Spis treści

Spis treści

Spis treści

1 Wprowadzenie

1.1 Czym jest Subversion

Czym jest Subversion

- czym jest system kontroli wersji
- czym Subversion jest dla Ciebie
- krótka historia

System kontroli wersji:
Kontrola wersji jest sposobem zarządzania zmianami informacji (danych).
Subversion zarządza plikami oraz katalogami, zmianami jakie są w nich wprowadzane. Pozwala to na odzyskanie starych danych lub też sprawdzenie jak informacje zmieniały się z biegiem czasu.
Czym Subversion jest dla Ciebie?
Subversion jest wspaniałym narzędziem, które może sprawdzi się w wielu sytuacjach, ale nie w każdej. Jeżeli potrzebujesz:

- śledzić zmiany w swoich plikach (wiedzieć co, kiedy i przez kogo zostało zmienione, móc przywrócić dowolną poprzednią wersję danych)
- wymieniać się danymi z innymi ludźmi
- edytować pliki łącznie z innymi ludźmi (nawet w tym samym czasie)

Krótka historia
Subversion powstało jako zamiennik systemu kontroli wersji CSV, mający usunąć najbardziej irytujące błędy tegoż. Taki założenie miało następujące implikacje:

- zachowanie w SVN wszystkich właściwości CSV
- maksymalnie podobny (taki sam) sposób obsługi
- minimalizacja ilości przesyłanych danych

1.2 Czym jest repozytorium

Czym jest repozytorium

- scentralizowany system współdzielenia danych
- operacje jakie można wykonać na repozytorium:
  - dodawanie plików
  - usuwanie plików
  - zmiany nazwy pliku
  - przeniesienie pliku do innego katalogu
  - zmiana właściwości
- historia zmian
- odczyt danych przez klienta

Repozytorium

Subversion jest scentralizowanym system współdzielącym dane. Jego główną część stanowi repozytorium, w którym są zapisywane wszystkie dane. Klienci mogą połączyć się z repozytorium i zarówno odczytywać jak i zapisywać dane.

Repozytorium poza oczywiście możliwością zapisywania plików posiada dodatkową właściwość: pamięta *każdą* zmianę jaka zostanie w nim wprowadzona, każdą zmianę pliku, katalogu, właściwości, operacje kasowania, dodawania, zmiany nazwy czy inne.

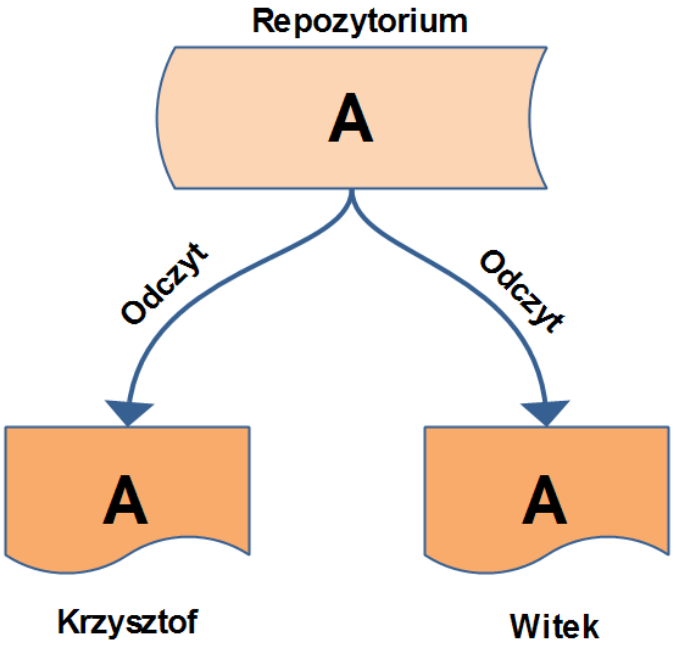
Podczas zwykłego odczytu danych z repozytorium klient odczytuje jego ostanie stan. Ale nic nie stoi na przeszkodzie aby odczytać dowolny poprzedni stan repozytorium. Można np. sprawdzić co zawierało ono dwa dni wcześniej lub też kto ostatnio zmodyfikował dany plik i jaka była jego postać wcześniejsza.

Na tym polega właśnie idea systemu kontroli wersjo: system zaprojektowany w taki sposób, aby śledzić zmiany danych w czasie.

2 Modele współdzielenia plików

2.1 Proste współdzielenie plików

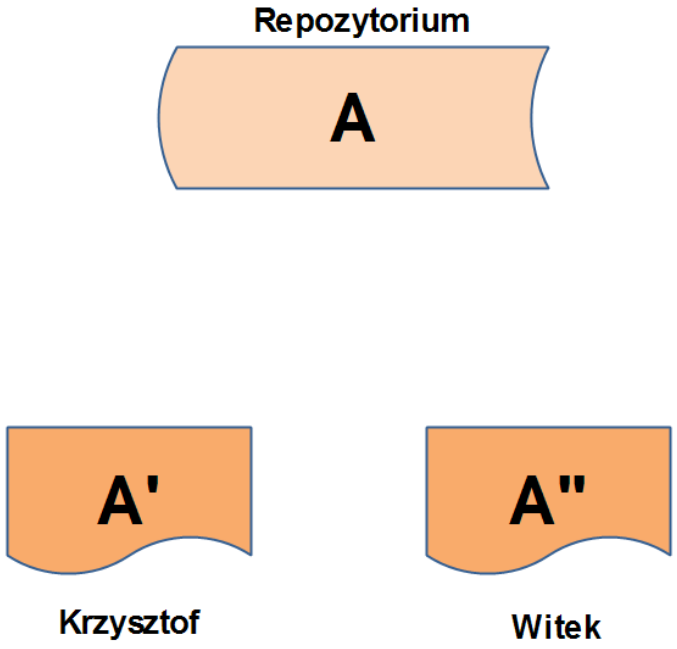
Współdzielenie plików Pobranie plików z repozytorium



Rysunek 1: Użytkownicy pobierają plik z repozytorium

Krzysztof i Witek posiadają kopię pliku znajdującego się w jakimś wspólnym zasobie.

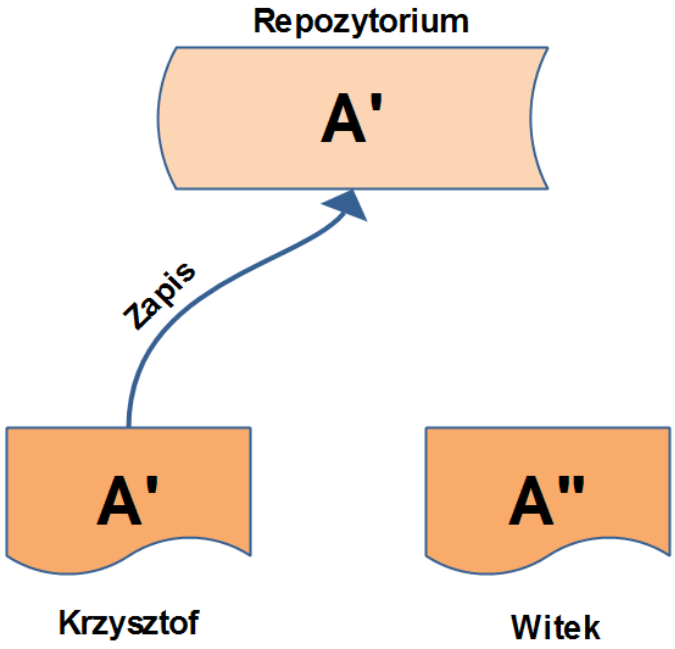
Współdzielenie plików Edycja plików



Rysunek 2: Użytkownicy edytują pobrane pliku

Krzysztof i Witek rozpoczynają edycję pobranego pliku. Każdy pracuje na własnej jego własnej kopii i w niej wprowadza konieczne modyfikacje.

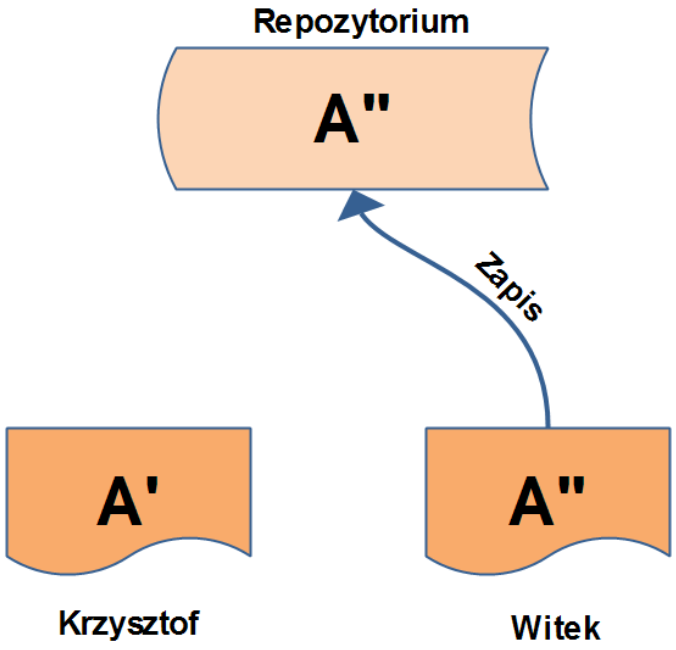
Współdzielenie plików Zapisanie pliku w repozytorium



Rysunek 3: Zapisanie zmodyfikowanego pliku

Krzysztof zapisuje zmodyfikowany plik do repozytorium.

Współdzielenie plików Kolejne zapisanie pliku w repozytorium



Rysunek 4: Drugi użytkownik zapisuje plik w repozytorium

Witek zapisuje własną wersję pliku do wspólnego repozytorium. Nie wie jednak, że posiada już nieaktualną wersję tego pliku i nadpisuje wersję Krzysztofa.

Tym samym część informacji zostaje stracona i jednocześnie nikt nie ma o tym pojęcia.

2.2 Zablokuj – zmodyfikuj – odblokuj

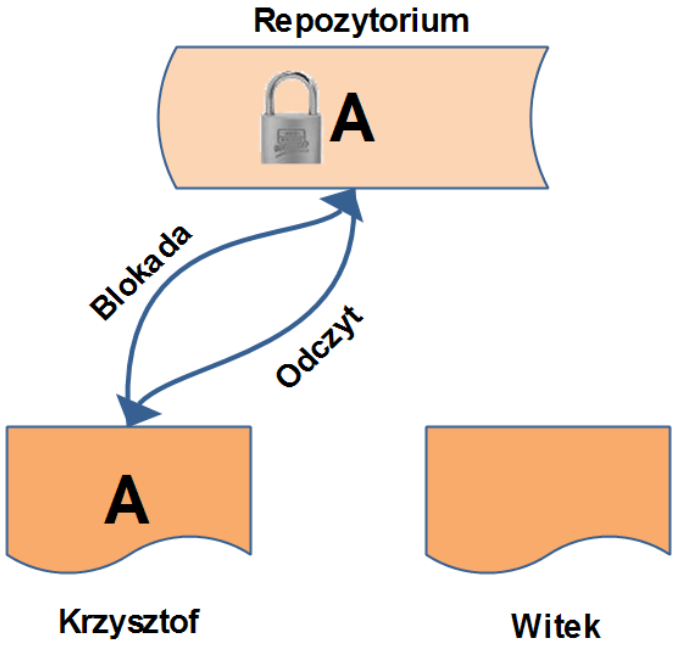
Zablokuj – zmodyfikuj – odblokuj Schemat działania

- sposób działania:
    1. zablokuj plik
    2. zmodyfikuj plik
    3. odblokuj plik

Istnieją systemy kontroli wersji które używają modelu zablokuj – odblokuj – zmodyfikuj w celu rozwiązania problemu współbieżnej edycji pliku przez kilku autorów. W tym modelu tylko jedna osoba może edytować plik w tym samym momencie.

Aby się to odbywało poprawnie, przed przystąpienie do edycji Krzysztof musi zablokować plik. Jeżeli w tym czasie Witek będzie chciał edytować ten sam plik, dostanie informacje nie jest to możliwe. Będzie musiał poczekać do czasu zwolnienia blokady przez Krzysztofa i dopiero wtedy przystąpić do edycji pliku.

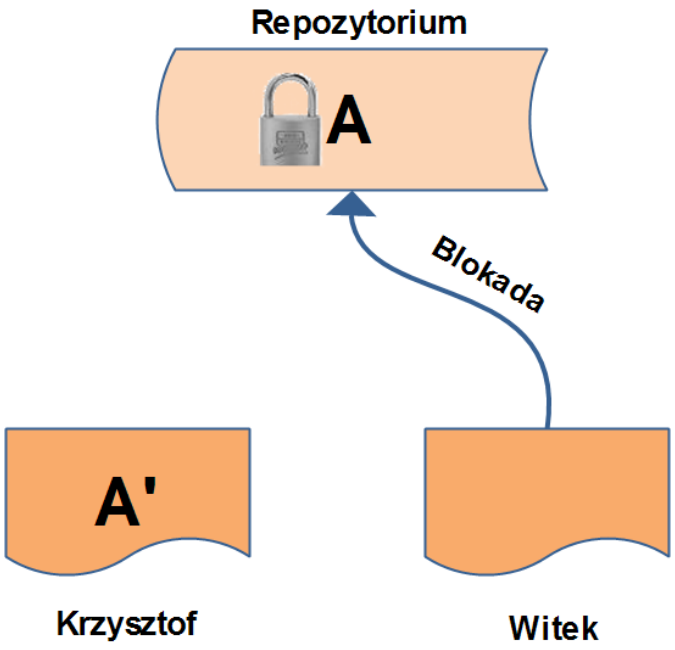
Blokowanie pliku Założenie blokady



Rysunek 5: Blokada i odczyt pliku

Krzysztof pobiera kopie pliku A i jednocześnie zakłada na nim blokadę. Dzięki temu nikt inny nie będzie mógł pobrać tego pliku w celu edycji ani zapisać jakichkolwiek zmian (przynajmniej do czasu istnienia blokady).

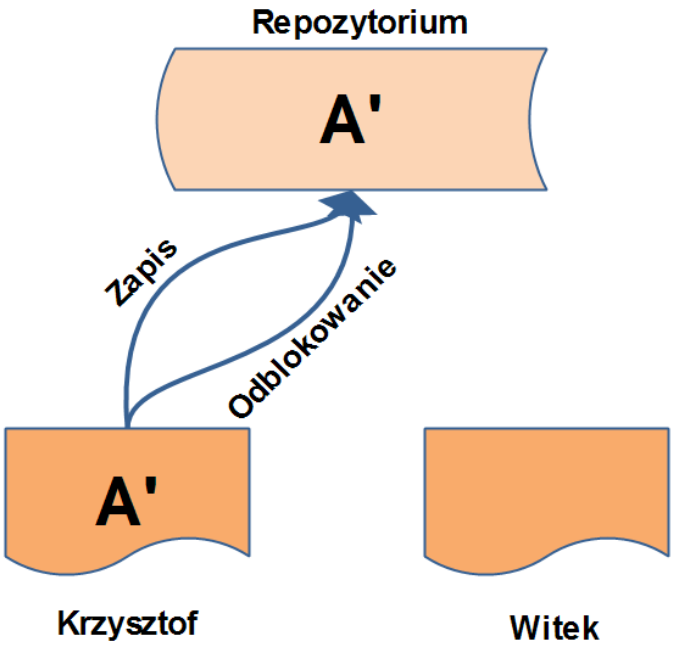
Blokowanie pliku Odczyt zablokowanego pliku



Rysunek 6: Próba pobranie zablokowanego pliku

Witek dokonuje próby pobrania pliku A także w celu edycji. Nie jest w stanie tego zrobić dopóki istnieje blokada założona na tym pliku.

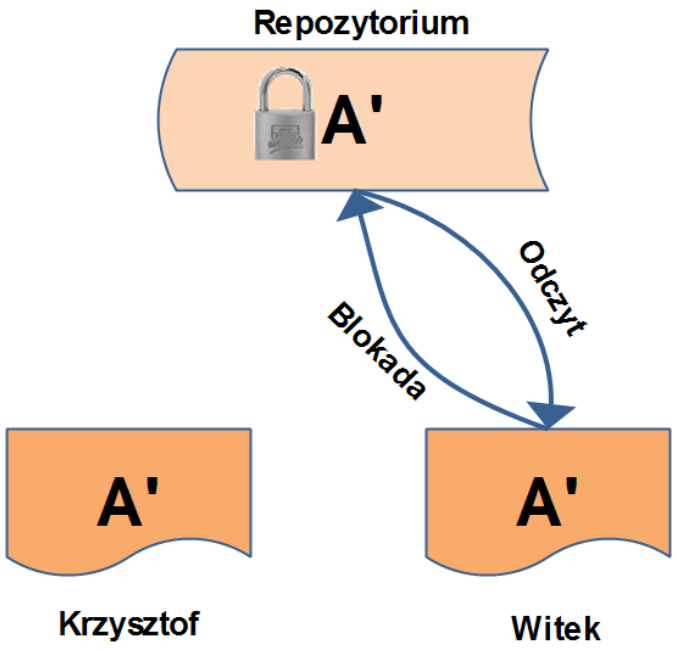
Blokowanie pliku Zapisanie pliku i zdjęcie blokady



Rysunek 7: Zapisanie pliku i zdjęcie blokady

Krzysztof zapisuje zmodyfikowany plik i zdejmuje z niego blokadę.

Blokowanie pliku Ponowna blokada pliku



Rysunek 8: Odczyt pliku i założenie blokady

Witek może teraz pobrać plik A i modyfikować go. Podczas pobierania zakłada na nim blokadę, dzięki czemu uzyskuje wyłączność na wprowadzenie modyfikacji w pliku A.

Problemy związane z blokowaniem plików

- blokowanie prowadzi do problemów administracyjnych
- blokowanie prowadzi do serializacji pracy
- blokowanie daje fałszywe poczucie bezpieczeństwa

Blokowanie może prowadzić do problemów administracyjny

Łatwo można wyobrazić sobie sytuację, że Witek zablokuje plik do edycji i zapomni o tym. Może nawet pojechać w tym czasie na wakacje, tym samym uniemożliwiając innym edycję zablokowanego pliku. W takiej sytuacji potrzebna jest interwencja administratora, który zdejmie odpowiednią blokadę. Oczywiście skutkuje to dłuższymi przestojami w pracy, oraz większą ilością pracy.

Blokowanie często prowadzi do niepotrzebnej serializacji pracy

Co występuje w sytuacji, gdy Witek chce edytować początek danego pliku a Tomek koniec? Każdy z nich edytuje inny jego fragment, ale podanym przypadku tylko jeden z nich może w danym momencie edytować plik. Druga osoba musi poczekać na zwolnienie blokady. Powoduje to, że nie można pracować nad tym samym plikiem jednocześnie.

Blokowanie może prowadzić do fałszywego poczucia bezpieczeństwa

Wyobraźmy sobie sytuację, że istnieją w repozytorium dwa pliku A i B. Pliki te są zależne od siebie, modyfikacje w jednym z nich mogą powodować błędy w drugim. Jeżeli teraz Witek zablokuje dostęp do pliku A a Krzysztof do pliku B, mogą oni odnieść wrażenie, że blokowanie zabezpiecza ich przed taką sytuacją.

Należy także pamiętać, że najważniejsza jest dobra komunikacja w zespole ludzi. Nie zastąpi jej w żaden sposób blokowanie plików, i nie może ona stad się tego substytutem.

2.3 Kopiuj – modyfikuj – scal

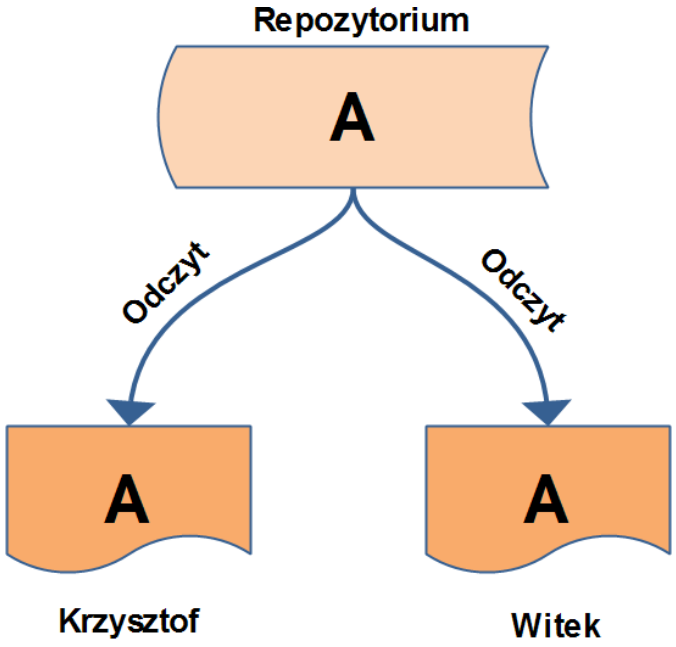
Kopiuj – modyfikuj – scal Schemat działania

- pobierz plik z repozytorium
- zmodyfikuj lokalną kopię pliku
- zapisz (scal) lokalny plik w repozytorium

Subversion (jak i również większość innych systemów kontroli wersji) używa modelu kopiuj – modyfikuj – scal jako podstawowego w stosunku do blokowania zasobów.

W takim modelu każdy użytkownik repozytorium tworzy jego własną kopię roboczą, zawierającą wszystkie pliki i katalogi. Następnie może na tej lokalnej kopii pracować i następnie scalić własne zmiany z głównym repozytorium. System kontroli wersji bardzo często pomaga podczas scalania, ale to człowiek jest odpowiedzialny za to aby zostało to wykonane prawidłowo.

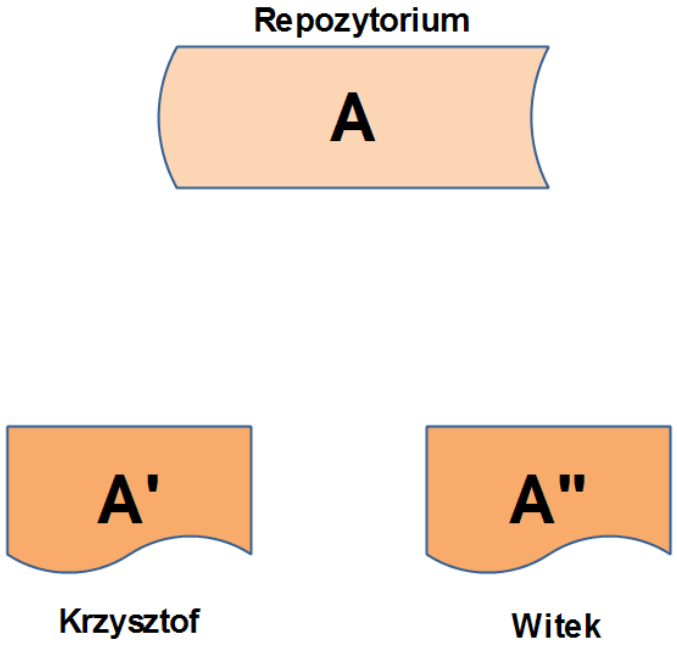
Kopiuj pliki Pobranie plików z repozytorium



Rysunek 9: Użytkownicy pobierają pliki z repozytorium

Zarówno Krzysztof jak i Witek pobierają plik A z wspólnego repozytorium do swoich lokalnych zasobów.

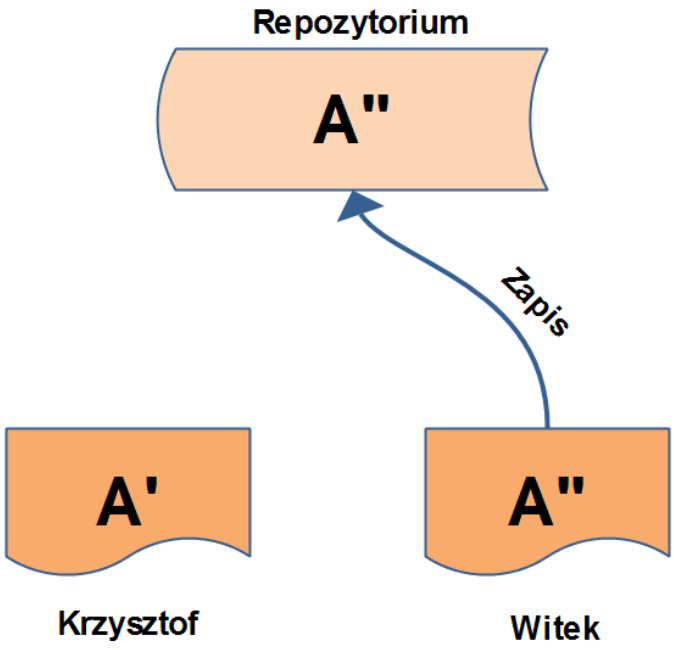
Edytuj pliki Użytkownicy edytują pliku



Rysunek 10: Użytkownicy edytują pliki

Krzysztof i Witek równocześnie wprowadzają zmiany w pobranym pliku.

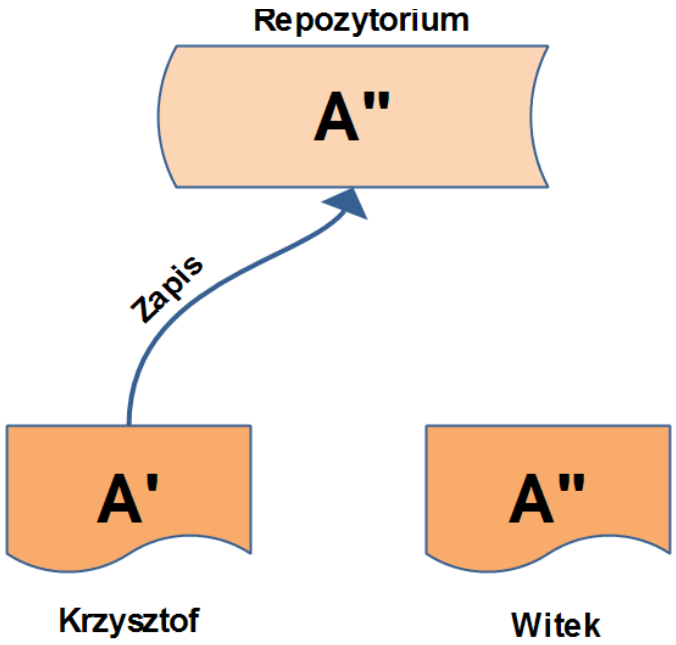
Scal modyfikacje Zapisanie pliku do repozytorium



Rysunek 11: Witek zapisuje plik w repozytorium

Witek jako pierwszy zapisuje plik w repozytorium. System odpowiedzialny za zarządzanie repozytorium sprawdzi, czy zapisywane zmiany odnoszą się do tej samej wersji pliku która jest w nim zapisane.

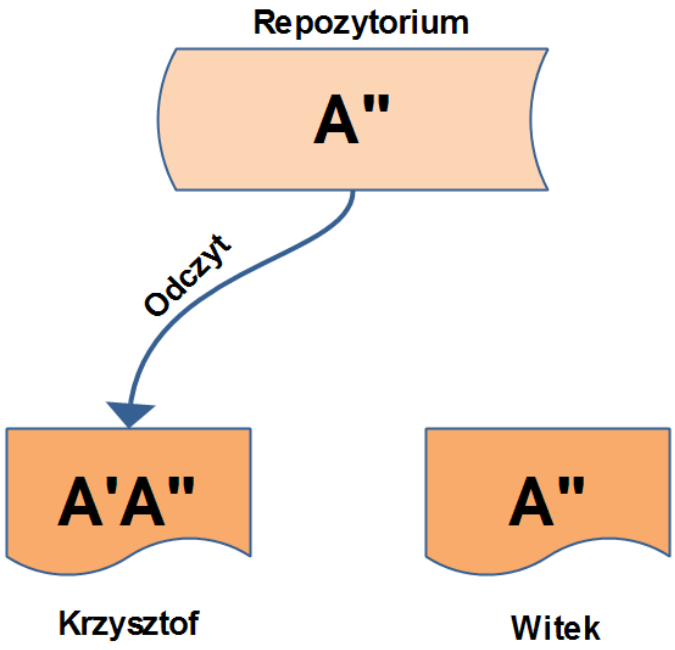
Scal modyfikacje Zapisanie pliku do repozytorium



Rysunek 12: Krzysztof zapisuje plik w repozytorium

Krzysztof chce zapisać swój plik w repozytorium. System kontroli wersji wykrywa jednak problem: w repozytorium znajduje się już nowsza wersja tego samego pliku. Nie można wobec tego nadpisać wersji w repozytorium.

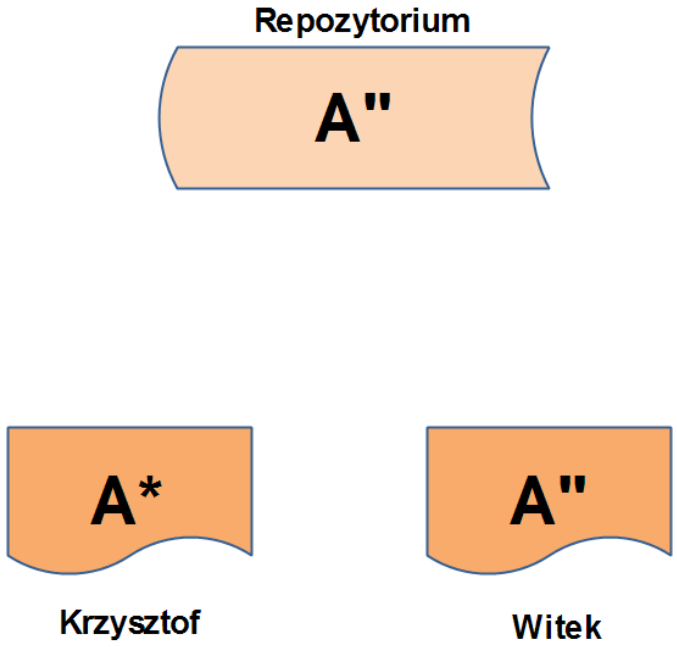
Scal modyfikacje Konflikt podczas zapisu modyfikacji



Rysunek 13: Konflikt podczas zapisu, odczytanie nowej wersji pliku

System kontroli wersji wykrył konflikt: próbę nadpisania nowszej wersji pliku. Aby poradzić sobie z tym konfliktem, zostaje pobrana wersja pliku z repozytorium i zaprezentowana użytkownikowi.

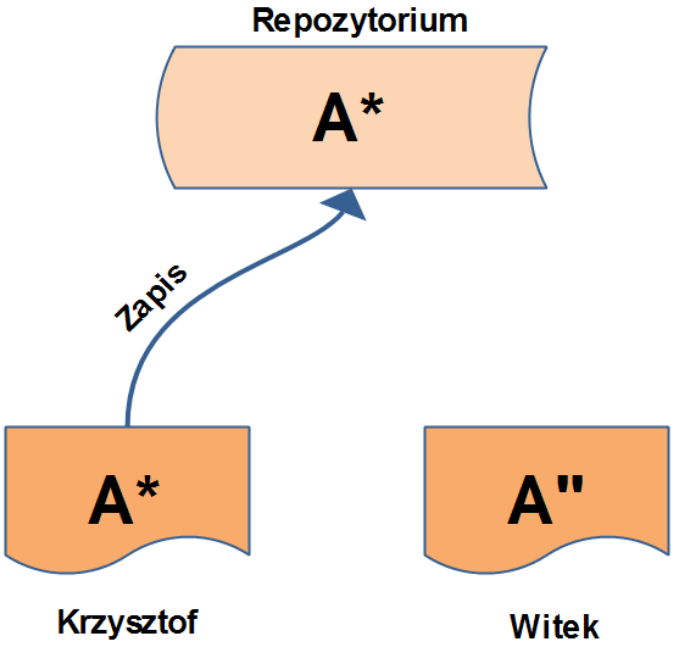
Scal modyfikacje Scalenie dwóch wersji plików



Rysunek 14: Użytkownik samodzielnie scala dwie wersje plików

Krzysztof musi teraz scalić zmiany z dwóch wersji plików do jednej. Cała odpowiedzialność za poprawność tego procesu spoczywa na nim.

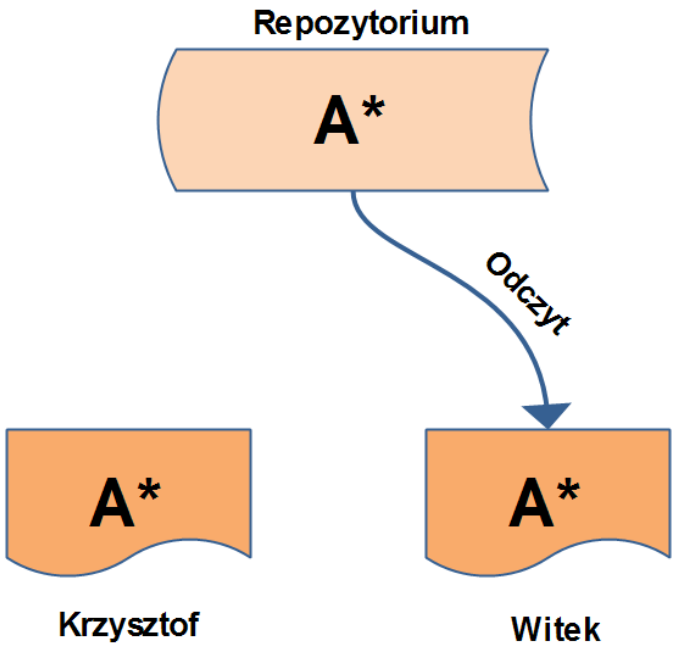
Zapisz plik Zapisanie do repozytorium scalonego pliku



Rysunek 15: Użytkownik zapisuje scalony plik w repozytorium

Po dokonaniu scalenia, może zapisać zmiany w repozytorium. Zostanie utworzona nowa wersja pliku zawierająca już zmiany Krzysztofa i Witka.

Aktualizacja kopii roboczej Aktualizacja własnej kopii roboczej



Rysunek 16: Użytkownicy pobierają aktualną wersję repozytorium

Witek pobiera z repozytorium nową wersję pliku i posiada także już aktualny stan.

Kiedy blokowanie jest potrzebne

Model kopiuj – modyfikuj – scal sprawdza się bardzo dobrze przy założeniu, że mamy do czynienia z plikami tekstowymi, które są łatwe w zarządzaniu i sprawdzaniu co się zmieniło.

W przypadku pracy z plikami binarnymi, których nie da się w łatwy, czy też nawet automatyczny sposób scalić, przydatne może być blokowanie plików przed rozpoczęciem ich modyfikacji.

3 Subversion w akcji

3.1 Udostępnianie repozytorium

Sposoby udostępniania repozytorium

Subversion umożliwia dostęp do repozytoriów na różne sposoby:

`file:///` bezpośredni dostęp do repozytorium (na lokalnym dysku, udziale sieciowym)

`http://` dostęp przez protokół WebDav (np. przy użyciu serwera WWW)

`https://` to samo co `http://` ale z szyfrowaniem SSL

`svn://` natywny protokół dostępu serwera `svnserver`

`svn+ssh://` to samo co `svn://` ale przy użyciu tunelowania SSH

Subversion umożliwia dostęp do repozytoriów na wiele sposobów. Może się zdarzyć, że dany klient SVN nie obsługuje wszystkich możliwych trybów dostępu do repozytorium.

Protokół `file:` jest najprostszy i jako jedyny nie wymaga żadnej konfiguracji dodatkowej (klient SVN troszczy się o wszystko). Podczas udostępniania repozytorium przy użyciu tego protokołu należy upewnić się, że dwaj klienci nie są w stanie w tym samym momencie modyfikować repozytorium. Odbywa się to na zasadzie blokowania plików, w związku z tym system plików musi udostępniać tę właściwość (nie zawsze tak się dzieje przy systemach sieciowych)

Protokół `http:` oraz `https:` umożliwiają integrację z serwerem WWW oraz udostępniają repozytorium poprzez protokół WebDav. Protokół ten jest używany także przez różne aplikacje niekoniecznie widzące co to jest SVN. Daje im to możliwość publikacji danych bezpośrednio w repozytorium, podczas gdy aplikacja nie ma pojęcia o jego zastosowaniu (przy użyciu autowersjonowania).

Protokoły `svn:` oraz `svn+ssh:` wymagają skonfigurowania oraz uruchomienia dodatkowej aplikacji serwera (svnserve).

3.2 Kopia robocza

Kopia robocza

- katalog, w którym można pracować
- można mieć wiele kopii roboczych (tego samego repozytorium)
- zmiany są zapisywane w repozytorium
- zawiera katalogi o nazwie `.svn`

Katalog roboczy SVN jest zwykłym katalogiem w systemie lokalnym. W ty miejscu można pracować nad projektem (łącznie z kompilacją projektu, jeżeli tego wynika).

SVN nigdy automatycznie nie umieszcza niczego w repozytorium, robi to dopiero na wyraźne żądanie użytkownika.

W każdym katalogu można znaleźć dodatkowy katalog administracyjny SVN: `.svn`. Zawiera on dodatkowe informacje administracyjne o danym katalogu (m.in. właściwości plików i katalogów, aktualną kopie każdego pliku z danego katalogu).

Dlaczego umieszczono kopię każdego pliku w `.svn`?

- zajmuje to miejsce na dysku
- umożliwia łatwe sprawdzenie jakie zmiany nastąpiły pomiędzy danym stanem repozytorium a aktualną wersją
- mniejsze ilości danych do przesłania między klientem i serwerem

3.3 Rewizje repozytorium

Rewizje repozytorium

Rewizje to kolejne wersje repozytorium, charakteryzują się:

- kolejne rewizje powstają po każdej zatwierdzonej zmianie stanu repozytorium
- każda rewizja powstaje poprzez zwiększenie poprzedniego numeru o jeden
- numer rewizji ma zastosowanie do wszystkich elementów repozytorium

Każda zmiana repozytorium powoduje zwiększenie numeru rewizji. W uproszczeniu dzieje się to po każdej akcji `commit` wykonanej na lokalnej kopii.

Ponieważ można także wprowadzać zmiany bezpośrednio w repozytorium (z pominięciem lokalnej kopii) to każda taka akcja także zwiększa numer wersji repozytorium.

W przeciwieństwie do wiele innych systemów kontroli wersji (np. CSV, gdzie numeracja jest w ramach pojedynczego pliku) rewizja repozytorium odnosi się do wszystkich plików i katalogów. Najprościej sobie to wyobrazić, że numerem rewizji jest po prostu numerem kolejnego obrazu repozytorium.

4 Operacje na repozytorium

Operacje na repozytorium

- utworzenie repozytorium
- import danych
- pobranie danych z repozytorium
- aktualizacja kopii roboczej
- rozwiązywanie konfliktów
- zatwierdzanie zmian
- odrzucanie zmian w kopii roboczej
- zmiany w strukturze katalogów
- historia zmian
- trwałe usuwanie pliku z repozytorium

W tej części prezentacji zostaną zaprezentowane metody posługiwania się repozytorium przy użyciu odpowiednich narzędzi. Prezentacja odbędzie się przy użyciu klienta tekstowego (svn) oraz graficznego (TortoiseSVN).

Obaj klienci mogą byd stosowaniu równolegle do pracy na repozytorium i dysponują możliwością dostępu do wszystkich możliwości Subversion. Jednakże większy nacisk zostanie położony na klienta graficznego, ponieważ dla większości osób będzie on bardziej intuicyjny.

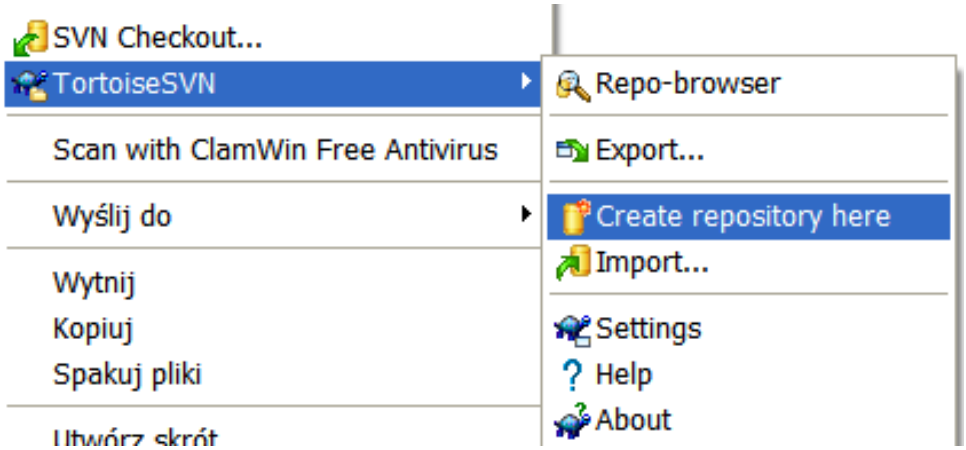
4.1 Tworzenie repozytorium

Utworzenie repozytorium

Linia poleceń

```
svnadmin create NoweRepozytorium
```

TortoiseSVN



Zarówno klient tekstowy jak i graficzny umożliwiają utworzenie repozytorium. Dostępne są dwa typy repozytoriów:

- oparte o bazę danych BerkleyDB
- oparte o system plików

W przykładzie będziemy zakładali repozytorium lokalne. Należy pamiętać, że jeżeli ma z niego korzystać więcej niż jedna osoba, to musi ono zostać udostępnione. Jak to zrobić pozostaje jednak poza zakresem tej prezentacji.

W podanym przykładzie zostanie utworzone repozytorium w katalogu `NoweRepozytorium`.

W podanym przykładzie dostęp do repozytorium może byd zdefiniowany następująco:

```
file:///SVN/NoweRepozytorium
```

Budowa repozytorium

Zalecany podział repozytorium na katalogi:

`trunk` aktualna wersja danych (np. bieżący rozwój projektu)

`branches` poszczególne odgałęzienia projektu

`tags` oznakowane wersje

Jeżeli posiadamy kilka projektów, musimy rozstrzygnąć czy będą one trzymane w jednym repozytorium czy też w oddzielnych, oraz w jaki sposób zorganizujemy podział katalogów (`Projekt/trunk` czy `trunk/Projekt`).

Utworzenie katalogu za pomocą linii poleceń:

```
svn mkdir
```

W TortoiseSVN wybiera się odpowiednią pozycję podczas przeglądania repozytorium.

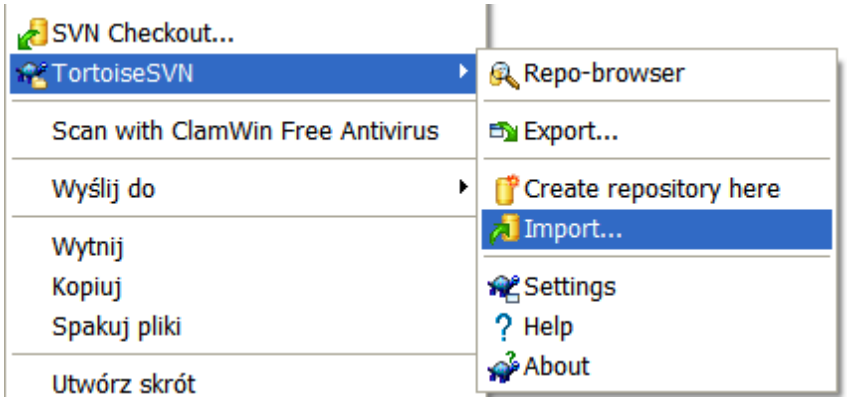
4.2 Aktualizacja repozytorium i kopii roboczej

Import danych do repozytorium

Linia poleceń

```
svn import PROJEKT file:///NoweRepozytorium/trunk \
-m "Import początkowy"
```

TortoiseSVN



Po utworzeniu repozytorium możemy zaimportować do niego aktualny projekt. Robimy to używając polecenia:

```
svn import
```

z linii poleceń. Parametr `-m` pozwala na podanie komentarza aktualnej wersji.

Jeżeli używamy aplikacji TortoiseSVN wybieramy opcję Import z menu. Zostanie wyświetlone okienko dialogowe, które pozwoli na wybranie domyślnego repozytorium jak i możliwość wpisania komentarza.

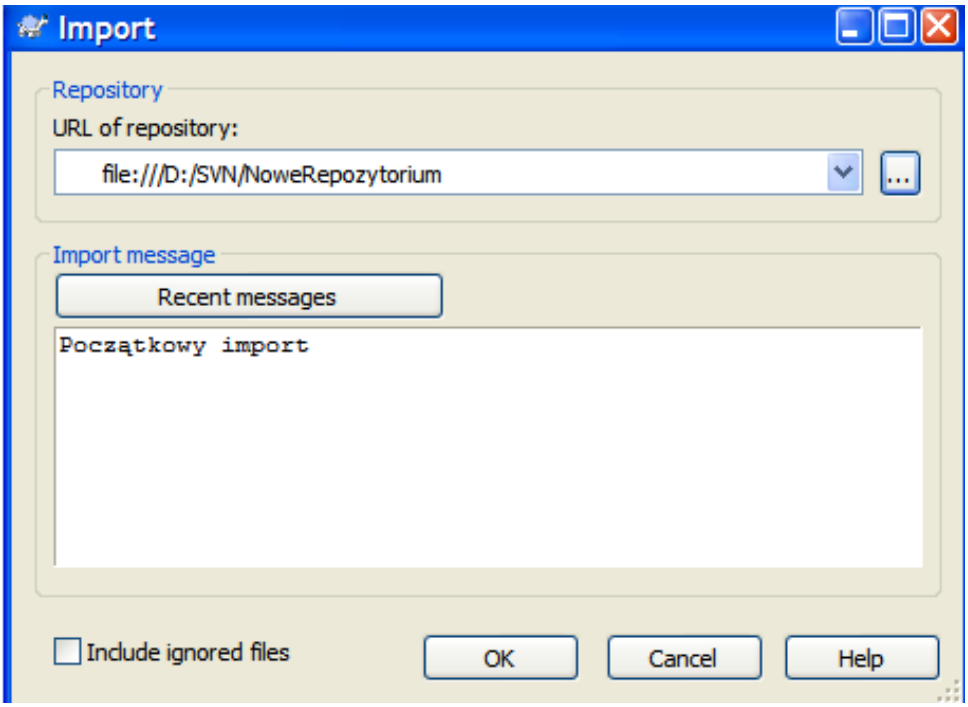
Po zaimportowaniu plików do repozytorium nie posiadamy ciągle lokalnej kopii repozytorium. Podane polecenie tylko zapisało plik w repozytorium. Aby móc korzystać z danych należy pobrać dane z repozytorium.

Pobranie danych z repozytorium

Linia poleceń

```
svn checkout file:///NoweRepozytorium/trunk
```

TortoiseSVN



Pierwsze pobranie danych z repozytorium odbywa się za pomocą komendy:

```
svn checkout
```

Jako parametr podajemy ścieżkę dostępu do repozytorium wraz z katalogiem wewnątrz repozytorium który ma zostać pobrany.

Używając TortoiseSVN wybieramy opcję *Checkout*. Zostanie wyświetlone podane okienko dialogowe, w którym możemy zdefiniować:

URL of repository – lokalizacja repozytorium (wraz z katalogiem który ma zostać pobrany)

Checkout directory – katalog w którym ma zostać utworzona lokalna kopia

Checkout Depth – ile poziomów katalogów powinno zostać pobrane

Omit externals – nie pobiera połączonych repozytoriów

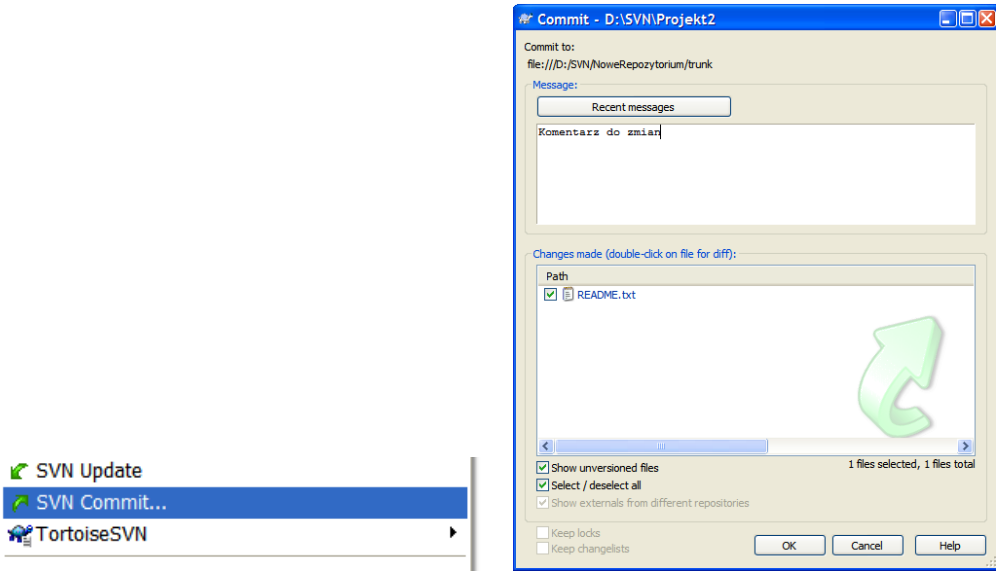
Revision – pozwala na określenie, czy chcemy pobrać aktualną wersję repozytorium (HEAD) lub też wybrać dowolną inną

Zatwierdzanie zmian

Linia poleceń

```
svn commit -m "Komentarz do zmian"
```

TortoiseSVN



Aby zatwierdzić wprowadzone zmiany przy użyciu linii poleceń należy wydać komendę:

```
svn commit
```

Opcjonalnie można dodać komentarz do wprowadzonych zmian.

Zatwierdzenie zmian z TortoiseSVN odbywa się poprzez wybranie odpowiedniej pozycji z menu (*SVN commit...*).

Zostanie wyświetlone okienko dialogowe, które pozwoli na podanie komentarza i wybranie plików jakie mają zostać wysłane do repozytorium. Okienko to pozwala także na wyświetlenie i dodanie plików które nie zostały jeszcze umieszczone w repozytorium.

Możliwe jest także wyświetlenie okienka pokazującego różnice pomiędzy plikiem aktualnie znajdującym się w repozytorium a jego nową wersją. Pozwala to w łatwy sposób sprawdzić co zostało zmienione.

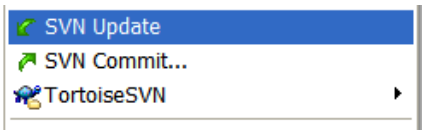
Po zatwierdzeniu zmian zostanie wyświetlone okienko z postępem przesyłania informacji do SVN.

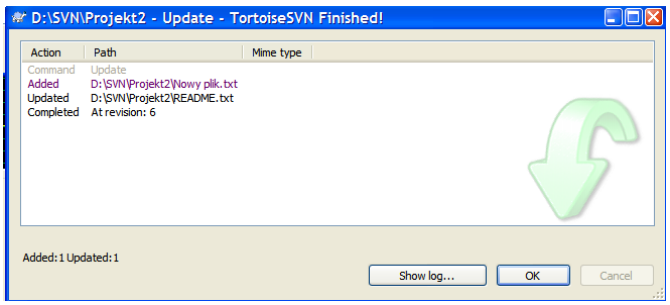
Aktualizacja kopii roboczej

Linia poleceń

`svn update`

TortoiseSVN





Aktualizacja kopii roboczej jest jedną z najczęściej wykonywanych operacji w każdym systemie kontroli wersji. Pozwala ona na zachowaniu aktualnej kopii roboczej repozytorium. Także w trakcie tego procesu dochodzi do najbardziej skomplikowanych operacji (jak konflikt wersji, automatyczne scalanie plików).

Aby dokonać aktualizacji z linii poleceń, należy wydać komendę (będąc w katalogu który chcemy zaktualizować):

```
svn update
```

Nastąpi połączenie z serwerem oraz pobranie najnowszej wersji repozytorium oraz jej zapisanie.

W trakcie tej operacji zostaną wyświetlone aktualizowane pliki i informacje o potencjalnych lub faktycznych problemach (szczegółowe informacje są dostępne po wpisaniu `svn help update`).

Aktualizacja przy pomocy TortoiseSVN jest możliwa po wybranie odpowiedniej komendy z menu. Podczas procesu aktualizacji zostanie wyświetlone okienko dialogowe, wewnątrz które będą wyświetlane wszystkie pliki które podlegają modyfikacji. W oknie dialogowym różnymi kolorami są oznaczone poszczególne operacje, jakie zostały wykonane na plikach podczas aktualizacji repozytorium:

- purpurowy – nowe pliki dodane do kopii roboczej
- zielony – zmiany z repozytorium które zostały automatycznie i pomyślnie scalone ze zmienionymi plikami w kopii roboczej
- jasno czerwony – zmiany z repozytorium których nie udało się scalić z kopią roboczą, wymagają ręcznej interwencji
- ciemno czerwony – brakujące pliki zmienione w repozytorium
- czarny – zaktualizowane pliki z repozytorium

Zmiany w strukturze plików i katalogów

Różne operacje na systemie plików należy wykonywać przy użyciu odpowiednich komend Subversion:

- `svn add`
- `svn delete`
- `svn copy`
- `svn move`
- `svn mkdir`

Ponieważ SVN śledzi zmiany w strukturze plików, musi zostać poinformowany że plik został przeniesiony w inne miejsce, została mu zmieniona nazwa. Jeżeli przeniesiemy plik w zwykły sposób (za pośrednictwem standardowych narzędzi), SVN nie będzie miał o tym pojęcia. Potraktuje wtedy zmianę nazwy jako dwie oddzielne operację: usunięcie pliku oraz dodanie nowego pliku, a co za tym idzie zostanie utracona historia zmian.

TortoiseSVN potrafi wykryć proste sytuacje dotyczące zmiany w strukturze katalogów (usunięcie pliku, pojawienie się nowego pliku lub katalogu) i automatycznie potrafi wprowadzać odpowiednie zmiany w repozytorium. Ale w dalszym ciągu nie wie nic o kopiowaniu lub przenoszeniu (zmianie nazwy) pliku.

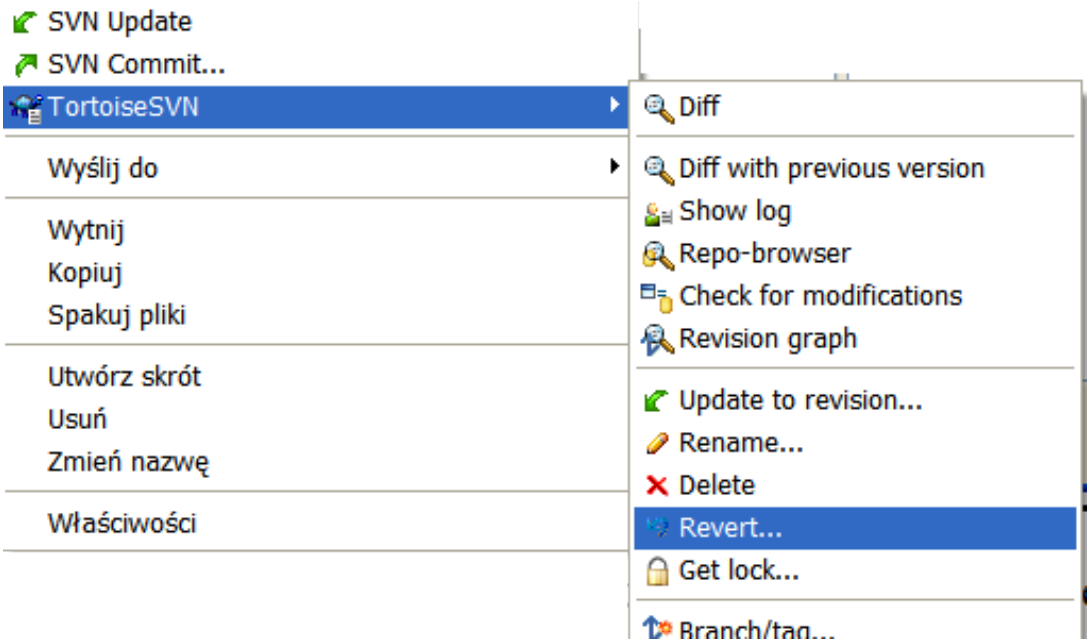
4.3 Usuwanie zmian

Usuwanie wprowadzonych zmian

Linia poleceń

`svn revert`

TortoiseSVN



Czasem może się zdarzyć, że chcemy przywrócić pliki do wersji pobranej z repozytorium. Służy do tego specjalna funkcji SVN: `revert`.

W TortoiseSVN wybieramy odpowiednią opcją.

Usuwanie wprowadzonych zmian umożliwia łatwe przywrócenie stanu repozytorium w przypadku gdy wprowadzone zmiany okazały się błędne lub też nastąpiło popsucie pliku. Można przywracać poprzednią wersję całemu katalogowi roboczemu jak i tez pojedynczym plikom.

4.4 Rozwiązywanie konfliktów

Rozwiązywanie konfliktów Rodzaje konfliktów

Podczas pracy z systemem kontroli wersji można spotkać dwa rodzaje konfliktów:

- konflikty związane z pojedynczymi plikami – konflikt ten pojawia się gdy dwaj użytkownicy dokonali zmian w tych samych liniach pliku
- konflikty związane z drzewem katalogów – konflikt ten pojawia się w sytuacji gdy dwóch użytkowników zmodyfikuje ten sam element drzewa katalogów

Konflikty związane z plikami Zmiany w wewnątrz pliku

W przypadku wystąpienia tego rodzaju konfliktu w pliku źródłowym można znaleźć następujące linie:

```
<<<<<<< nazwa pliku
twoje zmiany
=======
kod pobrany z repozytorium
>>>>>>> nr rewizji
```

Można teraz tak zmodyfikować plik, aby najlepiej odzwierciedlał stan faktyczny.

Konflikty związane z plikami Zmiany systemie plików

Dodatkowo w momencie wystąpienia konfliktu są tworzone pliki:

- `nazwa_pliku.mine` – plik, który znajdował się w lokalnym repozytorium, czyli oryginalny plik użytkownika, bez żadnych dodatkowych modyfikacji i oznaczeń
- `nazwa_pliku.rPOPRZEDNIA_REWIZJA` – oryginalny plik, w stosunku do którego użytkownik wprowadził zmiany
- `nazwa_pliku.rNOWA_REWIZJA` – aktualny plik znajdujący się w repozytorium

Konflikty związane z plikami Rozwiązywanie konfliktów

Po zakończeniu rozwiązywania konfliktu należy wykonać:

- polecenie `svn resolve`
- wybrać opcję *TortoiseSVN->Resolved*

Spowoduje to usunięcie tych 3 dodatkowych plików i oznaczany konflikt jako rozwiązany.

Konflikty związane z drzewem katalogów

- lokalne usunięcie pliku, w repozytorium jest zapisana nowa wersja pliku
- lokalna edycja pliku usuniętego już w repozytorium
- lokalne usunięcie pliku, w repozytorium plik także zostaje wcześniej usunięty
- lokalnie brak pliku, ale plik w repozytorium zostaje zmodyfikowany podczas scalania z inną gałęzią
- lokalna edycja pliku, ale plik zostaje usunięty podczas scalania z inną gałęzią kodu
- lokalne usunięcie pliku, plik w repozytorium usunięty podczas scalania z inną gałęzią kodu

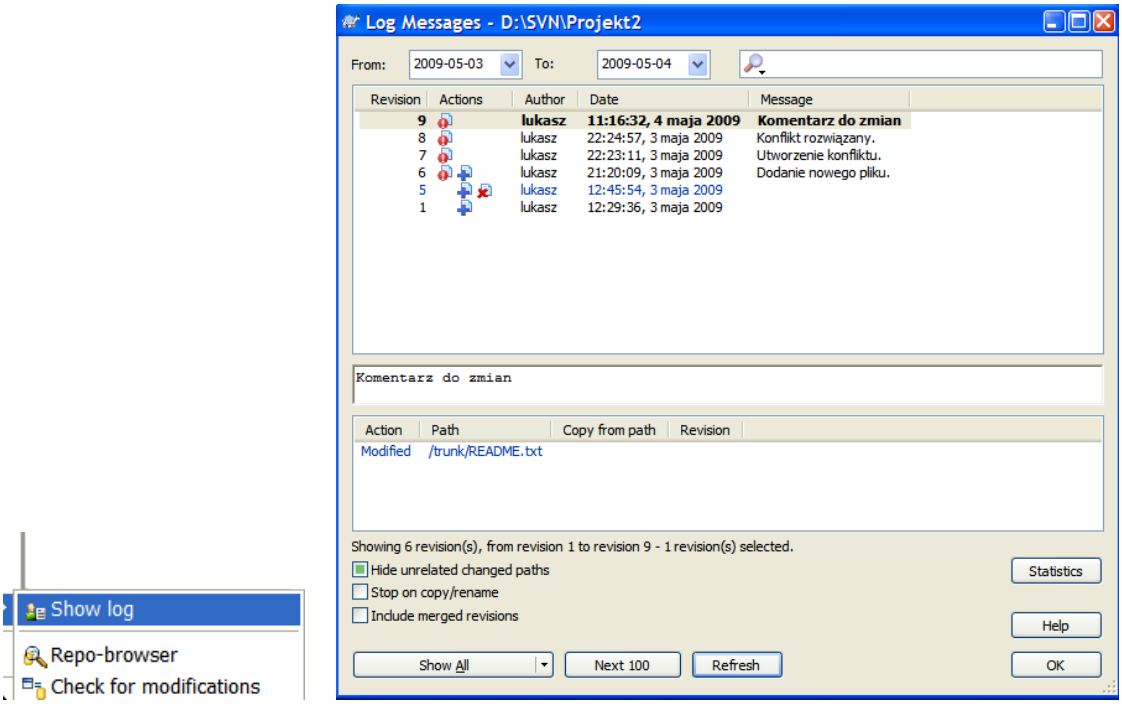
4.5 Historia zmian

Historia zmian

Linia poleceń

`svn log`

TortoiseSVN



4.6 Trwałe usuwanie danych z repozytorium

Trwałe usunięcie pliku z repozytorium

- odpowiedź krótka: *nie da się*
- w przyszłości:

  `svnadmin obliterate`
- odpowiedź długa:

  `svnadmin dump`
  `svndumpfilter`
  `svnadmin load`

Subversion od początku zostało zaprojektowane w taki sposób, aby przechowywać wszystkie informacje, które do niego zostaną zapisane.

Czasami jednak zdarza się, że zostanie zapisane coś, co nie powinno się tam znaleźć (np. lista haseł, katalog Windows). W takim przypadku należałoby usnąć te pliki z historii repozytorium. Niestety, nie jest to możliwe do osiągnięcia przy użyciu pojedynczej komendy. Dzieje się tak dlatego, że w SVN poszczególne rewizje są drzewiastą strukturą, która nie podlega zmianom i każda następna rewizja jest budowana na poprzedniej. Dlatego zmiana jednej rewizji z przeszłości pociąga za osobą efekt kuli śnieżnej, i wymaga wprowadzenia zmian we wszystkich następnych rewizjach.

Tutaj znajduje się odpowiednie zgłoszenie w systemie śledzenia błędów SVN: http://subversion.tigris.org/issues/show_bug.cgi?id=516. Jest ono otwarte od prawie 8 lat i ciągle nie wiadomo kiedy zostanie zrealizowane...

Ale wiadomo, że polecenie będzie się nazywało `svnadmin obliterate`.

Istnieje możliwość usunięcie pliku z repozytorium, ale wiąże się to z koniecznością przebudowania całego repozytorium, co w przypadku dużych repozytoriów może być bardzo kosztowną operacją. W skrócie:

- należy zapisać całe repozytorium w postaci pliku tekstowego (za pomocą polecenia `svnadmin dump`)
- za pomocą polecenia `svndumpfilter` odfiltrować zbędne pliki
- ponownie utworzyć repozytorium za pomocą polecenia textttsvnadmin load

5 Tematy zaawansowane

Tematy zaawansowane

- gałęzie i tagi
- scalanie gałęzi
- właściwości plików i katalogów
- wykonywanie skryptów podczas aktualizacji repozytorium

6 Klienci Subversion

Klienci Subversion

- natywny (linia poleceń)
- TortoiseSVN (Windows)
- Subversive / Subclipse (wtyczka do Eclipsa)
- AnkhSVN (wtyczka do Visual Studio)
- KSvn (KDE)
- i wiele innych

Koniec

Dziękuję za uwagę