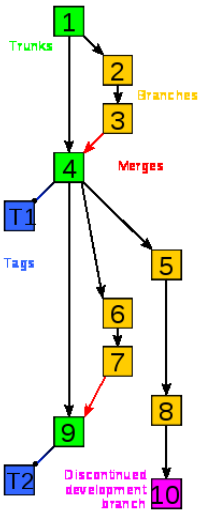
Wprowadzenie do systemów kontroli wersji

Witold Paluszyński
Katedra Cybernetyki i Robotyki
Politechnika Wrocławska
http://www.kcir.pwr.edu.pl/~witold/

Pojęcia występujące w systemach kontroli wersji



Delta — zbiór zmian pewnej wersji modułu

Branch — rozwidlająca się gałąź wersji

Trunk — gałąź bez rodzica, często nowa wersja

Tag — nazwa nadana pewnej wersji

Fork

Commit

Merge

Repository

SCCS: delty

Protoplastą wszystkich systemów kontroli wersji jest SCCS (*Source Code Control System*) wymyślony przez Marka Rochkinda w 1972 roku i początkowo zaimplementowany w języku Snobol4 w systemie IBM/370. Uniksowa implementacja napisana w C została opisana w publikacji w IEEE Trans. on Software Engineering Vol. 1 No. 4 w roku 1975.

Idea na której opiera się ten system wykorzystuje fakt, że większość zmian w kodach źródłowych jest mała w porównaniu z całym kodem, zatem każdą zmianę można pamiętać w postaci tzw. **delty**. Pamiętanie delt kolejnych rewizji pozwala zarówno otrzymać bieżącą wersję kodu (pliku), jak i każdej wersji pośredniej. Teoretycznie więc system nie pamięta ostatniej wersji pliku tylko odtwarza ją z łańcucha delt.

Jednak pojęcie delt stanowiło jedynie abstrakcyjną perspektywę.
W rzeczywistości implementacja SCCS wykorzystywała tylko dwie operacje: dodawania i usuwania całych wierszy.

SCCS: wydania i poziomy

W SCCS wprowadzono pojęcie **wydania** (*release*). Kolejne wersje w ramach wydania 1 nazywały się **poziomami** (*levels*): 1.1, 1.2, 1.3, itp. Wraz z wprowadzeniem kolejnej delty programista mógł zadeklarować nowe wydanie, i wtedy kolejne wersje nazywały się: 2.1, 2.2, itp.

Po zainicjowaniu kolejnego wydania, na przykład 2.2 po ostatnim 1.4 można kontynuować modyfikacje głównej wersji modułu źródłowego (2.2→2.3), jak również wprowadzać poprawki do wydania 1 (1.4→1.5).

SCCS umożliwiał również wprowadzanie delt warunkowych, które były uwzględniane lub nie przy generacji żądanej wersji.

SCCS: identyfikacja i dokumentacja

SCCS zawierał prosty schemat identyfikacji wersji polegający na rozpoznawaniu słów kluczowych i zastępowaniu ich odpowiednimi wartościami identyfikującymi wersję modułu źródłowego. Programista musiał sam zadbać, aby taki ciąg znaków znalazł się w module binarnym, aby umożliwić jego identyfikację:

słowo kluczowe	znaczenie
%R%	numer wydania
%L%	numer poziomu
%I%	numer wersji (%R%.%L%)
%D%	data YY/MM/DD (bez godziny)

W praktyce stosuje się magiczny string o nazwie `sccsid` rozpoznawany w modułach binarnych po sekwencji znaków `@(#)` np. przez program `what`:

```
static char sccsid[] = "@(#)mod.c %I %D";
```

SCCS automatycznie rejestrował datę (i czas) oraz programistę tworzącego deltę, oraz powód jej wprowadzenia (wymagany).

SCCS: tryby dostępu

```
# pobierz ostatnia wersje modulu modx do pliku modx.a
get modx

# pobierz ostatnia wersje wydania 1
get modx -r1

# pobierz konkretne wydanie i poziom z uwzglednieniem opcji X
get modx -r3.1 -oX

# pobierz konkretna wersje maksymalnie do lutego 1975
get modx -r3.4 -c7502

# pobierz ostatni poziom wydania 2 i zablokuj do edycji
get modx -r2 -e

# utworz delte porownujac zablokowana i biezaca wersje modulu
delta modx
```


SCCS: aspekty praktyczne

- SCCS jest starym systemem ze sporą grupą hardcorowych zwolenników, którzy nie pozwalają mu zginąć i gotowi są walczyć o jego dobre imię (głównie z jego następcą/konkurentem RCS).
- SCCS przetrwał w postaci kilku nowych i utrzymywanych na bieżąco wersji: wersji komercyjnej na platformach Unix, wersji opensource, wersji Gnu (CSSC). Format pliku historii SCCS pozostał niezmieniony od v4 z 1977r.
- SCCSv4 jest standardem POSIX systemów Uniksowych, cokolwiek to znaczy.
- Pliki kontrolne SCCS wykorzystują przedrostki (np. główny plik historii modułu `mod.c` nazywa się `s.mod.c` co nie współpracuje dobrze z narzędziami typu `make`. Stempel czasowy w pliku `p.` wykorzystuje czas lokalny bez strefy czasowej (mimo to jest elementem standardu POSIX).
- SCCS jawnie pamięta najwcześniejszą wersję modułu, i kolejne wersje zawsze generuje, co jest naturalnie niezbyt efektywne. Jednak ogólnie jest efektywny.

- SCCS oblicza i pamięta sumy kontrolne w plikach co z jednej strony zapobiega korupcji, ale z drugiej utrudnia manipulację na plikach zwykłymi narzędziami.
- SCCS domyślnie utrzymuje pliki historii w podkatalogu SCCS bieżącego katalogu projektu (ten podkatalog sam domyślnie tworzy w razie potrzeby).

RCS

RCS (*Revision Control System*) jest o 10 lat młodszym następcą SCCS (powstał w 1982). Miał być równoważnym zamiennikiem, z prostszym interfejsem. Funkcje RCS:

- pamiętanie pełnego zestawu archiwalnych wersji danego pliku z automatyczną numeracją wersji, dokumentacją i rejestracją czasu utworzenia i autora; dostęp do dowolnej wersji poprzez: numer, datę, autora, bądź nazwę,
- obsługa dodatkowych (bocznych) odgałęzień wersji pliku (*branches*),
- koordynacja pracy pomiędzy wieloma programistami pracującymi nad projektem poprzez ustawianie praw dostępu oraz mechanizm wypożyczania i zwracania plików,
- ograniczona możliwość automatycznego łączenia dwóch różnych zmodyfikowanych wersji jednego pliku; powstanie takich wersji może być wynikiem nieporozumienia programistów, bądź chęci przeniesienia na jedną gałąź rozwojową pliku poprawek dokonanych na innej gałęzi
- korzystanie z automatycznie aktualizowanych znaczników w plikach.

RCS: polecenia

- `ci` przekazuje plik systemowi RCS, nadaje mu numer wersji (kolejny) oraz pobiera i prosi o podanie opisu dokonanej modyfikacji; oryginalny plik zostaje skasowany (z wyjątkiem `ci -u` lub `ci -l`, to ostatnie wywołanie od razu blokuje plik do edycji)
- `co` pobiera plik z RCS; z opcją -l również blokuje plik tak, że wypożyczoną kopię można edytować
- `rlog` wyświetla kompletną historię pliku włącznie z opisami istniejących wersji i aktualnym stanem wypożyczenia pliku
- `rcsdiff` wyświetla różnice pomiędzy bieżącym plikiem a jego ostatnią wersją przechowywaną przez RCS
- `rcsmerge` służy do połączenia dwóch zmodyfikowanych wersji jednego pliku
- `rcs` – komenda administracyjna, służy do modyfikacji ustawień

RCS: przykład

```
shasta-201> ci upr_shell.txt
upr_shell.txt,v  <--  upr_shell.txt
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Prezentacja na temat Bourne shella i filtrow tekstowych
>> .
initial revision: 1.1
done
shasta-202> ls -l upr_shell.txt
upr_shell.txt: No such file or directory
shasta-203> co -l upr_shell.txt
upr_shell.txt,v  -->  upr_shell.txt
revision 1.1 (locked)
done
shasta-206> ls -l upr_shell*
-rw-r--r--   1 witold   gurus        180 Nov  5 11:58 upr_shell.aux
-rw-r--r--   1 witold   gurus      41556 Nov  5 10:59 upr_shell.dvi
-rw-r--r--   1 witold   gurus       9681 Nov  5 11:58 upr_shell.log
-rw-r--r--   2 witold   gurus     129537 Nov  5 11:58 upr_shell.pdf
-rw-r--r--   2 witold   gurus     215917 Nov  5 10:59 upr_shell.ps
-rw-r--r--   1 witold   gurus     243737 Nov  5 10:59 upr_shell.ps2
```



```
-rw-r--r--   1 witold   gurus      30300 Nov  5 10:59 upr_shell.tex
-rw-r--r--   1 witold   gurus      30300 Nov 12 15:17 upr_shell.txt
-r--r--r--   1 witold   gurus      30555 Nov 12 15:17 upr_shell.txt,v
shasta-207> rlog upr_shell.txt

RCS file: upr_shell.txt,v
Working file: upr_shell.txt
head: 1.1
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 1;     selected revisions: 1
description:
Prezentacja na temat Bourne shella i filtrow tekstowych
----------------------------
revision 1.1    locked by: witold;
date: 2003/11/12 14:17:50;  author: witold;  state: Exp;
Initial revision
=============================================================================
```


RCS: uwagi

Często użycie RCS polega na utworzeniu podkatalogu o nazwie RCS, w której polecenia RCS umieszczają swoje archiwa, co pozwala np. na łatwą archiwizację.

Często również każde wprowadzenie pliku do archiwum od razu związane jest z zatrzymaniem jego oryginalnej wersji źródłowej do dalszej edycji przez autora: `ci -l upr_shell.txt`

W RCS nie ma mechanizmu tworzenia jednego spójnego wydania wszystkich modułów źródłowych. Można to obejść tworząc wersje nazwane wszystkich modułów.

Paradoksalnie, koncepcja pamiętania ostatniej wersji modułu i odtwarzania wersji wcześniejszych umożliwia łatwiejszą korupcję pliku historii niż w SCCS, zwłaszcza przy braku sum kontrolnych. Różne wersje programu `diff` stosowanego do tworzenia wersji historycznych prowadzą do możliwości tworzenia historii, której nie da się spójnie przetworzyć.

Podsumowanie: lokalne systemy kontroli wersji

SCCS i RCS reprezentują grupę systemów kontroli wersji opartych na zestawie plików kontrolnych. Administracja takim systemem jest bardzo prosta i sprowadza się do tworzenia dowolnych modułów źródłowych. W razie potrzeby pliki kontrolne mogą być manipulowane przez programistę/ów. Zarządzanie prawami dostępu odbywa się przez mechanizm grup lub list ACL.

System nadzoruje współpracę programistów nad modułem, ale nie wymusza żadnego konkretnego reżimu pracy. Moduł zablokowany przez jednego programistę do edycji, i nigdy nie zwrócony, może być administracyjnie odblokowany przez innego. System automatycznie rejestruje tworzenie nowych wersji, daty i czasy, oraz autorów.

W wielu zastosowaniach ma to szereg zalet, zwłaszcza w małych, półformalnych grupach. Jednak w większych projektach administrator chce mieć nadzór nad sekwencją operacji na plikach. Do takich celów powstały systemy kontroli wersji typu klient-serwer.

CVS

CVS (*Concurrent Versioning System*) jest systemem kontroli wersji typu klient-serwer. CVS powstał w 1986, pierwotnie zaimplementowany jako zbiór skryptów wewnętrznie wykorzystujący RCS.

`http://www.bic.mni.mcgill.ca/~stever/cvs-tutorial/slides.pdf`

`http://duch.mimuw.edu.pl/~janusz/dydaktyka/2004-2005/info_zpp/prezentacje/cvs/index.html`

Subversion (SVN)

`http://www.tutorialspoint.com/svn/`

Git

`https://try.github.io/`

Materiały źródłowe

Mark Rochkind, The Source Code Control System, IEEE Transactions on Software Engineering Vol. 1 No. 4, 1975, dostępny jako:
`http://www.basepath.com/aup/talks/SCCS-Slideshow.pdf`