

Techniki komputerowe w robotyce

WYKŁAD III

Praca grupowa nad projektem informatycznym

Robert Muszyński
KCiR, W12, PWr

Cykl życia oprogramowania

Cykl życia oprogramowania to ciąg działań projektowo-programowych, obejmujący zakres od powstania zapotrzebowania na oprogramowanie aż do jego wycofania z eksploatacji, który jest zbieżny z cyklem projektu i obejmuje:

- fazę strategiczną,
- fazę specyfikacji i analizy wymagań,
- fazę projektowania,
- fazę konstrukcji — implementacji,
- fazę testowania, walidacji,
- fazę konserwacji.

Cykl życia oprogramowania

W ramach cyklu życia oprogramowania powstaje:

- wersja niestabilna (testowa) oprogramowania — seria wydań, podczas której dodawane są przede wszystkim nowe możliwości:
 - wersja robocza (pre-alpha), najczęściej dostępna tylko dla twórców programu w postaci repozytorium kodu źródłowego,
 - wersja alfa (pre-beta), czyli wersja doprowadzona do działania,
 - wersja beta, która ma pierwszych użytkowników,
 - RC (ang. Release Candidate) — wydanie kandydujące, których może być nawet kilka,
 - RTM (ang. Release to manufacture lub Ready to market) — produkt gotowy do wypuszczenia na rynek,
- wersja stabilna oprogramowania — wersja nadająca się do użytkowania zgodnie z założeniami,
- wersje stabilne z poprawkami bezpieczeństwa lub innych błędów,
- starzenie moralne programu.

Numeracja wersji oprogramowania

Numeracja wersji oprogramowania określa kolejność powstawania nowych wersji oprogramowania. Zazwyczaj jest liczbą naturalną lub zestawieniem kilku liczb naturalnych. Wtedy mają one znaczenie:

- numeru głównego (ang. major) — wspólne oznaczenie wszystkich wersji programu bazujących na tych samych założeniach, mechanizmach itd.,
- numeru dodatkowego (ang. minor) — który oznacza kolejne etapy rozwoju programu w ramach tej samej koncepcji, o rozbudowanej funkcjonalności,
- numeru wydania (ang. release) — mówiącym o tym, którym wydaniem w ramach wersji minor jest dana paczka programu,
- numeru kompilacji (ang. build) — numer kompilacji programu, pozwalający na śledzenie kompilacji np. przy rozpatrywaniu zgłaszanych błędów.

Przykłady

- linux kernel 2.6.28
- linux suse 11.3-Milestone3
- gcc version 4.3.2 [gcc-4_3-branch revision 141291]
- svn, wersja 1.5.7 (r36142)

Proces wytwarzania oprogramowania

- Jego organizacją zajmuje się **inżynieria oprogramowania** (koncentrując się na stronie praktycznej) w aspekcie technicznym, organizacyjnym, finansowym itp.
- Celem jest zapewnienie by oprogramowanie było „dobre”, czyli:
 - poprawne, zgodne z wymaganiami użytkowników,
 - łatwe w konserwacji, dokonywaniu zmian,
 - niezawodne (availability, reliability) i bezpieczne (safety, security),
 - przenośne,
 - wydajne i efektywne,
 - ergonomiczne.
- Wymaga dostarczenia mechanizmów zapewniających:
 - współdzielenie kodu,
 - przepływ informacji,
 - obieg dokumentów,
 - rozstrzyganie konfliktów.

Systemy kontroli wersji — funkcjonalności

- Kontrola wersji
- Współdzielenie kodu
- Monitorowanie pracy
- Archiwizacja pracy
- Tworzenie wydań
- Tworzenie gałęzi rozwojowych
- Rozstrzyganie konfliktów

Przy pracy w pojedynkę:

- ułatwiają pracę nad projektem na kilku komputerach,
- umożliwiają cofnięcie każdej zmiany, oznaczanie wydań, tworzenie gałęzi.

Modele życiowe oprogramowania

Można wyróżnić szereg modeli życiowych oprogramowania, w tym:

- Pisz i poprawiaj,
- Model kaskadowy,
- Model spiralny,
- Model prototypowy,
- Model przyrostowy (iteracyjny),
- Model równoległy,
- Programowanie odkrywcze,
- Programowanie zwinne (ang. agile programming),
- Programowanie ekstremalne (ang. extreme programming),
- Programowanie sterowane testami,
- Synchronizuj i stabilizuj.

Narzędzia CASE

Narzędzia CASE (ang. Computer-Aided Software Engineering) to oprogramowanie używane do komputerowego wspomagania tworzenia oprogramowania (na etapie analizy, projektowania i programowania).

Typowe narzędzia CASE:

- narzędzia do modelowania w języku UML i podobnych,
- narzędzia do zarządzania konfiguracją zawierające system kontroli wersji,
- narzędzia do refactoringu,

pozwalają na

- ułatwienie prawidłowego określenia struktury aplikacji,
- ograniczenie liczby nieporozumień pomiędzy projektantem a programistą,
- poprawę komunikacji w zespole,
- wymuszenie dokumentowania procesu tworzenia oprogramowania.

Systemy kontroli wersji

System kontroli wersji (VCS) służy do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu i modyfikacji zmian dokonanych przez wiele osób w różnych momentach.

- **Sposób rozpowszechniania:**

- Wolnodostępne — wszystkie niezamknięte pod Architektura
- Zamknięte — BitKeeper (BitMover), Code Co-op (Reliable Software), Rational ClearCase (IBM), StarTeam (Borland), Visual SourceSafe, Visual Studio Team Foundation Server (Microsoft)

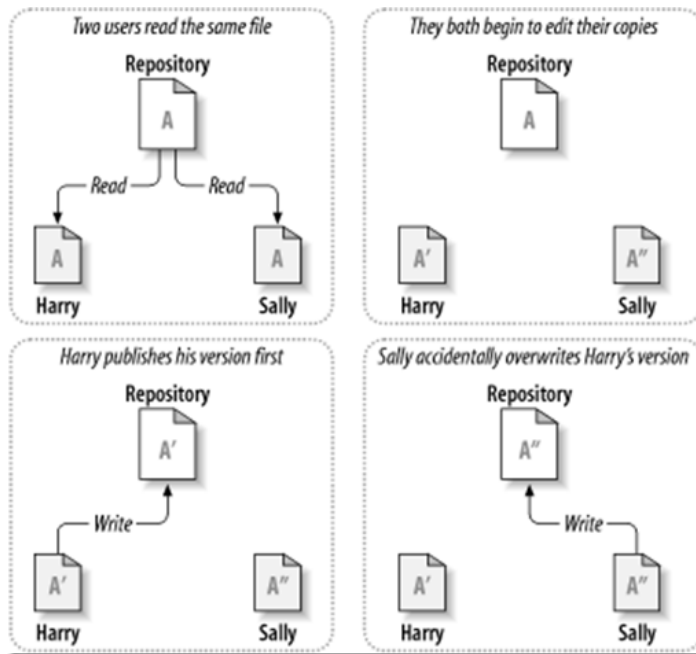
- **Architektura:**

- Scentralizowana — RCS, CVS, Subversion (SVN), GNU CSSC, Perforce
- Rozproszona — Bazaar, Codeville, Darcs, GNU Arch, Git, BitKeeper, Code Co-op, Mercurial, Monotone, svk

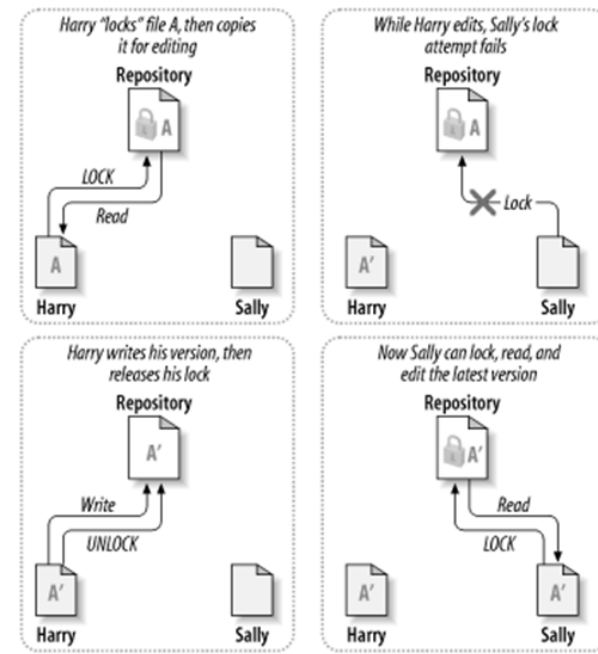
- **Modele pracy:**

- blokuj-modyfikuj-odblokuj (ang. lock-modify-unlock)
- kopiuj-modyfikuj-scal (ang. copy-modify-merge)
- hierarchiczne

System kontroli wersji — modele pracy

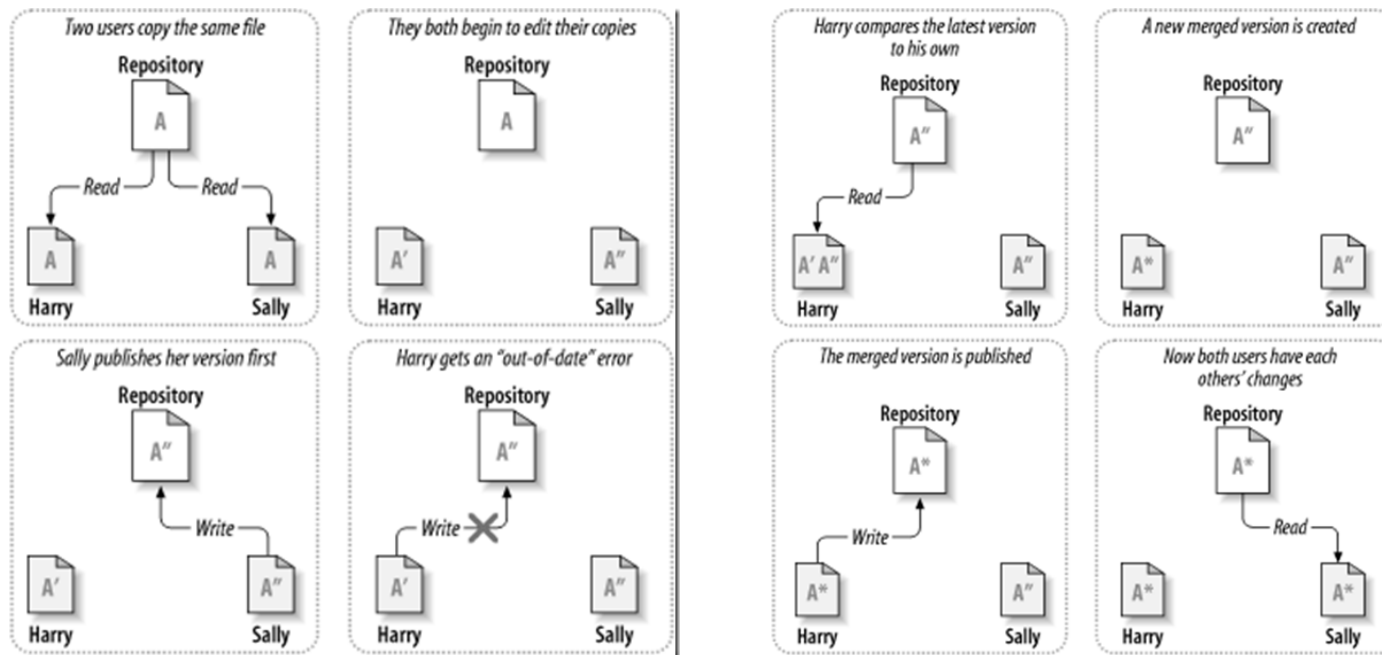


Typowy problem



Blokuj-modyfikuj-odblokuj

System kontroli wersji — modele pracy



Kopiuuj-modyfikuj-scal

System kontroli wersji Subversion

Subversion jest darmowym systemem kontroli wersji o otwartym źródle. Służy on do zarządzania plikami, katalogami i zmianami wprowadzanymi w miarę upływu czasu. To pozwala na odzyskanie starszej wersji danych lub przeglądnięcie historii zmian na plikach. Subversion może działać przez sieć, która umożliwia wykorzystanie go przez różne platformy.

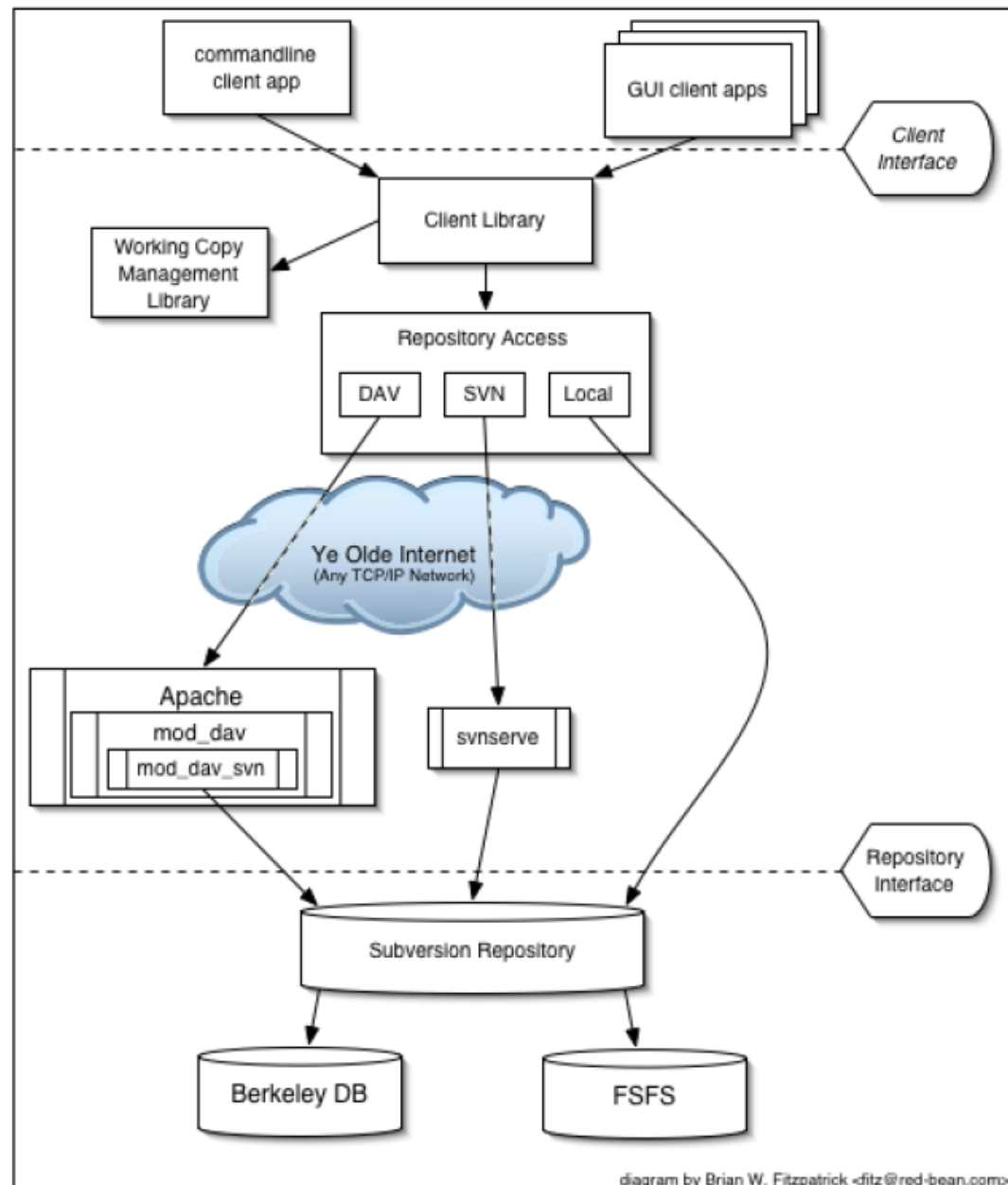
Dostarczana funkcjonalność

- współdzielenie kodu, w tym zdalne, np. poprzez serwer Apache,
- archiwizacja pracy, w tym historia zmian nazw katalogów i plików,
- kontrola wersji, w tym efektywne wersjonowanie plików binarnych,
- łatwe tworzenie wydań, gałęzi rozwojowych,
- rozstrzyganie konfliktów.

Pojedynczemu programiście pozwala na:

- pracę nad projektem na kilku komputerach,
- cofnięcie każdej zmiany, oznaczanie wydań, tworzenie gałęzi.

Subversion — architektura



Subversion — komponenty

- **svn** — klient SVN do pracy w trybie tekstowym
- **svnversion** — program do monitorowania stanu kopii roboczej
- **svnlook** — narzędzie do inspekcji repozytorium
- **svnadmin** — narzędzie do tworzenia, optymalizowania lub naprawy repozytorium
- **mod_dav_svn** — moduł serwera Apache do zdalnego udostępniania repozytorium
- **svnserve** — dedykowany serwer SVN,
- **svndumpfilter** — program do filtrowania repozytoriów SVN,
- **svnsync** — program do przyrostowego tworzenia obrazów repozytoriów.

Subversion — nomenklatura

Repozytorium — zestaw plików i folderów zarządzanych i udostępnianych przez serwer SVN. System SVN jest serwerem plików, który nadaje całemu repozytorium dodatkowy wymiar: numer rewizji. W każdej chwili użytkownicy mogą uzyskać dostęp do całego repozytorium (czyli drzewa katalogów i plików) w stanie z zadanej rewizji.

Kopia robocza — komplet plików tworzących projekt przekopiowany z serwera SVN na dysk lokalny. Kopia robocza jest w 100% prywatną własnością: oprogramowanie SVN nigdy samowolnie nie dokona w niej żadnych zmian.

Rewizja — stan systemu plików przechowywanych przez serwer SVN po wykonaniu zadanej liczby zmian. Każda operacja przesyłająca kopię roboczą na serwer zwiększa numer rewizji.

Kolizja — sytuacja, w której dwie lub więcej osób wykonały wykluczające się zmiany w jednym z plików. Serwer SVN nie może w takiej sytuacji automatycznie rozstrzygnąć, która wersja pliku ma być traktowana jako bieżąca. Decyzję o tym, który plik ma być aktualny, podejmuje człowiek, po ręcznej analizie plików.

Subversion — struktura repozytoriów i dostęp do nich

Zalecana struktura repozytoriów:

projekt/	główny katalog projektu
trunk/	główna ścieżka rozwoju projektu
tags/	katalog kopii projektu, które nie będą już modyfikowane
branches/	katalog kopii rozwojowych projektu

Adresy URL repozytoriów:

file://	bezpośredni dostęp do repozytorium (na dysku lokalnym)
http://	dostęp przez protokół WebDAV do serwera Apache
https://	jak http://, ale z szyfrowaniem SSL
svn://	dostęp przez protokół serwera svnserver
svn+ssh://	jak svn://, ale przez tunel SSH

Subversion — tworzenie repozytorium, jego struktura

```
svnadmin create  
svn ls
```

```
~>svnadmin create --fs-type fsfs /tmp/svn  
~>svn ls file:///tmp/svn  
~>ls /tmp/svn/  
conf  db  format  hooks  locks  README.txt  
~>du -sh /tmp/svn/  
136K /tmp/svn/
```

Subversion — operowanie na repozytorium

```
svn mkdir
svn copy
svn move
svn delete
```

```
~>svn mkdir file:///tmp/svn/Przykladowy -m "Utworzenie katalogu"
```

Zatwierdzona wersja 1.

```
~>svn ls file:///tmp/svn
```

Przykladowy/

```
~>ls /tmp/svn/
```

conf db format hooks locks README.txt

```
~>du -sh /tmp/svn
```

144K /tmp/svn

```
~>svn delete file:///tmp/svn/Przykladowy -m "Usuniecie katalogu"
```

Zatwierdzona wersja 2.

```
~>du -sh /tmp/svn
```

152K /tmp/svn

Subversion — historia zmian

```
svn log
```

```
~>svn log file:///tmp/svn/
```

```
-----  
r2 | mucha | 2010-03-15 19:00:09 +0100 (pon, 15.03.2010) | 1 line
```

```
Usuniecie przykladowego katalogu
```

```
-----  
r1 | mucha | 2010-03-15 18:59:19 +0100 (pon, 15.03.2010) | 1 line
```

```
Utworzenie przykladowego katalogu
```

Subversion — umieszczenie projektu w repozytorium

svn import

```
~>cd sciezka/do/projektu/  
~/sciezka/do/projektu>tree
```

```
.  
|-- README  
|-- bin  
|   '-- main  
|-- doc  
|   |-- README  
|   '-- index.html  
|-- obj  
|   |-- filter.o  
|   '-- main.o  
'-- src  
    |-- Makefile  
    |-- filter.cpp  
    |-- filter.hh  
    '-- main.cpp
```

4 katalogi, 10 plikow

```
~/s/d/p>make -f src/Makefile clean  
~/s/d/p>svn import . file:///tmp/svn/MojProjekt/trunk  
                        -m "Inicjalizacja mojego projektu"
```

```
Dodawanie      doc  
Dodawanie      doc/index.html  
Dodawanie      doc/README  
Dodawanie      src  
Dodawanie      src/filter.hh  
Dodawanie      src/main.cpp  
Dodawanie      src/filter.cpp  
Dodawanie      src/Makefile  
Dodawanie      bin  
Dodawanie      obj  
Dodawanie      README
```

Zatwierdzona wersja 3.

Subversion — umieszczenie projektu w repozytorium

```
~/sciezka/do/projektu>svn log file:///tmp/svn/
```

```
-----  
r3 | mucha | 2010-03-15 19:57:13 +0100 (pon, 15.03.2010) | 1 line
```

```
Inicjalizacja mojego projektu
```

```
-----  
r2 | mucha | 2010-03-15 19:00:09 +0100 (pon, 15.03.2010) | 1 line
```

```
Usuniecie przykladowego katalogu
```

```
-----  
r1 | mucha | 2010-03-15 18:59:19 +0100 (pon, 15.03.2010) | 1 line
```

```
Utworzenie przykladowego katalogu
```

Subversion — pobranie aktualnej kopii roboczej

svn checkout

```
~/sciezka/do/projektu>cd ..
~/sciezka/do>rm -rf projektu/
~/sciezka/do>svn checkout file:///tmp/svn/MojProjekt
A    MojProjekt/trunk
A    MojProjekt/trunk/doc
A    MojProjekt/trunk/doc/index.html
A    MojProjekt/trunk/doc/README
A    MojProjekt/trunk/src
A    MojProjekt/trunk/src/filter.hh
A    MojProjekt/trunk/src/main.cpp
A    MojProjekt/trunk/src/filter.cpp
A    MojProjekt/trunk/src/Makefile
A    MojProjekt/trunk/bin
A    MojProjekt/trunk/obj
A    MojProjekt/trunk/README
Pobrano wersje 3.
~/sciezka/do>ls
MojProjekt
```

Subversion — praca nad monitorowanym projektem

svn info

```
~/sciezka/do>cd MojProjekt/  
~/sciezka/do/MojProjekt>ls -a  
.  ..  .svn  trunk  
~/sciezka/do/MojProjekt>svn info  
Sciezka: .  
URL: file:///tmp/svn/MojProjekt  
Katalog glowny repozytorium: file:///tmp/svn  
UUID repozytorium: 729ab0da-304c-11df-91b3-c56cc90b270f  
Wersja: 3  
Rodzaj obiektu: katalog  
Zlecenie: normalne  
Autor ostatniej zmiany: mucha  
Ostatnio zmieniona wersja: 3  
Data ostatniej zmiany: 2010-03-15 19:57:13 +0100 (pon, 15.03.2010)  
~/sciezka/do/MojProjekt>cd trunk; ls -a  
.  ..  bin  doc  obj  README  src  .svn
```

Subversion — praca nad monitorowanym projektem

```
svn status  
svn add
```

#edytowano main.cpp dodano class.cpp class.hh

~/sciezka/do/MojProjekt/trunk>make

~/sciezka/do/MojProjekt/trunk>svn status

? src/class.hh

? src/class.cpp

M src/main.cpp

? bin/main

~/sciezka/do/MojProjekt/trunk>svn add src/class.hh src/class.cpp

A src/class.hh

A src/class.cpp

~/sciezka/do/MojProjekt/trunk>svn status

A src/class.hh

M src/main.cpp

A src/class.cpp

? bin/main

Subversion — zatwierdzanie zmian

svn commit

```
~/sciezka/do/MojProjekt/trunk>svn commit -m "Dodano pliki klas"
```

```
Dodawanie      trunk/src/class.cpp
```

```
Dodawanie      trunk/src/class.hh
```

```
Wysylanie      trunk/src/main.cpp
```

```
Przesylanie tresci pliku...
```

```
Zatwierdzona wersja 4.
```

```
~/sciezka/do/MojProjekt/trunk>svn status
```

```
?      bin/main
```

```
~/sciezka/do/MojProjekt/trunk>svn log
```

```
-----  
r3 | mucha | 2010-03-15 19:57:13 +0100 (pon, 15.03.2010) | 1 line
```

```
Inicjalizacja mojego projektu  
-----
```

Subversion — aktualizacja kopii roboczej

svn update

```
~/sciezka/do/MojProjekt/trunk>svn update
```

W wersji 4.

```
~/sciezka/do/MojProjekt/trunk>svn log
```

```
-----  
r4 | mucha | 2010-03-28 20:08:38 +0200 (nie, 28.03.2010) | 1 line
```

Dodano pliki klas

```
-----  
r3 | mucha | 2010-03-15 19:57:13 +0100 (pon, 15.03.2010) | 1 line
```

Inicjalizacja mojego projektu

```
-----
```

Subversion — pobieranie dowolnej wersji

```
svn checkout (co) -r nr_rewizji  
svn update (up) -r nr_rewizji
```

```
~/sciezka/do/MojProjekt/trunk>cd /tmp  
/tmp>svn co -r 4 file:///tmp/svn/ proba
```

```
A   proba/MojProjekt  
A   proba/MojProjekt/trunk  
A   proba/MojProjekt/trunk/doc  
A   proba/MojProjekt/trunk/doc/index.html  
A   proba/MojProjekt/trunk/doc/README  
A   proba/MojProjekt/trunk/src  
A   proba/MojProjekt/trunk/src/class.hh  
A   proba/MojProjekt/trunk/src/filter.hh  
A   proba/MojProjekt/trunk/src/main.cpp  
A   proba/MojProjekt/trunk/src/class.cpp  
A   proba/MojProjekt/trunk/src/filter.cpp  
A   proba/MojProjekt/trunk/src/Makefile  
A   proba/MojProjekt/trunk/bin  
A   proba/MojProjekt/trunk/obj  
A   proba/MojProjekt/trunk/README
```

Pobrano wersje 4.

Subversion — pobieranie dowolnej wersji

```
/tmp>ls proba
```

```
MojProjekt
```

```
/tmp>svn info proba/MojProjekt/trunk/src/main.cpp
```

```
Sciezka: proba/MojProjekt/trunk/src/main.cpp
```

```
Katalog glowny repozytorium: file:///tmp/svn
```

```
Wersja: 4
```

```
Rodzaj obiektu: plik Zlecenie: normalne Autor ostatniej zmiany: mucha
```

```
Ostatnio zmieniona wersja: 4
```

```
Data ostatniej zmiany: 2010-03-28 20:08:38 +0200 (nie, 28.03.2010)
```

```
Tresc ostatnio aktualizowana: 2010-03-28 20:30:21 +0200 (nie, 28.03.2010)
```

```
/tmp>svn info proba/MojProjekt/trunk/doc/index.html
```

```
Sciezka: proba/MojProjekt/trunk/doc/index.html
```

```
Katalog glowny repozytorium: file:///tmp/svn
```

```
UUID repozytorium: 729ab0da-304c-11df-91b3-c56cc90b270f
```

```
Wersja: 4
```

```
Rodzaj obiektu: plik Zlecenie: normalne Autor ostatniej zmiany: mucha
```

```
Ostatnio zmieniona wersja: 3
```

```
Data ostatniej zmiany: 2010-03-15 19:57:13 +0100 (pon, 15.03.2010)
```

```
Tresc ostatnio aktualizowana: 2010-03-28 20:30:21 +0200 (nie, 28.03.2010)
```

Subversion — pobieranie dowolnej wersji

```
/tmp>svn co -r 3 file:///tmp/svn/ proba
D   proba/MojProjekt/trunk/src/class.hh
D   proba/MojProjekt/trunk/src/class.cpp
U   proba/MojProjekt/trunk/src/main.cpp
Pobrano wersje 3.
/tmp>svn co -r 1 file:///tmp/svn/ proba
D   proba/MojProjekt
A   proba/Przykladowy
Pobrano wersje 1.
/tmp>ls proba
Przykladowy
/tmp>svn co file:///tmp/svn/MojProjekt proba1
... Pobrano wersje 4.
/tmp>svn co -r 5 file:///tmp/svn/MojProjekt proba1
svn: Nie ma takiej wersji 5
/tmp>svn co -r 1 file:///tmp/svn/MojProjekt proba1
svn: Nie odnaleziono lokalizacji repozytorium dla
      'file:///tmp/svn/MojProjekt' w wersji 1
```

Subversion — pobranie kopii niepodlegającej zarządzaniu

```
svn export
```

```
/tmp>svn export file:///tmp/svn/MojProjekt/trunk proba
```

```
A   proba
A   proba/doc
A   proba/doc/index.html
A   proba/doc/README
A   proba/src
A   proba/src/class.hh
A   proba/src/filter.hh
A   proba/src/main.cpp
A   proba/src/class.cpp
A   proba/src/filter.cpp
A   proba/src/Makefile
A   proba/bin
A   proba/obj
A   proba/README
```

```
Wyeksportowano wersje 4.
```

Subversion — codzienna praca

- aktualizacja kopii roboczej

```
svn update (up)
```

- praca nad kopią roboczą

```
svn add                svn delete (del, remove, rm)
svn copy (cp)          svn move (mv, rename, ren)
svn status (stat, st)  svn diff (di)
svn update (up)         svn resolve
svn revert
```

- naniesienie zmian w repozytorium

```
svn commit (ci)
```

Subversion — tworzenie wydań

```
svn mkdir  
svn copy (cp)  
svn commit (ci)
```

```
/tmp>cd ~/sciezka/do/MojProjekt/  
~/sciezka/do/MojProjekt>svn mkdir tags  
A          tags  
~/sciezka/do/MojProjekt>svn cp trunk/ tags/before-qt  
A          tags/before-qt  
~/sciezka/do/MojProjekt>svn ci -m "Wersja przed  
                                zastosowaniem Qt" tags  
  
Dodawanie      tags  
Dodawanie      tags/before-qt  
  
Zatwierdzona wersja 5.  
~/sciezka/do/MojProjekt>cd tags  
~/sciezka/do/MojProjekt/tags>rm -rf before-qt
```

Subversion — tworzenie wydań

```
# usuwamy Makefile, stworzymy src.pro by uzywac qmake'a,  
# zmieniamy nazwe katalogu z dokumentacja  
~/sciezka/do/MojProjekt/tags>cd ../trunk/src  
~/sciezka/do/MojProjekt/trunk/src>svn rm Makefile  
D      Makefile  
~/sciezka/do/MojProjekt/trunk/src>svn add src.pro  
A      src.pro  
~/sciezka/do/MojProjekt/trunk/src>cd ..  
~/sciezka/do/MojProjekt/trunk>svn rename doc html  
A      html  
D      doc/index.html  
D      doc/README  
D      doc  
.../trunk>svn commit -m 'Przejscie na qmake. Zmiana... doc -> html'  
Usuwanie      trunk/doc  
Dodawanie      trunk/html  
Usuwanie      trunk/src/Makefile  
Dodawanie      trunk/src/src.pro  
Przesylanie tresci pliku.  
Zatwierdzona wersja 6.
```

Subversion — tworzenie wydań

```
~/sciezka/do/MojProjekt>du -sh
```

```
316K      .
```

```
~/sciezka/do/MojProjekt>du -sh /tmp/svn
```

```
184K      /tmp/svn
```

```
~/sciezka/do/MojProjekt>svn copy trunk/ tags/1.0.0
```

```
A          tags/1.0.0
```

```
~/sciezka/do/MojProjekt>svn commit -m "Wersja 1.0.0 z Qt" tags/1.0.0/
```

```
Dodawanie      tags/1.0.0
```

```
Usuwanie       tags/1.0.0/doc
```

```
Dodawanie      tags/1.0.0/html
```

```
Usuwanie       tags/1.0.0/src/Makefile
```

```
Dodawanie      tags/1.0.0/src/src.pro
```

```
Zatwierdzona wersja 7.
```

```
~/sciezka/do/MojProjekt>du -sh
```

```
544K      .
```

```
~/sciezka/do/MojProjekt>du -sh /tmp/svn
```

```
192K      /tmp/svn
```

Subversion — pobieranie wydań

```
~/sciezka/do/MojProjekt>cd /tmp
/tmp>svn export file:///tmp/svn/MojProjekt/tags/1.0.0
A      1.0.0/html
A      1.0.0/html/index.html
A      1.0.0/html/README
A      1.0.0/src
A      1.0.0/src/class.hh ...
A      1.0.0/src/filter.cpp
A      1.0.0/src/src.pro
A      1.0.0/bin
A      1.0.0/obj
A      1.0.0/README
Wyeksportowano wersje 7.
# tutaj do katalogu 1.0.0, mozna tez alternatywnie checkout
/tmp>svn ls file:///tmp/svn/MojProjekt
tags/
trunk/
/tmp>svn ls file:///tmp/svn/MojProjekt/tags
1.0.0/
before-qt/
```

Subversion — przywracanie poprzedniej wersji

```
svn diff
svn revert
```

```
# zawartosc przed edycja
~/s/d/M>cat trunk/src/main.cpp
cos
cos co dodal kuba
i jeszcze jedno
```

```
~/s/d/M>svn diff
```

```
Index: trunk/src/main.cpp
```

```
=====
```

```
--- trunk/src/main.cpp (wersja 8)
+++ trunk/src/main.cpp (kopia robocza)
@@ -1,3 +1,5 @@
+przed cos
  cos
-cos co dodal kuba
+poprawka kuby tekstu
  i jeszcze jedno
+ciag dalszy programu
```

```
~/s/d/M>svn revert trunk/src/main.cpp
```

```
Wycofano zmiany w 'trunk/src/main.cpp'
```

```
– Skład FoilTeX –
```

```
# zawartosc po edycji
~/s/d/M>cat trunk/src/main.cpp
przed cos
cos
poprawka kuby tekstu
i jeszcze jedno
ciag dalszy programu
```

Subversion — historia zmian

```
svn diff -r
```

```
~/sciezka/do/MojProjekt>svn -r 10 diff trunk/src/main.cpp
```

```
Index: trunk/src/main.cpp
```

```
=====
```

```
--- trunk/src/main.cpp (wersja 10)
```

```
+++ trunk/src/main.cpp (kopia robocza)
```

```
@@ -1,8 +1,10 @@
```

```
funkcja1  mucha
```

```
linia1  mucha
```

```
- linia2  mucha
```

```
linia3  mucha
```

```
+funkcja3  mucha
```

```
+ linia1  mucha
```

```
~/sciezka/do/MojProjekt>svn -r 3:7 diff trunk/src/main.cpp
```

```
Index: trunk/src/main.cpp
```

```
=====
```

```
--- trunk/src/main.cpp (wersja 3)
```

```
+++ trunk/src/main.cpp (wersja 7)
```

```
@@ -0,0 +1 @@
```

```
+cos
```

Subversion — rozstrzyganie konfliktów

```
svn status -u
svn resolve
svn resolved
```

nic nie zmienialismy w zrodlach, ktos inny tak

```
~/sciezka/do/MojProjekt>svn status
```

```
?      trunk/bin/main
```

```
~/sciezka/do/MojProjekt>svn status -u
```

```
*      13      trunk/src/main.cpp
```

```
?      trunk/bin/main
```

dokonalismy zmian po ostatniej aktualizacji kopii roboczej

```
~/sciezka/do/MojProjekt>svn status
```

```
M      trunk/src/main.cpp
```

```
?      trunk/bin/main
```

```
~/sciezka/do/MojProjekt>svn status -u
```

```
M      *      13      trunk/src/main.cpp
```

```
?      trunk/bin/main
```

```
Status wzgledem wersji:      14
```

Subversion — rozstrzyganie konfliktów

```
~/sciezka/do/MojProjekt>svn ci
```

```
Wysylanie      trunk/src/main.cpp
```

```
svn: Zatwierdzenie nie powiodlo sie (szczegoly ponizej):
```

```
svn: Plik '/MojProjekt/trunk/src/main.cpp' jest nieaktualny
```

```
svn: Opis zmian pozostawiono w pliku tymczasowym:
```

```
svn:      '~/sciezka/do/MojProjekt/svn-commit.tmp'
```

```
~/sciezka/do/MojProjekt>svn up
```

```
G      trunk/src/main.cpp                <--- G - poloczony
```

```
Uaktualniono do wersji 14.
```

```
~/sciezka/do/MojProjekt>svn ci
```

```
Wysylanie      trunk/src/main.cpp
```

```
Przesylanie tresci pliku.
```

```
Zatwierdzona wersja 15.
```

```
~/sciezka/do/MojProjekt>svn up
```

```
Odkryto konflikt w 'trunk/src/main.cpp'.      <--- moze sie zdarzyc
```

```
Wybierz: (p) odloz, (df) pokaz roznice w calosci, (e) zmien,
```

```
          (h) pomoc opisujaca wiecej opcji: p
```

```
C      trunk/src/main.cpp
```

```
Uaktualniono do wersji 16.
```

Subversion — rozstrzyganie konfliktów

```
~/s/d/M>ls trunk/src/
```

```
class.cpp  filter.cpp  main.cpp      main.cpp.r15  src.pro
```

```
class.hh   filter.hh   main.cpp.mine main.cpp.r16
```

#main.cpp - plik z zaznaczonymi różnicami, pozostałe - w/g nazwy

```
~/s/d/M>svn resolve --accept mine-full trunk/src/main.cpp
```

Rozwiązano konflikt w 'trunk/src/main.cpp'

```
~/s/d/M>svn resolved
```

Rozwiązano konflikt w 'trunk/src/main.cpp'

#poprzez skasowanie plików pomocniczych - zostaje main.cpp!!!

```
~/s/d/M>svn ci
```

Wysyłanie trunk/src/main.cpp

Przesyłanie treści pliku.

Zatwierdzona wersja 17.

Subversion — automatyczne rozstrzyganie konfliktów

plik przed zmianami

```
int funkcja1 () {  
    int tmp1=0;  
    printf("Z funkcji 1\n");}  
  
int funkcja2 () {  
    int tmp2=0;  
    printf("Z funkcji 2\n");}
```

zmiany pierwszego użytkownika

```
int funkcja1 () {  
    int tmp1=221;  
    printf("Z funkcji 1\n");}  
  
int funkcja2 () {  
    int tmp2=0;  
    printf("Z funkcji 2\n");}
```

zmiany drugiego użytkownika

```
int funkcja1 () {  
    int tmp1=0;  
    printf("Z funkcji 1\n");}  
  
int funkcja2 () {  
    int tmp2=0;  
    printf("Z funkcji 2 rozszerzone\n");}
```

plik po automatycznym połączeniu

```
int funkcja1 () {  
    int tmp1=221;  
    printf("Z funkcji 1\n");}  
  
int funkcja2 () {  
    int tmp2=0;  
    printf("Z funkcji 2 rozszerzone\n");}
```

Subversion — rozstrzyganie konfliktów z interwencją

plik przed zmianami

```
int funkcja1 () {  
    int tmp1=0;  
    printf("Z funkcji 1\n");}
```

zmiany pierwszego użytkownika

```
int funkcja1 () {  
    int tmp1=221;  
    printf("Z funkcji 1\n");}
```

zmiany drugiego użytkownika

```
int funkcja1 () {  
    int tmp1=113;  
    printf("Z funkcji 1\n");}
```

plik roboczy po wykryciu konfliktu

```
int funkcja1 () {  
<<<<<<< .mine  
    int tmp1=221;  
=====  
    int tmp1=113;  
>>>>>>> .r24  
    printf("Z funkcji 1\n");}
```

- Możliwe wartości opcji `--accept` przy `svn resolve`:
 - `mine-full`,
 - `theirs-full`,
 - `base`,
 - `working` – ze znacznikami jak w przykładzie powyżej!!!

Subversion — atrybuty plików

```
svn propset  
svn propedit  
svn propget  
svn proplist  
svn propdel
```

Przykład: słowa kluczowe w plikach

```
# w pliku main.cpp dodajemy na początku linie zawierające kolejno:  
# $Date$, $Revision$, $Author$, $HeadURL$, $Id$  
~/s/d/M>svn propset svn:keywords "Date Revision Author  
                                         HeadURL Id" trunk/src/main.cpp  
  
~/s/d/M>svn ci  
~/s/d/M>svn up  
# teraz plik main.cpp zawiera  
$Date: 2010-04-05 21:30:24 +0200 (pon, 05.04.2010) $  
$Revision: 26 $  
$Author: mucha $  
$HeadURL: file:///tmp/svn/MojProjekt/trunk/src/main.cpp $  
$Id: main.cpp 26 2010-04-05 19:30:24Z mucha $
```

Subversion — inne komendy

```
svn blame (praise, annotate, ann)
svn diff > patchfile    #latka w standardowym formacie diff
svn cleanup
svn help opcja
svn lock
```

```
~/sciezka/do/MojProjekt>svn blame trunk/src/main.cpp
```

```
10      mucha  funkcja1
10      mucha   linia1
10      mucha   linia3
10      mucha
10      mucha  funkcja2
12      mucha   linia11
10      mucha   linia2
11      kuba
12      mucha  funkcja3
12      mucha   linia1
```

Subversion — zarządzanie repozytorium

- kopia zapasowa repozytorium
 - w formacie tekstowym

```
svnadmin dump
```

- w formacie binarnym (jest zwykłym repozytorium gotowym do pracy)

```
svnadmin hotcopy
```

- migracja repozytorium – pobranie historii projektu

```
svndumpfilter
```

- wczytanie archiwum

```
svnadmin load
```

```
~>svnadmin dump /tmp/svn | gzip > kopia9.gz
```

```
~>svnadmin hotcopy /tmp/svn /tmp/backup
```

```
~>svnadmin dump /tmp/svn | svndumpfilter include Proj > proj.svn
```

```
~>svnadmin load /tmp/svn < proj.svn
```

Subversion — podsumowanie funkcjonalności

- Tworzenie, zarządzanie, archiwizowanie repozytorium
- Umieszczenie projektu
- Aktualizacja kopii roboczej
- Zatwierdzanie zmian
- Historia zmian
- Tworzenie i pobieranie wydań
- Tworzenie gałęzi
- Historia zmian
- Rozstrzyganie konfliktów
- Nadawanie atrybutów

Subversion — klienty

- TortoiseSVN (integruje się z MS Exploratorium Windows)
- RapidSVN (linux)
- KdeSVN (linux)
- eSVN (napisany w Qt)
- WebSVN (nakładka webowa, napisana w PHP i działająca w przeglądarce internetowej)
- AnkhSVN (rozszerzenie do środowiska Microsoft Visual Studio)
- integracja z edytorami, środowiskami (GNU Emacs, Eclipse, Code::Blocks)
- hosting projektów Open Source
 - SourceForge
 - Google Code
 - OpenSVN
 - CodePlex (prowadzony przez Microsoft)
- repozytoria publiczne

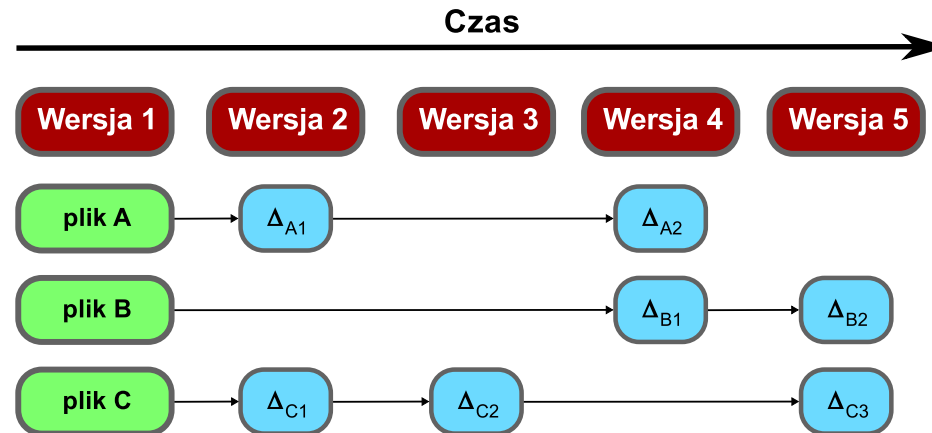
Był CVS — jest SVN — będzie Git?!¹

- CVS — pierwszy powszechnie używany, wciąż utrzymywany ale nierozwijany
- SVN — obecnie podstawowy, główne założenie przy tworzeniu: pozbyć się największych błędów CVSa
- Git — coraz bardziej popularny, główne założenie przy tworzeniu: WWCVSND²
 - system rozproszony — możliwość pracy offline, bez połączenia z głównym repozytorium (a nawet bez niego)
 - kryptograficzna autentykacja historii — nazwa danej wersji zależy od całej historii prowadzącej do niej (SHA-1)
 - duża szybkość — ze względu na lokalność większości dokonywanych operacji
 - darmowe (dla projektów opensource'owych/niedużych), ogólnodostępne repozytoria Git — github, bitbucket
 - integracja z popularnymi narzędziami — GNU Emacs, Eclipse, Redmine

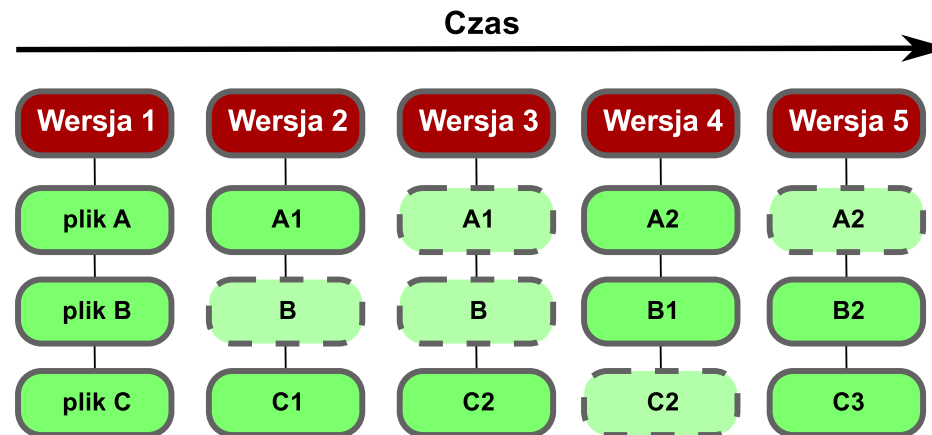
¹tytuł historyczny

²What Would CVS Not Do

Filozofia struktury Gita



Przyrostowa archiwizacja plików z wykorzystaniem listy zmian (patches — łatki)



Asocjacyjna archiwizacja plików z wykorzystaniem referencji (snapshots — migawki?)

Internetowe serwisy hostingowe

Serwis	Wsparcie GIT	Wsparcie SVN	Inspekcja kodu	System ticketów
Amazon AWS CodeCommit	×			×
Assembla	×	×	×	×
Beanstalk	×	×	×	
Bitbucket	×	×	×	×
CloudForge	×	×		×
Codebase	×	×	×	×
Fog Creek Kiln	×	×	×	×
GitHub	×	×	×	×
GitLab	×		×	×
Launchpad	×		×	×
Microsoft Azure DevOps	×		×	×
Planio	×	×		×
Perforce	×		×	×
RhodeCode	×	×	×	
SourceForge	×	×		×

na podstawie <https://www.slant.co/topics/153/~best-hosted-version-control-services>

i <https://www.git-tower.com/blog/git-hosting-services-compared/>

SVN vs Git

Różnice:

- Git jest systemem rozproszonym, SVN scentralizowanym
- Git jest szybszy w działaniu
- Repozytoria Gita są znacznie mniejsze (np. dla projektu Mozilla (3GB, 10 lat) 420MB vs 12GB)
- SVN pozwala na pobranie fragmentu projektu, Git wymaga sklonowania całego repozytorium
- Obsługa gałęzi w Gicie jest prostsza i mniej zasobożerna
- Format repozytorium Gita jest prosty, przez to niepodatny na uszkodzenia i łatwy w naprawie
- Łatwiej archiwizować jest scentralizowane repozytoria SVN, jednakże klony rozproszonych repozytoriów Git mogą być postrzegane jako archiwa
- SVN posiada bardziej dojrzały interfejs
- Przeglądanie sekwencji wersji jest prostsze w SVN dzięki ich sekwencyjnemu numerowaniu (w Gicie jedynie wstecz)
- Sposób konwersji znaków końca linii (Git – automatyczna, SVN – konfigurowalna)

SVN vs Git

Zalety Gita:

- Trywialne inicjowanie repozytorium (`mkdir phi; cd phi; git init`)
- Mniejsze repozytoria i większa szybkość działań (ze względu na ich lokalność)
- Możliwość pracy offline
- Możliwość rejestrowania zmian w lokalnym repo częściej niż w głównym
- Prostszy sposób obsługi gałęzi (a także etykietowania (tagging))
- Klony repozytoriów pełnią rolę archiwów
- Zawiera polecenie `blame-someone-else`

Zalety SVNa:

- Większa liczba interfejsów użytkownika (graficzne, integracyjne) (git ma w zasadzie 3: linia poleceń, git-gui i QGit, ale ostatnio też gitk, TortiseGit, gitweb)
- Centralne repozytorium
 - łatwiejsza archiwizacja
 - łatwiejsza kontrola dostępu
- Możliwość pobrania fragmentu projektu
- Krótsze i przewidywalne numery rewizji
- Możliwość wykonywania bardziej złożonych zdarzeń

SVN vs Git

- **SVN wyewoluował z systemów kontroli wersji (VCS)**
(3. pokolenie po RCS i CVS)
- **Git wyewoluował z systemów zarządzania treścią (CMS)**
(konceptyjnie podobne do VCS, ale VCS są optymalizowane do przechowywania tekstu)
- **Jak pogodzić „gitowców” z „esfauenowcami”?**
Centralne repozytorium SVN z klientami SVN i klientami Git?

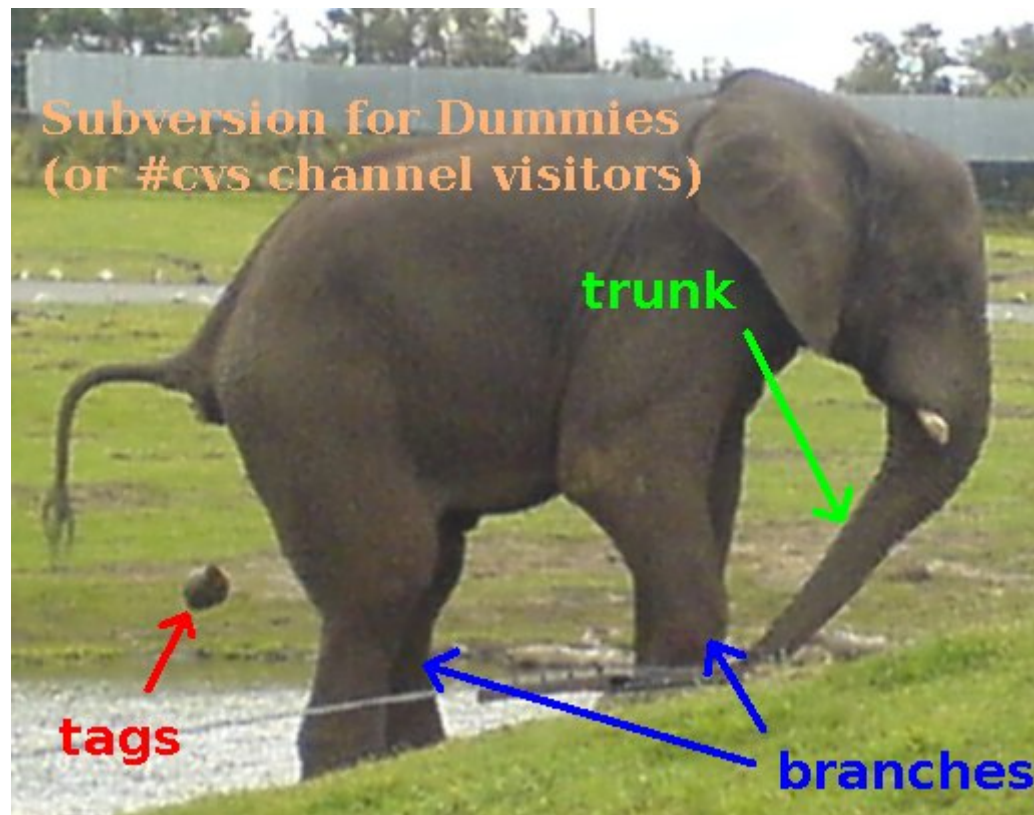
Więcej na

- <http://git-scm.com/book/pl/v1/Pierwsze-kroki-Podstawy-Git> (Pro Git)
- <https://try.github.io/> (Gitouczek)

Kontrola wersji — dobre praktyki

- **Kontroli wersji poddajemy „wszystko”** — kod aplikacji, dokumentację techniczną, projektową, instrukcje użytkownika, instalacji
- **Wykorzystujemy jednolitą konwencję struktury repozytoriów** — w ten sam sposób wydzielone główne/poboczne ścieżki rozwojowe, oznakowane wydania, jednolity sposób przechowywania dokumentacji, instrukcji
- **Określamy listę plików, które nie będą podlegały monitorowaniu** — np. przez ustalenie i przekazanie zespołom zasad pracy z systemem kontroli wersji
- **Zmiany zatwierdzamy często, niedużymi partiami** — np. po zrealizowaniu zadania, dodaniu funkcjonalności, co ułatwia rozwiązywanie konfliktów, scalanie gałęzi
- **Każdą zatwierdzoną zmianę opatrujemy komentarzem** — a w nim numer zadania/funkcjonalności, jego/jej krótki opis, informację o ewentualnych brakach
- **Informujemy wszystkich zainteresowanych o wprowadzaniu istotnych modyfikacji** — np. pocztą elektroniczną
- **Regularnie aktualizujemy kopię roboczą**
- **Oznakowujemy wydania stabilne oraz wszystkie rewizje przed scalaniem gałęzi**
- **Regularnie tworzymy kopie zapasowe repozytorium**
- **Nie modyfikujemy plików bezpośrednio w repozytorium**

SVN dla opornych



Motto

Nie wyobrażam sobie jak mogłem
programować kiedyś bez kontroli wersji