CLEARCASE – ORODJE ZA UPRAVLJANJE KONFIGURACIJE

Uroš Sajko
Hermes SoftLab d.d., Litijska 51, Ljubljana
projekt OmniBack
uros.sajko@hermes.si

Besedilo predstavlja osnovne probleme in zahteve, s katerimi se srečuje upravljavec konfiguracije pri velikem projektu. Predstavljeni so zgodovinski razvoj in orodja, ki omogočajo upravljanje konfiguracije. Eno izmed teh orodij je ClearCase, ki ga uporabljamo tudi pri projektu OmniBack v podjetju Hermes SoftLab. Prikazane so prednosti novega orodja, težave pri prehodu na orodje ClearCase. Opisana so osnovna pravila, ki jih je potrebno upoštevati, da bi lahko vsak član projekta nemoteno delal in je administracija okolja kar najbolj enostavna in razvidna.

Uvod

Pravočasno izdati kvalitetne programske proizvode in jih vzdrževati je osnovni cilj in politika večine programskih organizacij. Preden se proizvod lahko izda, ga je potrebno zgraditi. Za vzdrževanje pa je potrebno vedeti, kaj je tisto (katera inačica izvorne kode), kar stranka ima.

Problem in izziv je razviti programski proces za pretvarjanje tisočev datotek, ki jih spreminja desetine ljudi, v konsistentno celoto kode in dokumentacije. Potrebni so postopki raznih nivojev, predpisi, izbrana orodja in organizacijski napori, da stalne spremembe zahtev, in zato načrtov kode, privedejo do stabilnega proizvoda, primernega za izdajo. Potrebno je določiti vse, od načina shranjevanja in dostopa, do datotek, dogovorov o prevajanju modulov, komponent in celega proizvoda, do banalnosti, kot so varnostne kopije proizvoda v primeru nesreč.

Upravljanje konfiguracije (UK) je organizacija takega programskega okolja. S tem pojmom označujemo danes postopke in metodologije za organizacijo delovnega okolja, administracijo kode in drugih dokumentov, in izgradnjo programskega proizvoda. Kot formalizacija osnovnih politik organizacije je UK eden od temeljev proizvodnje programske opreme.

Upravljanje konfiguracije je hkrati veda in postopek. Njegov namen je zagotoviti organizaciji informacije, ali bo končni proizvod takšen kot je bil načrtovan. Ko govorimo o upravljanju konfiguracije, mislimo na sledenje in kontrolo razvoja programske opreme in povezanih aktivnosti. Ob tem mislimo na vodenje razvoja projektov programske opreme, ko več razvijalcev dela isto izvorno kodo ob istem času, razvija proizvod za več okolij, ob tem pa sta omogočena shranjevanje revizij kode in nadzor nad stanjem kode.

Upravljanje konfiguracije je postopek, ki obsega identifikacijo, organiziranje in kontrolo sprememb razvoja programske opreme v projektni skupini. Med drugim zagotavlja učinkovita orodja z namenom, da bi izdelali pravi proizvod. UK je posebej pomembno danes, ko uporabniki zahtevajo poceni proizvode, pričakujejo hitro dostavo in storitve, ki bi vedno bolj in bolj sovpadala z njihovimi pričakovanji. V teh primerih je UK ključno orožje v konkurenčnem boju.

Pogled v zgodovino

Z razvojem programske opreme se je pojavila potreba po spremembi le te. Prvo "upravljanje konfiguracije" je potekalo ročno. V času, ko so se programi še shranjevali na luknjanih karticah, je bila kontrola revizij izvedena tako, da so kup luknjanih kartic povezali v sveženj, katerega so označili z datumom in imenom programa. Svežnje so shranili, da bi lahko razvijalcem zagotovili vrnitev na prejšnjo verzijo.

Tehnologija je napredovala in fizične medije so zamenjali magnetni. V tem času so se arhivirali koluti trakov, kasneje so kolute zamenjali vedno manjši mediji z vedno večjimi kapacitetami. Kljub vsemu pa je bila v tem času kontrola revizij še vedno ročna, saj je bilo potrebno na vsak medij ročno shraniti določeno verzijo.

Kasneje so se pojavili veliki diski, kar je omogočilo skupno rabo izvorne kode. Odjava (check-out) kode je bila še vedno ročna, kar je pomenilo, da so se morali razvijalci dogovoriti in si rezervirati module, ki so jih želeli spreminjati. Za te zadeve so navadno uporabljali tablo, na katero so napisali svoje ime ob imenu modula, ki so si ga je rezervirali. Prijava (check-in) kode je potekala tako, da so izbrisali svoje ime ob imenu modula, kar je pomenilo, da je lahko modul rezerviral katerikoli izmed razvijalcev.

V poznih 60-ih in zgodnjih 70-ih letih je profesor Leon Pressor iz Univerze Santa Barbara v Kaliforniji

postavil tezo, da je potrebno spremeniti kontrolo konfiguracije. Posledica teze je bil podpis pogodbe z vojaškim dobaviteljem, ki je izdeloval letalske motorje za mornarico. Tudi vojaško letalstvo je želelo kupiti enako orodje za upravljanje konfiguracije.

Tako je bil leta 1975 zgrajen prvi komercialni proizvod, ki se je imenoval "Change and Configuration Control" in ga je prodajala programska hiša SoftTool. Od takrat naprej se na tržišču pojavljajo vedno novejša in boljša orodja, brez katerih si ni mogoče predstavljati upravljanja konfiguracije.

Skoraj istočasno z razvojem programske opreme za UK, so se začeli pojavljati prvi standardi. Tako za prvi standard na področju UK štejemo standard AFSCM 375-1, ki ga je leta 1962 izdal American Air Forces (Leon 2000). S standardom so želeli urediti komunikacijske probleme pri načrtovanju reaktivnih letal. Standardu so sledili drugi standardi večinoma iz ameriške vojske in oddelkov za obrambo.

Med bolj znanimi standardi je sveženj standardov znanih pod imenom niz 480, ki so bili sprejeti leta 1970. Leta 1991 so niz 480 združili v en standard znan kot MIL-STD-973, ki je še vedno v uporabi, vendar je tik pred razveljavitvijo. V začetku leta 1990 je bila dana pobuda za sprejetje nevladnega standarda, kjerkoli je to mogoče. To je bil začetek standarda, ki ga danes širši krogi sprejemajo kot temeljnega na področju upravljanja konfiguracije.

Poznamo ga pod imenom "National Consensus Standard for Configuration Management", ANSI/EIA-649-1998 in ga je izdelal komite G33 za upravljanje konfiguracije podjetja Electronic Industries Alliance. Seveda pa to ni edini standard, ki ga poznamo. Danes najbolj znani in uporabljeni standardi, ki obravnavajo področje upravljanja konfiguracije, so IEEE-828, ISO 9001, ISO 10007, SEI TR-8 in drugi.

Orodja za upravljanje konfiguracije

Pri upravljanju konfiguracije ločimo med dvema, v osnovi popolnoma različnima vrstama inačic - revizija in variacija. Revizija je nova inačica, ki je nadomestila staro. Revizije elementov navadno odražajo napredek v odstranjevanju napak v elementu, lahko pa tudi pomenijo dodajanje nove funkcionalnosti ali izboljšavo v učinkovitosti. Variacije elementa vsebujejo popolnoma enake funkcije, ki pa se različno obnašajo v določenih situacijah. Variacija določenega elementa je torej druga izbira (alternativa). Primer variacije je lahko modul za tiskanje, pri čemer vsaka variacija predstavlja določeno vrsto tiskalnika.

Orodja za UK avtomatizirajo številne procese, kot so nadzor konfiguracije, sledenje napak, spremljanje stanja, kontrola revizij in upravljanje izgradnje proizvoda. Prav te naloge so nujno zlo ročnega upravljanja konfiguracije. Orodja za UK z avtomatizacijo prej naštetih nalog prinašajo več prednosti:

- izboljšanje učinkovitosti razvoja (zaposleni lahko več časa namenijo razvoju, saj se jim ni treba ukvarjati z ročnimi postopki)
- informacijska integracija (vse informacije, ki si jih želijo uporabniki so integrirane in jih lahko kadarkoli prikličemo in prikažemo v želeni obliki)
- prilagodljivost (enostavno lahko izvedemo raznoliko, distribuirano ali paralelno, upravljanje konfiguracije na eni ali več lokacijah)
- boljša analiza in planiranje (uporaba osnovnih funkcija UK in podatkov le-te nam omogoča mnogo različnih zvrst odločitev, funkcij simulacije in analiz kaj če)
- zmanjšanje napak (avtomatizacija monotonih in ponavljajočih nalog, ki so jih prej izvajali zaposleni, odpravlja možnost napak)
- uporaba novejše tehnologije (z uporabo avtomatskih orodij uporabljamo tudi novejšo tehnologijo in pristope vgrajene v ta orodja).

Kljub veliko prednostim, ki jih prinašajo avtomatizirana orodja za UK, se je potrebno zavedati, da takšna orodja niso rešitev vseh problemov s področja UK. Namen teh orodij je le korak k učinkovitem upravljanju. Uporaba napačnih orodij ali nesmiselna uporaba pravih orodij nas lahko pripelje še v večje težave.

Vrste orodij

Na tržišču veliko orodij, ki nam omogočajo upravljanje konfiguracije. Orodja delimo v tri skupine (Leon 2000):

- specialna orodja za UK
- orodja za upravljanje in/ali nadzor sprememb
- orodja v javni lasti.

V prvo skupino spadajo orodja, ki popolnoma pokrivajo vse funkcionalnosti UK in so navadno procesno orientirana. V drugo skupino spadajo orodja, ki ne pokrivajo celotnega spektra UK, temveč le najpomembnejše funkcije upravljanja konfiguracije, kot so upravljanje sprememb, nadzor sprememb, upravljanje izgradnje ...

Orodja so lahko tudi v javni lasti, torej so brezplačna. Uporaba takšnih orodij je priporočljiva le za projekte, ki niso zelo zahtevni. Pri takšnih orodjih ne moremo pričakovati tehnične podpore. Navadno se takšna orodja uporabljajo v akademskih projektih.

Večina današnjih orodij za UK pokriva področje za upravljanje sprememb, kot tudi njihov nadzor. Vsako izmed orodij ima svoje prednosti in slabosti, zato je pri odločitvi potrebno vedeti kaj pričakujemo od takšnega orodja, na primer ali bomo razvijali le na eni platformi ali več, ali bomo izvorno kodo razvijali na eni lokaciji ali na več lokacijah in podobno.

Danes najzmogljivejša orodja za UK omogočajo vse osnovne funkcije UK, kot so določitev, nadzor, spremljanje stanja in presojanje konfiguracije. Takšna orodja so: AideDeCamp (True Software), CCC/Harvest (Platinum Technology), ClearCase (Rational Software Corporation), ClearGuide (Rational Software Corporation), Continuus (Continuus), PVCS Dimensions (MERANT) in drugi (Leon 2000). Orodja se med seboj razlikujejo v notranji arhitekturi in načinu shranjevanja in beleženja sprememb, številu podprtih operacijskih sistemov, možnosti dela na več lokacijah in podobno (Eaton 2001). Orodje ClearCase bo bolj natančno opisano v naslednjem razdelku.

V skupino orodij za upravljanje in nadzor sprememb, ki ne pokrivajo celotnega področja UK, spada večino orodij za UK. Naštejmo le nekatere izmed njih: +1CM, Alchemist, ChangeMaster, CONTROL, MKS Source Integritiy, PVCS Version Manager, Sun WorkShop TemaWare, Visual SourceSafe.

Orodja v javni lasti so dostopna po internetu ali jih dobimo z operacijskim sistemom UNIX. Takšna orodja imajo precej dobro dokumentacijo. Primeri teh orodij so Aegis, DVS, CVS, Revision Control System (RCS), Source Code Control System (SCCS), TkCVS.

Uvajanje upravljanja konfiguracije

Uvajanje novega orodja za upravljanje konfiguracije pomeni ogromno spremembo. Novo orodje uvajamo v organizacijo, ker z obstoječim orodjem nismo zadovoljni, ali ker ne uporabljamo še nobenega orodja in bi ga v bodoče želeli uporabljati.

Uvajanje orodja za UK pomeni pomembno priložnost in veliko odgovornost. Priložnost je vse dokler nam takšno orodje omogoča odkrivanje problemov UK in izvajanje procesnih izboljšav za upravljanje razvoja in vzdrževanja programov. Uvedba novega orodja pomeni tudi veliko odgovornost zaradi razvejanosti in zaradi virov, potrebnih za izvedbo spremembe. Vsaka sprememba v organizaciji je zahtevna, še posebej če se nanaša na UK. UK namreč vpliva na vse združene podatkovne shrambe v organizaciji in na vse njene razvojne, testne, kakovostne in upravljavske procese. Spremembe morajo biti dobro vodene, kar pomeni, da mora biti uvajanje izvedeno tako, da bo uporaba orodja za UK uspešna, učinkovita in optimalna.

Uvajanje orodja za UK v organizacijo je treba skrbno načrtovati. Tipična strategija za prehod na novo orodje za UK je sestavljena iz petih faz:

- priprava in planiranje
- definiranje procesov v organizaciji
- izvedba pilotnega projekta
- širjenje uporabe
- izboljševanje procesov.

Z zgoraj naštetimi koraki začnemo potem, ko smo že ovrednotili vsa orodja med katerimi smo se odločali in smo že izbrali orodje, ki ga bomo v bodoče uporabljali v organizaciji.

Namen faze priprave in planiranja je izvesti nujne priprave za uvajanje orodja. Ta faza je izjemno pomembna in vsako izpuščanje te faze je lahko zelo boleče, dolgotrajno in navadno se pokaže kot neuspešno uvajanje novega orodja.

V tej fazi je potrebno ustanoviti skupino, ki bo zadolžena za uvajanje novega orodja. Skupina je odgovorna za izvedbo strategije uvajanja. Njena naloga je spremljati in sodelovati pri vseh fazah uvajanja orodja.

Skupina za uvajanje najprej izdela načrt uvajanja orodja za UK. Načrt opisuje prednosti orodja za UK, določa urnike in postopke realizacije uvajanja orodja, ustanavlja kriterije uspeha in vlogo skupine za uvajanje orodja. V tej fazi je treba razviti tudi načrt tveganja, to je tveganja, ki povzročijo odstopanja od načrta uvajanja orodja.

Namen definiranja procesov v organizaciji je zbirati in razumeti proces upravljanja konfiguracije, da bi ga lahko kar najbolje avtomatizirali in razumeli vlogo orodja v odnosu do procesa upravljanja konfiguracije. V tej fazi definiramo obstoječi proces in napišemo novega, če je to potrebno, napišemo dokument procesnega modela, ga implementiramo in opredelimo tveganje.

Namen izvedbe pilotnega projekta je testiranje modela procesa in preseljevanja skupnih podatkov v podatkovno shrambo, z namenom preverjanja delovanja orodja, določitvi krivulje učenja za različne uporabnike in za prototipiranje tveganih vprašanj.

Pri izvedbi pilotnega projekta je pomembno, da je v njem zajeto področje velikega tveganja, ki pa ne vpliva na kritično pot projekta. S pilotnim projektom razvija skupina zadolžena za uvajanje standarde, predpise in postopke, hkrati pa se uri za kasnejše delo pri upravljanju konfiguracije. Uspehi in neuspehi pri delu se dokumentirajo ter se primerjajo s kriteriji, določenimi v načrtu uvajanja.

Namen faze širjenja uporabe je postopno uvajanje orodja med posameznike in projektne skupine. Šolanje in zmanjševanje odpora do sprememb je ključna aktivnost te faze. Orodje UK, procesi, postopki in šolanje morajo biti izvedeni in uvedeni za vsako projektno skupino. Skupina, ki uvaja novo orodje za UK izvaja, ovrednoti in spremlja aktivnosti razširjanja uporabe novega orodja. Faza je končana, ko se orodje, procesi in postopki tekoče uporabljajo.

Namen faze izboljševanja procesov je ovrednotenje aktivnosti uvajanja, potrditev delujočih strategij in priporočanje procesnih izboljšav. To nam omogoča, da lahko v organizaciji izboljšamo procese in tako uživamo vse prednosti, ki jih omogoča orodje za upravljanje konfiguracije.

Orodje ClearCase

ClearCase je aplikacija tipa odjemalec-strežnik, ki omogoča kontrolo revizij, upravljanje okolja, kontrolo postopka in upravljanje prevajanja. Na strežniku se nahaja projektna podatkovna baza, v kateri so shranjene verzije izvorne kode. Odjemalci dostopajo do projektne podatkovne baze s pomočjo posebnih vmesnikov in omogočajo izmenjavo izvorne kode. Strežnik je lahko okolje HP-UX ali Windows, odjemalec pa lahko katerokoli okolje Unix ali Windows.

Projektna podatkovna baza se v orodju ClearCase imenuje Version Object Base (VOB). VOB je skladišče elementov, ki so shranjeni v več verzijah, s katerimi upravlja ClearCase. Element je objekt s stisnjenimi verzijami organiziranimi v obliki drevesne strukture. Glede na lastništvo in mesto hrambe elementov, ločimo javne in zasebne VOB-e.

Na strežnik ClearCase lahko namestimo več VOB-ov, da bi lahko elemente združili glede na naše potrebe. Direktno pisanje in branje VOB-a ni mogoče. Branje je mogoče le s posebnim vmesnikom, ki ga namestimo na odjemalcu. Pred spreminjanjem elementov, ki se nahajajo v VOB-u, je potrebno zahtevati odjavo elementa, strežnik zaklene element in na lokalnem disku se shrani kopija določene verzije elementa, ki jo lahko spreminjamo dokler spremembe ponovno ne prijavimo. Ob prijavi se element odklene, na strežniku se kreira nova revizija in na odjemalcu se pobriše kopija.

Ker lahko vsak odjemalec prikazuje elemente različnih VOB-ov, vsak element pa ima več verzij, je potrebno nekako določiti katere VOB-e, katere elemente in katere verzije želimo videti. Z drugimi besedami, potrebno je določiti konfiguracijo, ki jo želimo imeti na odjemalcu.

Konfiguracijo v orodju ClearCase nastavimo s tako imenovanim ConfigSpec-om. ConfigSpec je zaporedje pravil, s katerimi določimo VOB-e, elemente in verzije, ki bodo vidne v zasebnem delovnem prostoru (pogledu) določenega razvijalca. V zasebnem delovnem prostoru prikazujemo verzije imenikov in datotek enega ali več VOB-ov, kot tudi privatne datoteke, ki niso vidne pogledom. Vsak razvijalec ima lahko več pogledov. Ker je ConfigSpec lokalnega pomena, ga je potrebno določiti za vsak delovni prostor posebej.

Privzeti ConfigSpec zasebnega delovnega prostora je:

```
element * CHECKOUTED
element * /main/LATEST
```

Omenjeni ConfigSpec določa, da bo zasebni delovni prostor vseboval elemente, ki jih je razvijalec odjavil ali zadnje inačice elementov, shranjenih na strežniku v glavni veji.

Ločimo dve vrsti pogledov:

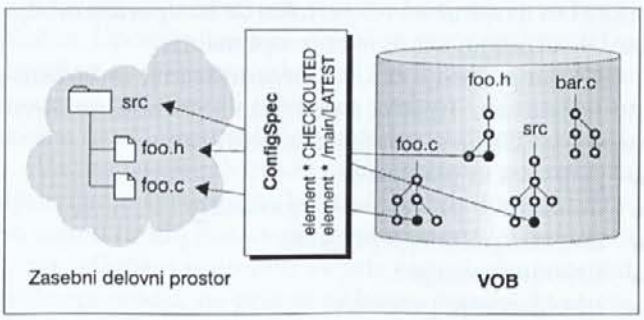
- Dinamični pogled se posodobi dinamično in ne zavzema prostora na disku, saj vidimo datoteke direktno iz VOB-a prek posebnega datotečnega sistema MultiVersion File System (MVFS)
- Posnetek (snapshot) vsebuje kopijo verzije elementov, ki jih shrani na lokalni disk in se ne posodobi dinamično, ampak le na zahtevo uporabnika

Slika 1 prikazuje VOB na katerem so shranjeni štirje elementi. Vsak izmed elementov ima več revizij. ConfigSpec se nahaja na odjemalcu in upošteva pravilo, ki določa, da želimo videti zadnje revizije vsakega elementa. Zato vidimo v zasebnem delovnem prostoru zadnjo verzijo imenika src in datoteki foo.c in foo.h. Datoteke bar.c ne vidimo, saj se ne nahaja v imeniku src. Datoteka je bila verjetno izbrisana v kateri izmed prejšnjih verzij imenika src, ali pa nikoli ni bila element tega imenika.

Verzije elementov so v VOB-u shranjene kot razlike (delte) med novo in staro revizijo. Zraven informacij o sami spremembi elementa shrani ClearCase še dodatne informacije, ki nam omogočajo slediti spremembe. Te dodatne informacije, ki se shranijo v seznam sprememb, so: datum in ura, uporabniško ime, dogodek (nova revizija, označevanje, kreiranje veje), verzija in komentar. Revizije so v orodju ClearCase predstavljene s krogom. Kvadrat predstavlja vejo ali variacijo. Privzeto ima vsak element glavno vejo, po potrebi pa lahko kreiramo nove stranske veje.

Orodje ClearCase omogoča označevanje revizij. Vsako revizijo, ki smo jo uporabili v postopku prevajanja in pakiranja, označimo z zaporedna število prevajanja. Takšno označevanje nam zagotavlja ponovljivost kateregakoli prevajanja, če le nastavimo ConfigSpec na želeno označbo.

Stranske veje lahko uporabljamo za razvoj novih funkcionalnosti in izdelavo popravkov zato, da s tem ne vplivamo na trenutni razvoj. Veja s popravki je primerna tudi zato, ker na enem mestu vidimo vse spremembe kode, ki so posledice odkritih napak,



Slika 1: **Preslikava elementov VOB-a v zasebni delovni prostor**

potem ko smo proizvod že oddali. Te popravke lahko po potrebi enostavno zlijemo tudi v glavno vejo ali katero drugo stransko vejo.

Orodje ClearCase je zelo prilagodljivo. Zraven opisanega nam nudi možnost dodajanja lastnih atributov in funkcij, ki jih vežemo na dogodke v orodju. To so na primer prijava elementa konfiguracije, odjava elementa, kreiranje označbe, kreiranje veje, brisanje elementa ... Vnaprej definirana funkcija se lahko sproži v dveh primerih:

- pred dogodkom
- po dogodku.

Funkcije navadno pišemo v jeziku Perl, saj je le-ta dodan orodju ClearCase in zato tako napisani sprožilci delujejo neglede na operacijski sistem, ki ga uporabljamo na odjemalcu.

Dodatek k orodju ClearCase z imenom Multisite nam omogoča, da izbrane VOB-e uporabljamo hkrati na več različnih lokacijah. Pri tem ima vsaka lokacija svoj strežnik ClearCase na katerem je nameščena kopija (replika) VOB-a. Strežniki na različnih lokacijah se med seboj sinhronizirajo s pošiljanjem nadgradnih paketov. Tak sistem nam omogoča nemoteno delo na različnih lokacijah, kljub slabi mrežni povezavi. Prvi sinhronizacijski paket vsebuje vse informacije izbranega VOB-a, vsi nadaljnji nadgradni paketi pa le razlike, ki so se zgodile v času od zadnje sinhronizacije.

Sinhronizacija se lahko sproži ročno ali avtomatsko. Avtomatska sinhronizacija se navadno izvaja ponoči, saj je mrežni promet takrat majhen. Nočna avtomatska sinhronizacija omogoča, da je zjutraj pred začetkom dela stanje na obeh lokacijah enako.

Uporaba pri projektu OmniBack

Projekt OmniBack II ima v Hermes SoftLab-u precej dolgo tradicijo. Njegovi začetki segajo v leto 1990, kar z drugimi besedami pomeni, da je bil OmniBack II prvi projekt v podjetju Hermes SoftLab. Na začetku so pri omenjenem projektu delali štirje ljudje. Z vsakim izidom programa je na projektu sodelovalo več ljudi, ki so pisali kodo za vse več in več platform. Tako se je projekt v prvih šestih letih povečal iz štirih programerjev na petindvajset (25) ljudi, ki so kodirali kodo za UNIX. Tega leta se je tudi prvič pojavila potreba po združitvi skupine, ki je razvijala kodo za sisteme UNIX z novo ustanovljeno skupino, ki je razvijala enak produkt za sisteme Windows in se je ustanovila sredi leta 1995. Do združitve obeh skupin je prišlo leta 1997. Sedaj šteje skupina OmniBack II 90 ljudi in ima za seboj 19 oddanih verzij programa.

Projekt OmniBack II razvija istoimenski produkt za naročnika Hewlett Packard. Produkt spada v skupino HP OpenView produktov in omogoča izdelavo varnostnih kopij podatkov in njihovo obnavljanje. Produkt je posebej prilagojen za globalne, široko-podjetne in distribuirane sisteme. Trenutno podpira približno 19 okolij (Windows NT i386, Windows DEC-Alpha, Novell, HP-UX 10.x, HP-UX 11.x, Silicon Graphics IRIX, IBM AIX, Sun Solaris, Sun OS, Alpha, Linux, SCO, NCR, Dynix, Sinix, ...), za katere ima lastne agente za delo z diskom in mediji, in se zna integrirati z večino najbolj znanih integracij (Sybase, Oracle, EMC, OPC, SAP, Informix, Symmetrix, Fastrax ...). V svetu si trenutno deli drugo mesto glede na število uporabnikov in se še vedno razvija.

Prehod na ClearCase

V začetni fazi smo pri projektu OmniBack II za Unix uporabljali program RCS, medtem ko smo pri sistemih Windows uporabljali program Source Integrity. Obe bazi sta bili ločeni. Vsaka izmed baz je zajemala približno štiri tisoč (4000) datotek izvorne kode, razvrščenih v približno dvesto petdeset (250) imenikov. Težava je bila, da smo na obeh sistemih (UNIX in Windows) razvijali isti produkt, kar pomeni, da je bila koda zelo podobna, nikakor pa ne enaka. Dnevno je bilo zato treba ročno sinhronizirati kodo med obema sistemoma, kar je bilo zelo zamudno in težavno opravilo. Vendar pa to ni bila edina pomanjkljivost prejšnjih orodij. Obe orodji sta bili zelo slabo zaščiteni pred zlorabami. Zato smo začeli kmalu razmišljati o orodju, ki bi rešilo te težave. Ob predhodnem pogovoru z ekspertom smo izbrali orodje ClearCase.

Prehod na ClearCase je potekal v več korakih:

- uskladitev kode s sistema Unix s sistemom Windows (imena datotek)
- uvoz kode v ClearCase (cca. 4000 datotek s povprečno 100 revizijami)
- prilagoditev starega okolja na novo orodje (sprememba poti, načina dela, ...)
- priprava tečaja in dokumentacije za razvijalce
- priprava skript za lažje delo (skripte za množično prijavo elementov v konfiguracijo).

Časovno najzahtevnejša koraka sta bila prva dva. Na srečo smo z uskladitvijo pričeli, še preden smo resno razmišljali o prehodu na novo orodje. Res pa je tudi, da smo se zadnji mesec pred prehodom ukvarjali samo z uskladitvijo.

Ker je bilo potrebno uvoziti okrog štiri tisoč datotek, nekatere med njimi pa so imele več kot 200 revizij, je to pomenilo, da bo celoten uvoz trajal 96 ur. Ker v tem času ni dovoljeno vpisovanje v staro bazo,

nova pa v tem času ni uporabna, bi to pomenilo, da razvijalci ne bi mogli nič delati. Zato smo se odločili za naslednji postopek:

1. Označimo celotno razvojno drevo (UNIX & NT) z označbo CC1, kar bo pomenilo prvo fazo uvoza v ClearCase.
2. Napravimo varnostno kopijo celotne baze (UNIX & NT) in jo obnovimo na strežnik ClearCase, da tako omogočimo razvijalcem nemoteno nadaljevanje dela.
3. Poženemo uvoz iz UNIX-ove varnostne kopije baze kar traja 96 ur, takoj po tem pa še uvoz iz NT-jeve varnostne kopije baze. Pri tem je treba paziti, da uvozimo le NT-specifične datoteke. Med tem časom, ko teče uvoz pa pripravimo tečaje in dokumentacijo za delo z orodjem.
4. Pred končno drugo fazo uvoza preverimo, ali so vse datoteke prijavljene.
5. Ponovno označimo celotno drevo (UNIX & NT) z označbo CC2.
6. Napravimo varnostno kopijo stare baze (UNIX & NT) in jo obnovimo na strežnik ClearCase. Staro bazo umaknemo iz uporabe, novo pa zaščitimo pred uporabo, dokler vsa koda ni uvožena. To pomeni, da kakšen dan ali dva nihče ne bo nič razvijal. Ta čas uporabimo za tečaj uporabe ClearCase.
7. Ponovno poženemo uvoz kode (UNIX & NT), vendar tokrat le morebitnih novih verzij, ki so bile prijavljene po označbi CC1, kar bi po naši oceni trajalo manj kot 24 ur.
8. S skriptami preverimo, če so bili uspešno vključeni vsi imeniki z vsemi datotekami in če so zadnje verzije kode znotraj ClearCase enake verzijam znotraj RCS.

Predstavitev pravil

Vsaka sprememba orodja zahteva precej dela in prilagajanja konfiguracije novemu okolju. V času zamenjave orodja lahko veliko stvari, ki so bile prej slabo realizirane ali definirane, izboljšamo in jih bolj natančno definiramo. Tukaj imamo v mislih predvsem poimenovanje VOB-ov, pogledov, datotek ... Ugotovili smo namreč, da se lahko z dobrim in jasnim poimenovanjem izognemo veliko težavam pri avtomatizaciji procesa, pri procesu usposabljanja novo zaposlenih in administraciji orodja.

VOB-i

Čeprav pojma VOB v prejšnjih orodjih nismo poznali, smo imeli celotno kodo kljub vsemu razdeljeno na štiri sestavne dele. Tem delom smo dodali nove in tako smo celotno kodo razbili na 12 VOB-ov ali delov. Vsak del tvori logično celoto in se večinoma lahko obravnava ločeno od ostalih delov. Ti logični deli so:

Ime VOB-a	Opis VOB-a
ADM	administracijski VOB - najvišji v hierarhični strukturi
BIN	datoteke, ki jih povezujemo v končni produkt in jih nismo napisali sami
BUILDS	končni paketi in produkti
CFG	orodja in skripte za prevajanje, povezovanje in pakiranje
DOC	projektna in interna dokumentacija
MANUALS	dokumentacije o produktu
PLAY	prostor za učenje, igranje in testiranje
SRC	izvorna koda produkta
SUPPORT	orodja, dokumentacija in skripte za skupino, ki skrbi za podporo strank
TEST	orodja in skripte za testno skupino
TOOL	orodja in skripte za administracijo celotnega okolja prevajanja
WEB	datoteke, orodja in skripte za WEB

Pogledi in datoteke

Ker se vsak pogled registrira na strežniku ClearCase, je potrebno poglede poimenovati, da vsakdo lahko določi kadarkoli, kdo je lastnik določenega pogleda in na katerem računalniku se ta pogled nahaja. Takšna pravila se izkažejo za nujna v primerih, ko razvijalec zapusti skupino in za seboj ne počisti delovnega okolja.

Tako se pri sistemih Windows zahteva, da poimenujemo pogled v obliki:

<uporabniško_ime>_<niz>_view

pri sistemih Unix pa ga poimenujemo v obliki:

<uporabniško_ime>_<niz>

pri čimer je niz poljubna kombinacija številk in črk, ki nam pomaga razlikovati poglede v primerih, ko ima en razvijalec več pogledov.

Ob kreiranju novih datotek v pogledih se zahtevajo imena brez presledkov. Pri sistemih Unix smo uvedli še dodatno zahtevo, ki prepoveduje poimenovanje datotek, ki bi se med seboj razlikovale le v velikih in malih črkah. Takšnih datotek ni mogoče pregledovati pri sistemih Windows, saj jih le-ti med seboj ne ločijo.

Novo datoteko (element) lahko v VOB doda kdorkoli, ki je član projektne skupine in je pred tem napravil osnovne teste s katerimi je preveril, da z dodajanjem ne ogrozi delovanja produkta. Dodana datoteka mora biti izvorna koda. Izvorna koda je datoteka, ki je ni mogoče kreirati iz nobene druge datoteke ali skupine datotek, ki so že shranjene v

VOB-u. Lokacija hrambe se določi v dogovoru z upravljavcem konfiguracije, katerega naloga je tudi, da nadzira upoštevanje pravila o dodajanju novih datotek v VOB.

Označbe, veje in zaklepanje

Spremenili smo tudi pravila označevanja in vejenja:

- z velikimi črkami deklariramo označbe.
- z malimi črkami deklariramo veje.

Za potrebe upravljanja konfiguracije se oboje kreira globalno za vse VOB-e, za razvijalce pa so globalne definicije prepovedane.

Za vsako uradno prevajanje celotne kode se avtomatsko kreira unikatn1a označba, ki nam omogoči ponovno izgraditi produkt. Ta označba ima format:

A.<verzija>_<zaporedna_številka>

Primer take označbe je A.03.50_103. Prvi del je izposojen iz HP-notacije za označevanje programske opreme.

Po oddaji se vsa oddana koda označi z označbo v formatu:

R<verzija>_MR

Istočasno se kreira tudi veja, na kateri se bodo izdelovali popravki (na glavni veji bomo razvijali novo verzijo produkta). Veja se poimenuje:

r<verzija>_fix

Označba, ki označuje oddano kodo, se zaklene, da je ne bi kdo po pomoti premaknil ali izbrisal. *Fix* veja se zaklene za vse razvijalce razen za tiste, ki skrbijo za podporo strank. Le-ti jo po potrebi odklenejo tudi za ostale razvijalce, v kolikor stranke potrebujejo popravke.

ConfigSpec

Configspec-e shranjujemo v ADM VOB-u, da ne prihaja do napak pri prevajanju in prijavljanju kode. Vsak ConfigSpec mora biti shranjen v dveh oblikah - prvi za Windows (končnica *.nt) in drugi za Unix (končnica *.ux).

ConfigSpec poimenujemo v naslednjem formatu:

Ime ConfigSpec-a	Opis
build_r<verzija>	ConfigSpec za uradno prevajanje
dev_r<verzija>	ConfigSepc za razvoj
support_r<verzija>	ConfigSpec za izdelavo popravkov
proto_<funkcionalnost>	ConfigSpec za razvoj novih funkcionalnosti, ki se bodo oddajale ločeno od produkta

Sprožilci

Ker zaradi zgodovinskih razlogov za prevajanje ne uporabljamo orodja *clearmake*, ki nam avtomatsko shranjuje informacije o tem, katera izvorna koda je bila uporabljena, smo morali sami izdelati skripto. Ta nam ob prijavi elementa avtomatsko pregleda izvorno kodo in zamenja niz znakov "$Header" z novim nizom, ki vsebuje podatek o datumu in uri prijave, razvijalcu, ki je spremembo naredil in zaporedni številki spremembe in imenu datoteke. Ti podatki so pomembni, saj jih ob povezovanju vpišemo tudi v končno izvršljivo datoteko. Z njihovo pomočjo lahko kadarkoli ugotovimo, kakšen program ima stranka, kdaj je bil preveden in katere izvorne datoteke so bile uporabljene. Te informacije pridejo še posebej prav skupini za podporo strankam, ki izdeluje popravke.

Multisite

Vse naše VOB-e sinhroniziramo z vsaj eno lokacijo - največkrat z našimi poslovnimi partnerji v Nemčiji in poslovno enoto v Sarajevu. Sinhronizacija poteka navadno avtomatsko s pomočjo najete linije, redkeje pa ročno po elektronski pošti. Prvi sinhronizacijski paket se vedno pošlje posnet na traku (DAT), saj so podatki reda velikosti 500 Mb. Sinhronizacija se vedno opravlja ponoči, da ne moti razvojnega procesa, saj se baza zaklene, da so podatki konsistentni.

Zaključek

Orodje ClearCase je zmogljivo orodje, brez katerega si pri projektu OmniBack enostavno več ne znamo predstavljati dela. Njegove največje prednosti glede na stari način dela so, da omogoča delo v mešanem okolju (UNIX in Windows), zaradi česar ne potrebujemo dveh ločenih projektnih podatkovnih baz z izvorno kodo in kode ni potrebno ročno usklajevati med njima. Naslednja ogromna pridobitev je tudi *Multisite*, s čimer smo se izognili ročnemu pošiljanju kode našim poslovnim partnerjem in oddaljenim enotam. Prej je bilo potrebno v vsaki beta fazi zapakirati izvorno kodo in končni produkt in ga poslati na strežnik FTP. Sedaj to več ni potrebno, saj se koda vsako noč sinhronizira, odpade pa tudi skrb, ali je bilo vse preneseno, saj za to skrbi samo orodje. Če torej želimo, da se končni produkt pošlje poslovnemu partnerju, ga je potrebno le dodati pod kontrolo ClearCase, vse drugo pa se zgodi avtomatsko. Obstaja še nekaj manj pomembnih razlogov za uporabo orodja, kot so centralno upravljanje, integriranje z velikim številom razvojnih orodij in poizvedovanja po konfiguraciji, ki nam omogočajo dober pogled nad spremembami kode.

Seveda pa ima vsaka medalja dve plati in zato ima tudi ClearCase svoje pomanjkljivosti. Največja med

njimi je rahla nestabilnost odjemalcev ClearCase in orodja za dostop do mrežnega podatkovnega sistema (NFS) na sistemih Windows. V takšnih primerih je neobhodno potreben ponoven zagon operacijskega sistema. Naslednja, mogoče manj pomembna slabost, je visoka cena in zahtevnost glede strojne opreme (pomnilnik, omrežna kartica).

Odgovor na vprašanje ali uporabljati orodje za upravljanje konfiguracije, je očiten. Odločitev ali boste izbrali orodje ClearCase ali katero drugo, pa je prepuščena vam. Izbrati je namreč treba orodje, ki bo zadostilo vašim zahtevam!

Literatura

1. Configuration Management Information Center (2001), What is Configuration Management, http://www.pdmic-com-cmic-introtoCM.shtml
2. Configuration Management FAQ (2001), http://www.iac.honezwell.com pub/Tech/CM/CMFAQ.html, junij 2001
3. Leon, A. (2000): A Guide to Software Configuration Management, Artech House Publishers
4. IEEE/ANSI 828-1990 Standard for Software Configuration Management Plans, Institute of Electrical and Electronics Engineers, 1990
5. ISO 10007:1995 Quality management - Guidelines for configuration management, International Organization for Standardization, Geneva, Switzerland, 1995
6. ISO 9001:2000 Quality management systems - Requirements, International Organization for Standardization, Geneva, Switzerland, 2000
7. Dart A. Susan (1993): CMU/SEI-93-TR-8, A Study in Software Maintenance, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr08.93.pdf
8. Eaton, D. (2001): Configuration Management Tools Summary, verzija 8.3b, http://www.daveeaton.com/scm/CMTools.html, junij 2001
9. Rational University (1998a), ClearCase Fundamentals for Windows NT version 3.2, Student Guide, Rational Software Corporation
10. Rational University (1998b), ClearCase Meta-Data for UNIX version 3.0, Training Manual, Rational Software Corporation

◆

Uroš Sajko je diplomiral na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru. Sedaj je študent podiplomskega magistrskega programa računalništva in informatike na tej fakulteti. Zaposlen je v podjetju Hermes SoftLab d.d. v Ljubljani, kjer je uvedel orodje Rational ClearCase, s katerim že nekaj let uspešno vodi upravljanje konfiguracije pri projektu OmniBack.

◆