Inżynieria oprogramowania II

Wykład 4:

"UP: Zarządzanie konfiguracją i zmianami
(ang. *Configuration and Change Management*)"

Marek Krętowski
pokój 206
e-mail: mkret@ii.pb.bialystok.pl
http://aragorn.pb.bialystok.pl/~mkret

Wprowadzenie

- Tworzenie oprogramowania jest złożonym zadaniem, który obejmuje wiele niesekwencyjnych działań (proces iteracyjny i przyrostowy)
- Ostatecznym celem jest stworzenie **wykonywalnego programu**, który spełnia wymagania klienta
- Aby osiągnąć ten cel tworzone są różnorodne artefakty (opisują projekt w w terminach specyfikacji, modelują system, definiują poszczególne komponenty, opisują interfejsy, dokumentują status projektu, zapisują błędy); wszystkie te artefakty ewoluują wraz z postępem prac
- Jest wiele powodów uzasadniających istnienie wielu konfiguracji systemów
  - różne komputery
  - różne systemy operacyjne
  - specyficzne wersje dla konkretnych klientów
  - ...
- Same artefakty jak i ich zmiany muszą być odpowiednio zarządzane aby zapewnić stałą kontrolę nad produktem jak i nad procesem wytwórczym
- Czynności niezbędne do zarządzania i kontroli artefaktów i ich zmian tworzą dziedzinę zarządzania konfiguracją i zamianami (ang. *software configuration and change management* - SCCM)

Dynamika tworzenia oprogramowania

- Różne role wpływają na tworzony system (klienci proponują nowe specyfikacje; menadżerowie zmieniają harmonogram i czasami strukturę artefaktów; projektanci dzięki lepszemu zrozumieniu problemu proponują zmiany algorytmów; architekci systemowi znajdują lepsze sposoby budowy sytemu; programiści dodają szczegóły do produktu oraz zmieniają implementowane komponenty; testerzy opracowują nowe przypadki testowe)
- Wszystkie powyższe zmiany (zachodzące mniej lub bardziej równocześnie i asynchronicznie) są typowe dla przebiegu procesu i celem nie powinno być ich eliminowanie czy ograniczanie lecz raczej ich uporządkowane i kontrola
- SCCM zapewnia techniki, metody i procedury:
  - do utrzymywania historii produktu, w której każda wersja jest unikalnie zidentyfikowana i przechowywana
  - do inicjowania, oceny i kontroli zmian produktu zarówno podczas rozwoju jak i po dostarczeniu (przekazaniu) systemu

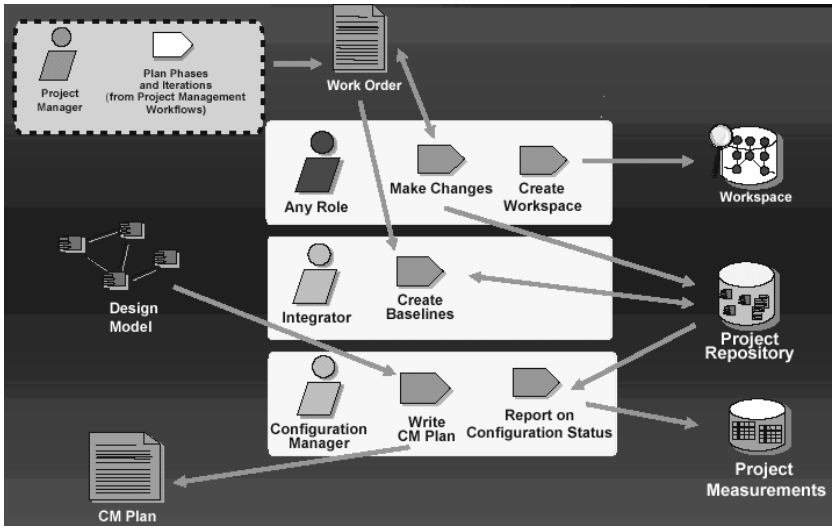
Kluczowe czynności wg. Sommerville'a

- Planowanie zarządzania konfiguracją
  - identyfikacja elementów konfiguracji (przypisanie unikalnej nazwy artefaktom; hierarchiczne systemy nazewnictwa =>często mapowanie w struktury katalogów)
  - baza elementów konfiguracji - wszystkie istotne informacje o konfiguracjach i ich elementach (także np. informacje o wykorzystaniu - klienci; zależnościach, proponowanych zmianach, ...) najlepiej zintegrowana z systemem zarządzania wersjami
- Zarządzanie zmianami
  - zwykle od pierwszej wersji odniesienia; CRF (ang. change request form)
- Zarządzanie wersjami i wydaniami
  - **wersja** (ang. version) - wyróżniająca się od innych instancji systemu np. inną funkcjonalnością, poprawioną efektywnością czy usuniętymi defektami; może też być inna platforma docelowa; wykorzystywana wewnętrznie w firmie
  - **wydanie** (ang. release) - wersja systemu, która jest przekazywana użytkownikom;
- Budowa systemu
  - kompilacja i linkowanie programu do wykonania w docelowej konfiguracji

Cele i korzyści

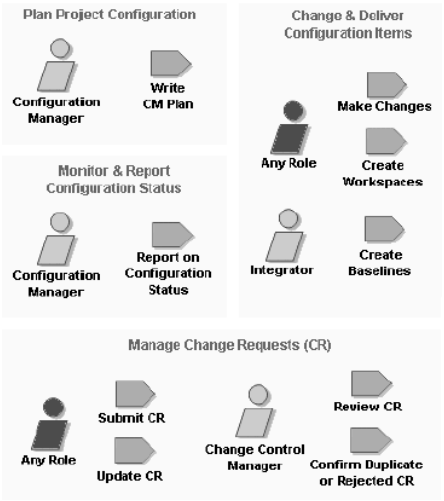
- Bezpośrednie korzyści:
  - wspieranie metod wytwórczych
  - utrzymywanie integralności produktu
  - zapewnianie kompletności i poprawności skonfigurowanego produktu
  - dostarczenie stabilnego środowiska w którym rozwijany jest produkt
  - zapewnienie, że wprowadzane zmiany są zgodne z przyjętym postępowaniem
  - zbieranie i przechowywanie historycznych informacji kontrolnych pokazujących dlaczego, kto i kiedy modyfikował poszczególne artefakty
  - obejmuje również zbieranie szczegółowej informacji o samym procesie (kto i kiedy tworzył konkretną wersję, które wersje plików źródłowych zostały wykorzystane w poszczególnych build-ach)
- Podstawowe cele:
  - zapewnienie integralności produktu
  - doprowadzenie do tego, aby ewolucja produktu była zarządzalna

Czynności procesu



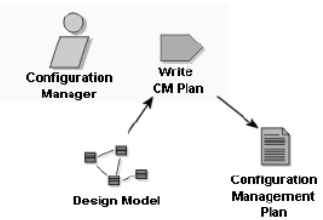
Czynności procesu (2)

- Planowanie konfiguracji projektu
  - pisanie planu zarządzania konfiguracją (ang. *Write CM Plan*)
- Monitorowanie stanu konfiguracji
  - raportowanie o stanie konfiguracji (ang. *Report on Configuration Status*)
- Modyfikowanie i dostarczanie elementów konfiguracji
  - przeprowadzanie zmian (ang. Make Changes)
  - tworzenie przestrzeni roboczych (ang. Create Workspaces)
  - tworzenie wersji odniesienia (ang. Create Baselines)
- Zarządzanie wnioskami zmian
  - składanie, uaktualniane, recenzowanie i rozpatrywanie wniosków



Planowanie konfiguracji projektu
(ang. *Plan Project Configuration*)

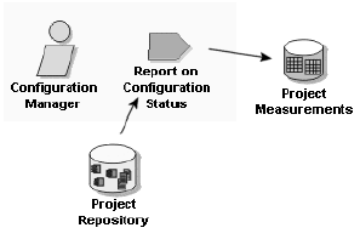
- Celem stworzenia planu zarządzania konfiguracją jest zdefiniowanie kroków i konkretnych działań niezbędnych do zarządzania konfiguracją
- Plan opisuje zakresy odpowiedzialności oraz środowisko w którym wykonywane będą funkcje zarządzania konf.



- Definiuje szczegóły identyfikacji artefaktów i strukturę wersji odniesienia (ang. *baseline*), opisuje proces występowania o zmiany, ich kontrolę, zbieranie informacji o stanie konfiguracji oraz audyty konfiguracji
- Pisany jest na samym początku cyklu życia oprogramowania, zaraz po tym gdy zaakceptowane zostanie finansowanie projektu; plan może być uaktualniany na początku każdej iteracji

Monitorowanie stanu konfiguracji
(ang. *Monitor & Report on Configuration Status*)

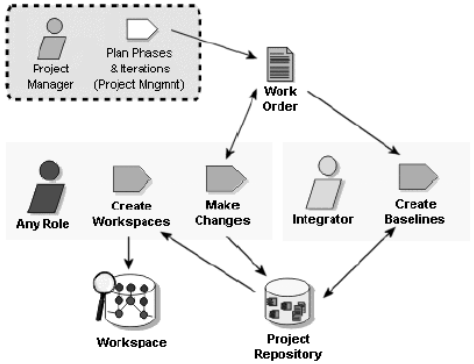
- Zbadanie czy produkt spełnia funkcjonalne jak i fizyczne wymagania, czy wymagane artefakty są odpowiednio przechowywane i modyfikowane w oparciu o wersje odniesienia (ang. *baselines*) oraz czy są dostępne



- Celem jest zdefiniowanie, które związane z produktem dane dotyczące zmian (ang. change data) będą raportowane, przez kogo i z jaką częstotliwością
- Metryki tworzone w tym aspekcie zarządzania konf. są często użyteczne do określenia całościowej kompletności projektu

Modyfikowanie elementów konfiguracji
(ang. *Change and Deliver Configuration Items*)

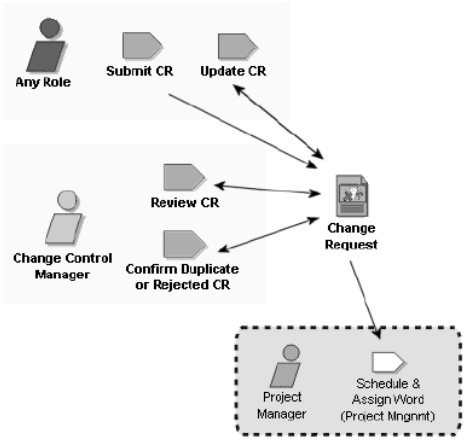
- Tworzenie wersji odniesienia - celem jest zapewnienie wychwycenia i zarchiwizowanie wszystkich opracowanych artefaktów w określonych momentach czasu, tworząc w ten sposób bazę do dalszych prac
- Przeprowadzanie zmian - umożliwia uzyskanie przez wytwórce konkretnej wersji elementu konfiguracji, zmodyfikowanie go a następnie opublikowanie (udostępnienie innym) zmienionej wersji (ang. *promote check-in and checkout*); zdefiniowany protokół postępowania



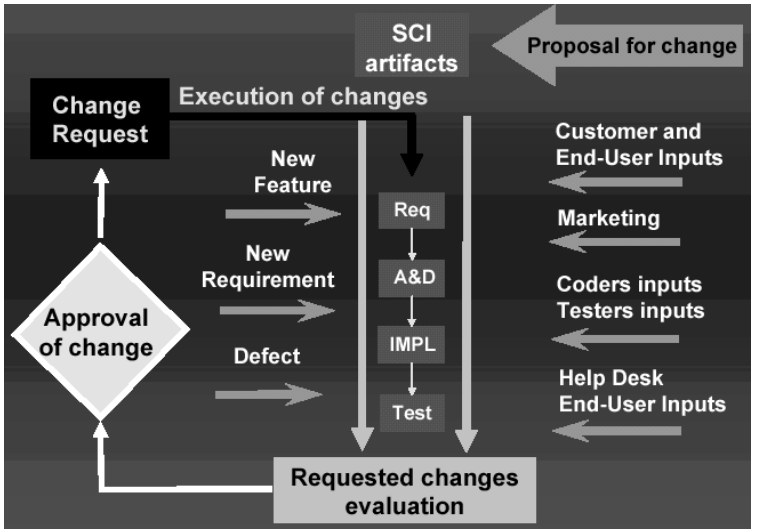
- Tworzenie przestrzeni roboczej - zdefiniowanie prywatnego środowiska wytwórczego, w którym członek zespołu jest odizolowany od przestrzeni pozostałych członków; przestrzeń musi umożliwiać zachowanie standardów projektowych (późniejsza integracja)

Zarządzanie wnioskami zmian
(ang. *Manage Change Requests*)

- Wprowadzenie standaryzowanego, udokumentowanego procesu zarządzania wnioskami zmian zapewnia, że zmiany wprowadzane są w sposób spójny i odpowiedni członkowie zespołu są informowani o aktualnym stanie, wprowadzanych zmianach, ich koszcie i wpływie na harmonogram
- W dużych projektach może wymagać powołania specjalnego zespołu do oceny wniosków (ang. Change Control Board), składającego się z przedstawicieli wszystkich zainteresowanych grup (klienci, wytwórcy, użytkownicy)
- Zaakceptowanym wnioskom przypisywane są priorytety i kierowane są one do realizacji



Zmiany dotyczące produktu są kontrolowane



Operacyjne aspekty zarządzania konfiguracją i zmianami

- Różne oczekiwania wobec zarządzania konfiguracją i zmianami:
  - menadżer (mechanizmy pozwalające na podejmowanie decyzji o statusie komponentów)
  - programista (wersje odniesienia - baselines)
  - *environment builder* (biblioteki i ponownie wykorzystywane komponenty)
  - użytkownik końcowy (wersje produktu)
- Standardowa definicja zarządzania konfiguracją i zmianami zawarta w IEEE 828-1990 obejmuje:
  - rozpoznawanie elementów konfiguracji (ang. *identification of software configuration items*)
  - kontrolę linii odniesienia oraz zmian (ang. *control of baseline and change*)
  - "monitorowanie" stanu (ang. *status accounting*)
  - przeglądy (ang. *audit and review*)

Rozpoznawanie elementów konfiguracji oprogramowania

- W celu zapewnienia integralności produktu, odpowiednie elementy (artefakty) są wybierane i poddawane kontroli konfiguracji (ang. *software configuration items*)
  - dokumenty wymagań, specyfikacje, dokumenty projektowe, kod, zestawy testów, podręczniki użytkownika i administratora, plany projektu i testów, harmonogramy, dokumenty proceduralne oraz wersje narzędzi
- Objęcie elementu kontrolą konfiguracji oznacza, że musi on zostać zachowany w miejscu łatwo dostępnym, w formie niezmienialnej, jednoznacznie nazwany w sposób umożliwiający łatwe odróżnienie wersji
- *Baseline* - wersja odniesienia, określa zbiór konkretnych elementów konfiguracji odnoszący się do stabilnej wersji produktu, w stosunku do której zgłaszane są błędy i zgłaszane wnioski zmian;
  - wykorzystywana jako wspólny punkt odniesienia w dalszych etapach prac aż do określenia kolejnej baseline; formalne (autoryzacja zmian) lub nieformalne (wspólne repozytorium) wersje odniesienia

Kontrola wersji odniesienia i zmian

- Kluczowa jest koordynacja równoległego działania wielu osób zaangażowanych w proces wytwórczy
- Kontrola ma na celu uniknięcie pomyłek i zamieszania, gdy istnieje wiele potencjalnych źródeł modyfikacji (różne zespoły, miejsca, iteracje, warianty, wersje, platformy, ...)
- Kontrola wymaga różnej funkcjonalności:
  - fizyczna (ograniczony dostęp do komponentów systemu)
  - procesowa (wsparcie online dla wypełniania wniosków zmian i raportowania problemów; umożliwienie śledzenia losów zmian (kto, kiedy, jak) przez menadżerów; zespół wykonawczy może mieć zapewnioną propagację zmian we wszystkich zależnych wersjach produktu; dzielenie produktu pozwala eliminować wpływ zmian)
- Baselines wykorzystywane mogą być do kontroli struktury produktu i jego wariantów (system zainstalowany, skonfigurowany, opakowany specyficznie dla danego klienta)

Zarządzanie wersjami i wydaniami

- Identyfikacja wersji:
  - numerowanie wersji - numer wersji jest dodawany do nazwy komponentu lub systemu; 1.0 pierwsza wersja, kolejne wersje 1.1, .1.2; kolejne wydanie 2.0
  - bazująca na atrybutach - przypisywanie charakterystycznych pól do każdej wersji; np. klient, język, stan, platforma docelowa, data utworzenia
  - bazująca na zmianach (ang. change-oriented)
- Zarządzanie wydaniami
  - Wydanie: program wykonywalny, pliki konfiguracyjne, dane, program instalacyjny, dokumentacja (elektroniczna i papierowa), opakowanie, reklama
  - przygotowanie na różne scenariusze zachowań klientów
  - strategie wydań (częstotliwość, zakres, cena, ...)
  - opublikowanie (przygotowanie) wydania (zebranie wszystkich niezbędnych elementów i przygotowanie do dystrybucji oraz jej przeprowadzenie)
  - dokumentowanie wydania (zapewnienie dokładnego odtworzenia wydania w przyszłości)

Monitorowanie stanu

- Trzy podstawowe kroki dotyczące stanu (statusu)
  - Przyjmowanie wniosków o zmiany
  - Raportowanie statusu komponentów i wniosków zmian
  - Zbieranie istotnych statystyk o komponentach produktu
- Monitorowanie stanu jest znacząco ułatwione, gdy odpowiednie pomiary mogą być w sposób automatyczny uzyskiwane ze środowiska wytwórczego
- Źródła stanu konfiguracji oprogramowania:
  - wnioski zmian (ang. *change request*)
  - raporty (ang. build reports)
  - opisy wersji (ang. *release notes*)
- Los wniosków zmian może być raportowany w różnej kontekstach (czas, ilość, częstotliwość, trend, ...)

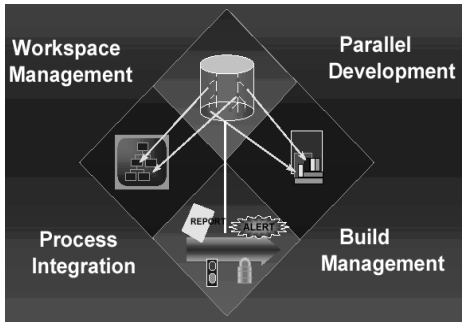
Sprawdzanie funkcjonalności i fizycznej konfiguracji

- Kontrola fukcjonalności (ang. *functionality auditing*) - celem jest sprawdzenie czy działanie elementów konfiguracji jest zgodne z wymaganiami
  - przygotowanie macierzy weryfikacji, która zawiera wszystkie wymagania funkcjonalne wraz z odpowiadającymi im rezultatami testu i/lub analizy i/lub raportu demonstracji, które w sposób kompletny dokumentują walidację odpowiadających wymagań
  - sprawdzenie, czy wszystkie wnioski zmian zostały odpowiednio zaimplementowane
  - dokumentowanie rozbieżności i ustanowienie akcji naprawczych wraz z datami wykonania
- Kontrola fizycznej konfiguracji (ang. *physical configuration auditing*) - celem jest zweryfikowanie, że artefakty w ramach wersji odniesienia mają odpowiednią wersję
  - stworzenie listy elementów, które powinny znajdować się pod kontrolą konfig.
  - inspekcja elementów podlegających kontroli konfiguracji
  - stworzenie listy rozbieżności pomiędzy tym co powinno a co podlega kontroli

Wyzwania podczas wprowadzania zarządzania konfiguracją i zmianami

- Czynniki powodujące (wymuszające) stosowanie zarządzania konfiguracją i zmianami z różnym poziomem automatyzacji, m.in. :
  - formalne wymogi podczas realizacji zleceń rządowych
  - dostępność narzędzi wspomagających
  - wystąpienie potrzeby zarządzania konfiguracją
  - konieczność usprawnienia wykorzystywanego systemu zarządzania konfiguracją
- Wprowadzenie mechanizmów zarządzania musi brać pod uwagę zdolność organizacji do przystosowania się pod różnymi względami
  - zarządczym (zakup systemu zarządzania czy opracowanie własnego narzędzia; porównanie zalet i kosztów obu rozwiązań; akceptacja przez zespoły projektowe; ustalenie zakresu kontroli i stopnia automatyzacji, ...)
  - technologicznym (platformy, narzędzia) (dostępna technologia nie zawsze jest wystarczająca do bezpośredniego rozwiązania wielu podstawowych nawet problemów zarządzania konf.; niezbędne dostosowanie do specyficznych potrzeb oraz przeszkolenie personelu; problemy ze spójnością pomiędzy różnymi narzędziami)
  - procesowym

Narzędzia CASE wspomagające zarządzanie konfiguracją



System zarządzania konfiguracją i zmianami powinien wspierać wszystkie role i zadania

- Zintegrowane środowiska
  - Unified Change Managment (ClearCase - budowanie systemu i zarządzanie wersjami + ClearQuest - śledzenie zmian)
- Wykorzystanie grupy narzędzi do różnych celów
  - zarządzanie zmianami wspierane przez system śledzenie błędów np. Bugzilla
  - zarządzanie wersjami RCS, CVS, Subversion ...
  - budowanie systemów: make, imake, ...

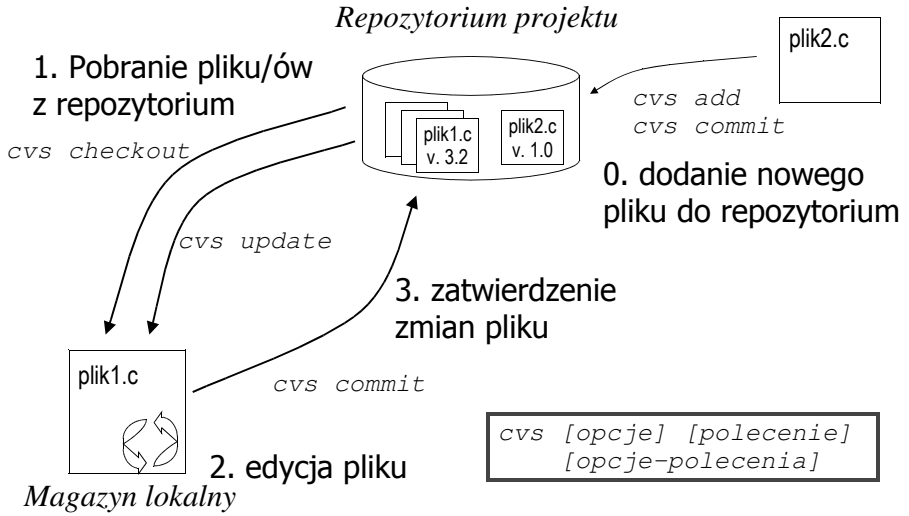
Systemy kontroli wersji

- System kontroli wersji zawiera zapisy zmian w plikach w trakcie procesu ich rozwoju, umożliwia odtworzenia dowolnej przechowywanej wersji (ang. revision) pliku oraz wspomaga produkcję wielu różnych wersji
- System przechowuje wszelkie dane historyczne na temat plików źródłowych w centralnym **repozytorium** (ang. repository)
- Przykłady:
  - RCS -Revision Control System - śledzenie zmian w jednym pliku
  - **CVS** - Concurrent Version System
  - Subversion
  - MS Visual Source Safe
- Historia każdego pliku jest przechowywana w formie wersji umożliwiających identyfikację, zwykle przy użyciu automatycznie generowanych numerów wersji (n.p. 2.5.15) i często znaczników (np. R3.5-STABLE-1) wybranych przez programistów w celu oznaczenia kluczowych etapów procesu rozwoju projektu

Systemy kontroli wersji (2)

- Kiedy programista kończy pracę nad wersją pliku, **zatwierdza** (ang. commit) lub **wpisuje** (ang. check in) plik w repozytorium
- System kontroli wersji przechowuje szczegółowe informacje o wszystkich wersjach pliku (często w celu oszczędzania przestrzeni dyskowej zapisywane są tylko różnice pomiędzy wersjami) + metadane dotyczące każdej operacji zatwierdzenia (np. data, czas, nazwisko, komentarz)
- Niektóre systemy umożliwiają programistom blokowanie (ang. lock) plików, kiedy nad nimi pracują, co zapobiega równoległemu wprowadzaniu zmian
- Inne systemy oferują sposób rozstrzygania konfliktowych zmian w razie ich wystąpienia
- **Rozdzielenie** (ang. split) prac rozwojowych nad plikiem na dwa różne odgałęzienia (ang. branches), np. na główną gałąź i odgałęzienie konserwacyjne
- Gałęzie mogą być **połączone** (ang. joined)
- W celu koordynacji publikowania pełnych systemów składających się z wielu plików, często zapewnia się sposób **znacznikowania** (ang. tag) zbiorów plików przy użyciu nazwy symbolicznej identyfikującej wersję systemu

Schemat pracy z CVS




```
cvs [opcje] [polecenie]
    [opcje-polecenia]
```


Subversion

- Powstał w celu poprawienia/zastąpienia CVS.
- Funkcjonalnie jest w większości zgodny z CVS. Z kompatybilności zrezygnowano tylko tam gdzie było to konieczne.

Zmiany w stosunku do CVS:

- zapis repozytorium z plikami wewnątrz bazy danych Berkley DB,
- zarządzanie zmianami całego drzewa plików i katalogów (możliwość przenoszenia plików i katalogów, zmian nazw, wspólny numer wersji dla całego drzewa),
- atomowe zapisy do repozytorium,
- szybkie tworzenie gałęzi i znaczników (poprzez proste kopiowanie)
- możliwość przyporządkowania do każdego pliku w repozytorium metadanych (atrybutów), wersjonowanie metadanych
- możliwość dostępu do repozytorium poprzez protokół HTTP (strona www repozytorium)

Subversion (2)

Sposób pracy z Subversion jest taki sam jak z CVS. Można wyróżnić 4 główne kroki:

- pobranie kopii roboczej (ang. checkout)
- edycja plików, katalogów
- pobranie nowszej wersji i scalenie zmian (ang. update)
- zapisanie zmian w repozytorium (ang. commit)

Korzystanie z repozytorium

svn [komenda] [opcje]

np. svn checkout file:///path/to/repos

Subversion w przeciwieństwie do CVS nie wersjonuje każdego pliku osobno, ale całe drzewo otrzymuje wspólny numer rewizji.