje na strežnik zahtevek z izbranimi parametri. V primeru dvakratnega klika z miško na zapis v tabeli, se na strežnik pošlje zahtevek za dnevnik postopka (4.1.7), na odjemalcu pa se odpre novo okno z vsebino dnevnika (primer na sliki 4.5). Če gre za aktivni postopek, se zahtevki za dnevnik pošiljajo periodično vsakih nekaj sekund, zato se sproti vidijo trenutno obdelane datoteke in različice.

Ko sta modul in oznaka ciljnega sistema izbrana, se zahtevek za nov razdeljevalni postopek (4.1.4) sproži s klikom na gumb začetek postopka.



Slika 4.5: Prikaz dnevnika razdeljevalnega postopka.

Poglavje 5

Sklepne ugotovitve

V okviru diplomske naloge smo si ogledali sisteme za upravljanje z različicami in razvili sistem za pripravo razdeljevalnih postopkov. Izkazalo se je, da uporaba takega sistema priprave prenosov programov na okolja strank ne le olajša to opravilo, ampak tudi v veliki meri zmanjša število napak. Izločena je namreč človeška komponenta pri določanju tega, katere datoteke in katere različice je potrebno prenesti za namestitev določenega popravka ali sklopa programske opreme.

Vrnimo se najprej na primerjavo sistemov CVS in Subversion. Opisani razdeljevaljni postopek v takšni obliki ni mogoče uporabiti s sistemom Subversion zaradi že omenjene omejitve pri postavljanju oznak. Namreč, če ne upoštevamo optimalnosti pri dodajanju oznak in vej, ima CVS, vsaj po našem mnenju in za naš način uporabe, bistveno prednost pred Subversion ravno v sistemu oznak in vej. Naši razdeljevalni postopki temeljijo na premikanju posameznih oznak na posameznih datotekah, česar Subversion ne omogoča. Najmanjša enota za pripenjanje oznake ali veje imenik, sama operacija pa je izvedena z ukazom "kopiraj", kar prej omenjeno možnost prestavljanja oznak onemogoči. Na splošno pri programskih jezikih, ki izvorno kodo prevajajo v izvršilno datoteko do takšnih problemov ne prihaja, saj tam premikanje oznak in tovrstno označevanje različic niti ni smiselno, vedno je potreben celoten sklop datotek, ki se ponavadi prevede na razvojnih sistemih, v našem primeru pa se izvorna koda prevaja na ciljnem sistemu, kjer so objekti, ki niso bili spremenjeni že na voljo in zato lahko s pomočjo tovrstnega označevanje pripravljamo učinkovitejše namestitvene skripte (prenesejo se samo spremenjene datotke, ki so potrebne za vzpostavitev željene različice).

Da bi lahko uvedli Subversion v naših projektih bi morali korenito spremeniti sistem za pripravo razdeljevalnih postopkov in sistem označevanja ra-

zličic, ki so nameščene na posameznih ciljnih podatkovnih zbirkah. Po dolgotrajni uporabi CVS-a (večanje števila različic) postane tudi postavljanje oznak v CVS težavno. Ob postavljanju oznake se vsebina datoteke preslika v začasno datoteko, ki se potem z novo oznako preimenuje nazaj v originalno. To je precej potratno kar se tiče časa in sistemskih virov, saj velikost datotek z novimi različicami samo rastejo, še posebej pa je to problem pri dvojiških datotekah. Delna (in začasna) rešitev bi bila arhiviranje starejših različic v drugo skladišče in brisanje teh različic iz delovnega skladišča. Dolgoročnejša rešitev pa zamenjava sistema za upravljanje različic, ki bi to pomanjklivost odpravljal, vseeno pa omogočal plavajoče oznake. Trenutno smo v podjetju v fazi iskanja nadomestnega sistema, ki bi izpolnjeval naše zahteve.

Literatura

[1] J Loeliger, *Version Control with Git*, O'Reilly Media, 2009.

[2] C. M. Pilato, B. Collins-Susman, B.W. Fitzpatrick, *Version Control with Subversion*, O'Reilly Media, Second Edition, 2008.

[3] J. Vesperman, *Essential CVS*, O'Reilly Media, Second Edition, 2006.

Dodatek A

Deli izvorne kode strežnika

A.1 Main.java

```
package com.hrc.cvs.distrobranch;

public class Main {
    public static void main(String[] args) {
        Exception excUporaba =
                    new Exception("Uporaba: distro [-a] oddelek oznaka\n"
                                                    + "          distro -s vrata\n"
                                                    + "  -a: izvoz za oznako\n"
                        + " -s: strezniski nacin, drugi parameter obvezno vrata\n\n");
        try {
            if (args.length < 2) {
                throw excUporaba;
            } else {
                String flags = "";
                if (args[0].startsWith("-")) {
                    flags = args[0];
                }
                if (flags.equals("-s")) {
                    int port;
                    try {
                        port = Integer.parseInt(args[1]);
                    } catch (NumberFormatException e) {
                        throw excUporaba;
                    }
                    System.out.println("Strezniski nacin ...\n");
                    DServer.listen(port);
                } else {
                   if (!flags.equals("")) {
                       if (args.length < 3) {
                           throw excUporaba;
                       }
                       args[0] = args[1];
                       args[1] = args[2];
                       args[2] = null;
                    }
                    System.out.println(args[0]+" "+args[1]+" "+flags);
                    Distro d = new Distro(args[0],args[1],flags,"erni");
                    d.setDebug(200);
                    d.prepareDistro();
                }
            }
        } catch (Exception e) {
            System.err.println(e.getMessage());
        }
    }
}
```

A.2 Dserver.java

```
package com.hrc.cvs.distrobranch;

import java.io.*;
import java.net.*;
import java.util.*;
import org.jdom.*;
import org.jdom.output.*;
import org.jdom.input.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

/*
   Razred za zagon strezniskega dela.
   Sprejemanje zahtevkov in jih vecnitno obdelovanje
*/
public class DServer {

    static Socket clientSocket = null;
    static ServerSocket serverSocket = null;

    public static void listen(int port_number) {

        try {
            serverSocket = new ServerSocket(port_number);
        }
        catch (IOException e)
            {System.out.println(e);}

        while(true){
            try {
                clientSocket = serverSocket.accept();
                (new clientThread(clientSocket)).start();
            }
            catch (IOException e) {
                System.out.println(e);}
        }
    }
}

class clientThread extends Thread{

    BufferedReader is = null;
    PrintStream os = null;
    Socket clientSocket = null;

    public clientThread(Socket clientSocket){
        this.clientSocket=clientSocket;
    }

    private void parseLine(String xml) {
        try {
            System.out.println(xml);
            InputStream xis = new ByteArrayInputStream(xml.getBytes("UTF-8"));
            DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document doc1 = (new DOMBuilder()).build(db.parse(xis));
            Element root = doc1.getRootElement();
            String action = (root.getAttribute("action")).getValue();
            String param1;
            String param2;
            String param3;
            String param4;
            try {
                param1 = (root.getAttribute("param1")).getValue();
            } catch (Exception e) {
                param1 = "";
            }
            try {
                param2 = (root.getAttribute("param2")).getValue();
            } catch (Exception e) {
                param2 = "";
            }
            try {
                param3 = (root.getAttribute("param3")).getValue();
            } catch (Exception e) {
                param3 = "";
```

```
            }
            try {
                param4 = (root.getAttribute("param4")).getValue();
            } catch (Exception e) {
                param4 = "";
            }
            if (action.equals("startDistro")) {
                //zacetek distrota!
                //os.println("Zacenjam distro za " + param1 + " - " + param2 + "!!");
                String status;
                String statusMsg;
                try {
                    Distro d = new Distro(param1,param2,param3,param4);
                    statusMsg = d.getStatus();
                    status = "1";
                    d.start();
                } catch (Exception e) {
                    statusMsg = e.getMessage();
                    status = "2";
                    e.printStackTrace();
                }
                //response
                Element response = new Element("Response");
                Document doc = new Document(response);
                response.setAttribute("action", action);
                response.setAttribute("status", status);
                response.setAttribute("message", statusMsg);
                String ret = (new XMLOutputter()).outputString(doc);
                os.println(ret);
            } else if (action.equals("listActive")) {
                os.println(DistroParse.listActive(param1));
            } else if (action.equals("listFinished")) {
                os.println(DistroParse.listFinished(param1,param2));
            } else if (action.equals("listModules")) {
                os.println(DistroParse.listModules());
            } else if (action.equals("listTags")) {
                os.println(DistroParse.listTags(param1));
            } else if (action.equals("listOptions")) {
                os.println(DistroParse.listOptions());
            } else if (action.equals("getLog")) {
                os.println(DistroParse.getLog(param1,param2,param3));
                //os.println(DistroParse.listOptions());
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void run() {
            String line;
            String name;
            try {
            is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
                os = new PrintStream(clientSocket.getOutputStream());
            line = is.readLine();
            parseLine(line);
            is.close();
            os.close();
            clientSocket.close();
            } catch(IOException e) {};
    }
}
```

A.3 Distro.java

```
package com.hrc.cvs.distrobranch;

import java.util.*;
import java.io.*;

/*
  Razred za uporavljanje z razdeljevalnimi postopki
*/
public class Distro extends Thread {
```

```
public static final String SETUPDIR_DEFAULT = "distro_setup";
public static final String CVS_DEFAULT = "cvs";
public static final String EXPORTDIR_DEFAULT = "distro_test";
public static final String TMPDIR_DEFAULT = "temp";

private String distroTag;
private HashMap tags;
private String[] tagSetup;
private String[] dirs;
private int logging = 0;
private String setupDir;
private String cvs;
private String exportDir;
private String tmpDir;
private BufferedWriter bwLog;
private BufferedWriter bwUndo;
private BufferedWriter bwLock;
private String distroType;
private HashMap activeDistros = new HashMap();
private String startTime;
private String lockFilename;
private String finishedFilename;
private String distroModul;

/**
 * Preveri ali obstaja oznaka v hash tabeli parametrizitanih tagov
 *
 * @param tag Naziv oznake, ki se preverja
 */
private void checkTag(String tag) throws Exception {
    //preveri ali je tag v hashtabeli, nalozi
    if (!tags.containsKey(tag)) {
        throw new Exception("Tag "+tag+" ne obstaja !!!");
    }
    String v = tags.get(tag).toString();
    tagSetup = v.split(" ");

}

/**
 * Metoda pretvori zapise, ki jih vrne ukaz cvs rls v urejen
 * seznam objektov z verzijami.
 *
 * @param lines Seznam vrstic ukaza cvs rls
 * @param idx Oznaka objekta
 * @param revList In/out parameter s hash tabelo objektov
 */
private void linesToRevList(String[] lines, int idx, HashMap revList) {
    String[] line = new String[6];
    for (int i=0;i<lines.length;i++) {
        if ((lines[i].length() > 0) && (lines[i].substring(0,1).equals("/"))) {
            line = lines[i].split("/");
            revList.put(line[1], line[2]+" "+String.valueOf(idx) +" "+line[5].substring(1));
            //printDebug(line[1] +" = "+revList.get(line[1]).toString(),200);
            //sestavimo ime_objekta = verzija idx tag
        }
    }
}

/**
 * Metoda za izvoz datoteke iz CVS-ja
 *
 * @param cvsDir Pot CVS repozitorija
 * @param fileName Naziv obravnavane datoteke
 * @param branch Naziv oznake ali veje za katero se dela izvoz
 * @param destDir Ciljni imenik za izvoz
 */
private void exportFile(String cvsDir, String fileName,
                        String branch, String destDir) throws Exception {
    printDebug("exporting "+fileName,10);

    //Ce datoteka obstaja na cilju, jo najprej odstrani, sicer export pade
    File f = new File(destDir + "/" + fileName);
    File fB = new File(destDir + "/" + fileName+ ".backup");
    if (f.exists()) {
        f.renameTo(fB);
    }
    //printDebug(cvs + " export -f -l -r " + branch + " -d " + destDir +
                " " + cvsDir + "/" + fileName,200);
    String[] lines = HRCUtil.runCmd(cvs +
```

```
                                        "-Q-export-f-l-r-" +
                                        branch + "-d-" + destDir + "-" +
                                        cvsDir + "/" + fileName);
        //ce pride do napake pri izvozu (ne obstaja datoteka) jo logiramo in povremo backup
        if (!f.exists()) {
            printDebug("Napaka-pri-izvozu:-"+lines[0],0);
            if (fB.exists()) {
                fB.renameTo(f);
             }
        } else {
            fB.delete();
        }
    }

    /**
     * Metoda za logiranje sprememb v log in undo datoteko ter oznacevanje
     * oznake prenosa na ustrezni verziji objekta.
     *
     * @param cvsDir Pot CVS repozitorija
     * @param fileName Naziv obravnavane datoteke
     * @param rev Obstojeca verzija (pred prenosom)
     * @param head Nova verzija (prenesena)
     * @param branch Naziv oznake ali veje prenosa
     * @throws Exception
     */
    private void processTag(String cvsDir, String fileName, String rev,
                            String head, String branch) throws Exception {
        String cvsFile = cvsDir+"/"+fileName;
        bwLog.write(cvsFile+"-"+rev+"-"+head);
        bwLog.newLine();
        bwLog.flush();
        if (rev == null) {
            bwUndo.write("cvs-rtag-F-l-d-"+distroTag+"-"+cvsFile);
            bwUndo.newLine();
        } else {
            bwUndo.write("cvs-rtag-F-l-r-"+rev+"-"+distroTag+"-"+cvsFile);
            bwUndo.newLine();
        }
        bwUndo.flush();
        String lines[] = HRCUtil.runCmd(cvs + "-rtag-l-f-F-r-"+branch+
                                        "-"+distroTag+"-"+cvsFile);
    }

    /**
     * Priprava izvoza datotek datotek znotraj enega imenika
     * v CVS repozitoriju s premikom oznak
     *
     * @param cvsDir Pot CVS repozitorija
     * @param destDir Pot ciljnega imenika
     * @param sourceRevSQL Parameter za izdelavo SQL skripte s podaki o verzijah (D/N)
     * @param fixedBranch Fiksna oznaka/veja za direktorij, prevlada glede na oznako prenosa
     * @throws Exception
     */
    private void tagAndExportDir(String cvsDir, String destDir, String sourceRevSQL,
                                 String fixedBranch) throws Exception {
        HashMap revList = new HashMap();
        HashMap headList = new HashMap();
        String[] lines;
        String fullDestDir = exportDir+"/"+distroTag+"/"+destDir;
        (new File(fullDestDir)).mkdirs();

        //runCmd(CVS+" rls skupno/proc");
        printDebug(cvsDir+"-"+sourceRevSQL+"-"+fixedBranch, 200);
        String tmpParamBranch = tagSetup[0];
        String tmpParamBranch1 = tagSetup[1];
        if (!(fixedBranch == null)) {
            tmpParamBranch = fixedBranch;
            tmpParamBranch1 = "";
        }
        printDebug("ParamBranch="+tmpParamBranch+";-"+"ParamBranch1="+tmpParamBranch1, 200);
        lines = HRCUtil.runCmd(cvs+"-rls-e-r-"+tmpParamBranch+"-"+cvsDir);
        linesToRevList(lines,1,headList);
        if (!(tmpParamBranch.equals("")) && !(tmpParamBranch1.equals(tmpParamBranch))) {
            lines = HRCUtil.runCmd(cvs+"-rls-e-r-"+tmpParamBranch1+"-"+cvsDir);
            linesToRevList(lines,0,headList);
        }
        // $CVS rls -e -r $PARAM $1
        lines = HRCUtil.runCmd(cvs+"-rls-e-r-"+distroTag+"-"+cvsDir);
        linesToRevList(lines,2,revList);
```

```
        Iterator i;
        i = headList.entrySet().iterator();
        while (i.hasNext()) {
            Map.Entry me = (Map.Entry)i.next();
            String fileName = me.getKey().toString();
            //printDebug(me.getKey().toString() + " - " + me.getValue().toString(),200);
            //kontrola ver banka od fajla (+dodaj)
            String[] headSplit = me.getValue().toString().split(" ");
            if (revList.containsKey(fileName)) {
                String[] revSplit = revList.get(fileName).toString().split(" ");
                if (!(revSplit[0].equals(headSplit[0]))) {
                    exportFile(cvsDir,fileName,headSplit[2],fullDestDir);
                    processTag(cvsDir,fileName,revSplit[0],headSplit[0],headSplit[2]);
                }
            } else {
                exportFile(cvsDir,fileName,headSplit[2],fullDestDir);
                processTag(cvsDir,fileName,null,headSplit[0],headSplit[2]);
            }
        }
    }

    /**
     * Priprava izvoza datotek datotek znotraj enega imenika
     * v CVS repozitoriju na osnovi obstojecih oznak
     *
     * @param cvsDir
     * @param destDir
     * @param sourceRevSQL
     * @param fixedBranch
     */
    private void exportDir(String cvsDir, String destDir,
                           String sourceRevSQL,String fixedBranch) throws Exception {
        HashMap revList = new HashMap();
        String[] lines;

        String fullDestDir = exportDir+"/"+distroTag+"/"+destDir;
        (new File(fullDestDir)).mkdirs();

        lines = HRCUtil.runCmd(cvs+" rls -e -r "+distroTag+" "+cvsDir);
        linesToRevList(lines,2,revList);

        Iterator i;
        i = revList.entrySet().iterator();
        while (i.hasNext()) {
            Map.Entry me = (Map.Entry)i.next();
            String fileName = me.getKey().toString();
            String[] revSplit = revList.get(fileName).toString().split(" ");
            exportFile(cvsDir,fileName,revSplit[2],fullDestDir);
            bwLog.write(cvsDir+"/"+fileName+" "+revSplit[0]);
            bwLog.newLine();
            bwLog.flush();
        }
    }

    /**
     * Metoda za zapis debug sporocil
     * @param logText Tekst sporocila
     * @param logLevel Nivo sporocanja
     */
    private void printDebug(String logText, int logLevel) {
        if (logging >= logLevel) {
            System.out.println(logText);
        }
    }

    /**
     * Inicializacija zapisovanja debug sporocil
     * @param logLevel Nivo sporocanja
     */
    public void setDebug(int logLevel) {
        logging = logLevel;
    }

    /*
     * Metoda za zacetek razdeljevalnega postopka
     */
    public void prepareDistro() throws Exception {
        try {
            tags = DistroParse.getTags(distroModul);
            dirs = DistroParse.getDirs(distroModul);
```

```
            checkTag(distroTag);
            String[] dirConfTmp;
            String[] dirConf = new String[5];
            int i = 0;
            while (dirs[i] != null) {
                printDebug(dirs[i],500);
                dirConfTmp = dirs[i].split("␣");
                dirConf[4] = null;
                System.arraycopy(dirConfTmp, 0, dirConf, 0, dirConfTmp.length);
                if (!distroTag.substring(1,3).equals("hrc") || !dirConf[2].equals("N")) {
                    if (distroType.equals("-a")) {
                        exportDir(dirConf[0],dirConf[1],dirConf[3],dirConf[4]);
                    } else {
                        tagAndExportDir(dirConf[0],dirConf[1],dirConf[3],dirConf[4]);
                    }
                }
                i++;
            }
            bwLock.write("Distribucija␣uspesno␣koncana");
        } catch (Exception e) {
            //bwLock.write(e.getStackTrace().toString());
            bwLock.write("Error:"+ e.getMessage());
            bwLock.newLine();
            StackTraceElement[] x = e.getStackTrace();
            for (int i=0; i<x.length; i++) {
               bwLock.write(x[i].getClassName() +
                            "." + x[i].getMethodName() +
                            ":␣"+String.valueOf(x[i].getLineNumber()));
               bwLock.newLine();
            }
        } finally {
            bwLog.close();
            bwUndo.close();
            bwLock.close();
            printDebug("Rename␣" + lockFilename + "␣->␣" +
                       finishedFilename+"."+HRCUtil.getTime());
            File fFin = new File(finishedFilename+"."+HRCUtil.getTime());
            (new File(lockFilename)).renameTo(fFin);
            System.out.println("konc");
        }
    }

    /*
     * Konstruktor, nastavi zacetne parametre distribucije
     * config: po modulih, npr: salter, devizno...
     * tag: razdeljevalna oznak (ciljno okolje)
     * flags: -a prenos vsega
     */
    public Distro(String modul, String tag, String flag, String osuser) throws Exception {
        distroTag = tag;
        distroType = flag;
        setupDir = System.getProperty( "com.hrc.distrobranch.SETUPDIR",SETUPDIR_DEFAULT);
        cvs = System.getProperty( "com.hrc.distrobranch.CVS",CVS_DEFAULT);
        exportDir = System.getProperty( "com.hrc.distrobranch.EXPORTDIR",EXPORTDIR_DEFAULT) +
                                        "/prenos."+modul;
        tmpDir = System.getProperty( "com.hrc.distrobranch.TMPDIR",TMPDIR_DEFAULT);
        (new File(exportDir+"/cvslogs")).mkdirs();
        FilenameFilter x = new NameFilter("running." + modul + "." + distroTag);
        String[] list = (new File(exportDir + "/cvslogs")).list(x);
        if (list.length > 0) {
           throw new Exception("Distro␣" + modul + "␣-␣" + tag + "␣ze␣tece!");
        }
        startTime = HRCUtil.getTime();
        lockFilename = exportDir + "/cvslogs/running." + modul + "." +
                       distroTag + "." + osuser + "." + startTime;
        finishedFilename = exportDir + "/cvslogs/finished." + modul + "." +
                           distroTag + "." + osuser + "." + startTime;
        bwLock = new BufferedWriter(new FileWriter(lockFilename));
        bwLog = new BufferedWriter((new FileWriter(exportDir + "/cvslogs/prenos␣" +
                                                   distroTag + ".log." + startTime,true)));
        bwUndo = new BufferedWriter((new FileWriter(exportDir + "/cvslogs/prenos␣"  +
                                                    distroTag+".undo."+startTime,true)));
        distroModul = modul;
    }

    public void run() {
        this.setDebug(200);
        try {
            this.prepareDistro();
        } catch (Exception e) {
```

```
            //System.err.println(e.getMessage());
            e.printStackTrace();
        }
    }

    public String getStatus() {
        return lockFilename;
    }
```

A.4 DistroParse.java

```
package com.hrc.cvs.distrobranch;

import java.io.*;
import org.jdom.*;
import org.jdom.output.*;
import org.jdom.input.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.util.*;

/*
   Razred za obdelavo XML zahtevkov in odgovorov
*/
public class DistroParse {

    private static String setupDir =
        System.getProperty("com.hrc.distrobranch.SETUPDIR",Distro.SETUPDIR_DEFAULT);
    private static String exportDir =
        System.getProperty("com.hrc.distrobranch.EXPORTDIR",Distro.EXPORTDIR_DEFAULT);
    private static String tmpDir =
        System.getProperty("com.hrc.distrobranch.TMPDIR",Distro.TMPDIR_DEFAULT);


    public static String listModules() {
        FilenameFilter x = new NameFilter(".setup");
        String[] list = (new File(setupDir)).list(x);
        Element response = new Element("Response");
        Document doc = new Document(response);
        String modul;
        for (int i=0; i<list.length; i++) {
            modul = list[i].substring(7, list[i].indexOf("."));
            response.addContent(new Element("Modul").addContent(modul));
        }
        String ret = (new XMLOutputter()).outputString(doc);
        return ret;
    }

    public static String listTags(String modul) throws Exception {
        Element response = new Element("Response");
        Document doc = new Document(response);
        HashMap tagsMap = getTags(modul);
        SortedSet sortedSet = new TreeSet(tagsMap.keySet());
        Iterator it = sortedSet.iterator();
        String tag;
        while (it.hasNext()) {
          tag = it.next().toString();
          response.addContent(new Element("Tag").addContent(tag));
        }
        String ret = (new XMLOutputter()).outputString(doc);
        return ret;
    }

    public static String listActive(String modul) {
        String filter = "running.";
        if (!(modul==null) && !modul.equals("")) {
            filter = filter + modul + ".";
        }
        FilenameFilter x = new NameFilter(filter);
        String[] list = (new File(exportDir + "/prenos."+modul+"/cvslogs")).list(x);
        Element response = new Element("Response");
        Document doc = new Document(response);
        String[] distroData;
        Element activeDistro;
        if (list != null) {
            for (int i=0; i<list.length; i++) {
```

```
                distroData = list[i].split("\\.");
                activeDistro = new Element("Distro");
                activeDistro.setAttribute("modul", distroData[1]);
                activeDistro.setAttribute("tag", distroData[2]);
                activeDistro.setAttribute("osuser", distroData[3]);
                activeDistro.setAttribute("started", distroData[4]);
                activeDistro.setAttribute("finished", "");
                response.addContent(activeDistro);
            }
        }
        String ret = (new XMLOutputter()).outputString(doc);
        return ret;

    }

    public static String listFinished(String modul, String tag) {
        String filter = "finished."+modul+".";
        if (!(tag==null) && !tag.equals("")) {
            filter = filter + tag + ".";
        }
        System.out.println(filter);
        FilenameFilter x = new NameFilter(filter);
        String[] list = (new File(exportDir+"//prenos."+modul+"//cvslogs")).list(x);
        Element response = new Element("Response");
        Document doc = new Document(response);
        String[] distroData;
        Element activeDistro;
        if (list != null) {
            for (int i=0; i<list.length; i++) {
                distroData = list[i].split("\\.");
                activeDistro = new Element("Distro");
                activeDistro.setAttribute("modul", distroData[1]);
                activeDistro.setAttribute("tag", distroData[2]);
                activeDistro.setAttribute("osuser", distroData[3]);
                activeDistro.setAttribute("started", distroData[4]);
                activeDistro.setAttribute("finished", distroData[5]);
                response.addContent(activeDistro);
            }
        }
        String ret = (new XMLOutputter()).outputString(doc);
        return ret;

    }

    public static String getLog(String modul, String tag, String dat) throws Exception {
        //prenos_rkb_test.log.201010250849
        String logfile = exportDir+"//prenos."+modul+"//cvslogs//prenos_"+tag+".log."+dat;
        String line;
        BufferedReader br;
        String ret = "";
        try {
            br = new BufferedReader(new FileReader(logfile));
            while ((line = br.readLine()) != null) {
                if (ret.equals("")) {
                    ret = ret + line;
                } else {
                    ret = ret + "\n" +line;
                }
            }
            br.close();
        }
        catch (IOException e) {
            System.err.println("Error: " + e);
            throw new Exception("Napaka pri branju dnevnika prenosa!");
        }
        //poslje se vsebino datoteke finished, ce obstaja
        //finished.monitoring.hrcdh.ernest.201010271038
        FilenameFilter x = new NameFilter("finished."+modul+"."+tag, "."+dat+".");
        String[] list = (new File(exportDir+"//prenos."+modul+"//cvslogs")).list(x);
        if (list != null && list.length == 1) {
            br = new BufferedReader(new FileReader(exportDir + "//prenos." + modul +
                                                   "//cvslogs//"+list[0]));
            while ((line = br.readLine()) != null) {
                if (ret.equals("")) {
                    ret = ret + line;
                } else {
                    ret = ret + "\n" +line;
                }
            }
            br.close();
```

```
        }
        return ret;
    }
    public static String listOptions() {
        Element response = new Element("Response");
        Document doc = new Document(response);
        response.addContent(new Element("Option").addContent("  | Standarden postopek"));
        response.addContent(new Element("Option").addContent("-a | Izvoz za oznako"));
        String ret = (new XMLOutputter()).outputString(doc);
        return ret;
    }

    public static HashMap getTags(String modul) throws Exception {
        String line;
        String key;
        String value;
        HashMap tags = new HashMap();
        String tagsFile = "distro_" + modul + ".tags";
        try {
            FileReader file = new FileReader(setupDir+"/"+tagsFile);
            BufferedReader br = new BufferedReader(file);
            while ((line = br.readLine()) != null) {
                if (line.trim().length() == 0 || line.substring(0,1).equals("#")) {
                    continue;
                }
                key = line.substring(0,line.indexOf(' '));
                value = line.substring(line.indexOf(' ')+1);
                tags.put(key, value);
            }
         }
         catch (IOException e) {
            System.err.println("Error: " + e);
            throw new Exception("Napaka pri branju konfiguracijskih datotek!");
         }
         return tags;
    }

    public static String[] getDirs(String modul) throws Exception {
        String line;
        String key;
        String value;

        String[] dirs = new String[50];
        String dirsFile = "distro_" + modul + ".dirs";
        try {
            FileReader file = new FileReader(setupDir+"//"+dirsFile);
            BufferedReader br = new BufferedReader(file);
            int i = 0;
            while ((line = br.readLine()) != null) {
                if (line.trim().length() == 0 || line.substring(0,1).equals("#")) {
                    continue;
                }
                dirs[i++] = line;
            }

            //prebere se dirs
         }
         catch (IOException e) {
            System.err.println("Error: " + e);
            throw new Exception("Napaka pri branju konfiguracijskih datotek!");
         }
         return dirs;
    }

}
```

A.5 HRCUtil.java

```
package com.hrc.cvs.distrobranch;

import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Calendar;

/*
```

```
   Razred za pomozne metode, uporabljene v strezniskem delu
*/
public class HRCUtil {

    /**
     * Metoda za zagon ukaza na operacijskem sistemu
     *
     * @param command Ukaz za zagon
     * @return Seznam vrstic, ki jih ukaz izpise na standarden izhod
     *
     */
    public static String[] runCmd(String command)
      {
        Runtime rt = Runtime.getRuntime();
        int rc = -1;
        String[] ret;
        String output = new String();

        try
        {
            Process p = rt.exec(command);
            int bufSize = 4096;
            BufferedInputStream bis =
            new BufferedInputStream(p.getInputStream(), bufSize);
            int len;
            byte buffer[] = new byte[bufSize];
            String value;
            while ((len = bis.read(buffer, 0, bufSize)) != -1) {
                value = new String(buffer,0,len);
                output = output.concat(value);

             }
            rc = p.waitFor();
        }
        catch (Exception e)
        {
          e.printStackTrace();
          rc = -1;
        }
        finally
        {
          ret = output.split("[\n|\r]");
          return ret;
        }
      }

    public static String getTime() {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmm");
        return sdf.format(Calendar.getInstance().getTime());
    }
}
```