PODSTAWY INŻYNIERII OPROGRAMOWANIA

LABORATORIUM SYSTEMÓW KONTROLI WERSJI

System GIT - podstawy

przygotował:
dr inż. Radosław Adamus

HISTORIA WERSJI

DATA	WERSJA	AUTOR	OPIS
08.11.2013	0.1	Radosław Adamus	Pierwsza robocza wersja dokumentu
17.11.2013	0.2	Radosław Adamus	Uzupełnienie opisu, dodanie zadania “Praca zespołowa”, poprawki edycyjne, uzupełnienie informacji licencyjnych

Cel:

Celem laboratorium jest:
1. Zapoznanie się z funkcjonalnością podstawowych komend rozproszonego systemu kontroli wersji GIT oraz wprowadzenie do wykorzystania platformy GitHub.
2. Zrozumienie znaczenia kontroli wersji i potrzeby wykorzystywania systemu kontroli wersji w codziennej pracy.

Efekty:

Po ukończniu laboratorium student potrafi:
1. Inicjować repozytorium zdalne i lokalne, łączyć je ze soba oraz przesyłać informacje pomiędzy nimi.
2. Dodawać pliki do repozytorium i zatwierdzać w nim zmiany.
3. Tworzyć gałęzie, przełączać się pomiędzy nimi oraz scalać zmiany w nich wprowadzane.
4. Wykorzystać mechanizmy platformy GitHub do tworzenia wydań projektu oraz realizacji podstawowego scenariusza pracy zespołowej.

Wymagania wstępne:

Założone konto w systemie GitHub (https://github.com/).

Narzędzia:

Lokalnie zadania wykonywane są z wykorzystaniem linii poleceń systemu git oraz graficznego narzędzia TortoiseGit[1]. Zakłada się, że operacje wykonywane są z poziomu linii poleceń, natomiast narzędzie z interfejsem graficzne służy do wizualizacji grafu wersji (ang. revision graph).

Komendy, które na pewno wykorzystasz w trakcie zajęć:

- git add
- git checkout
- git clone
- git commit
- git config
- git branch
- git init
- git merge
- git push
- git rebase
- git remote
- git status

[1] Lub innego (TortoiseGit jest aplikacją systemu Windows), który potrafi wizualizować graf wersji.

Reguły wykonywania ćwiczeń laboratoryjnych:

1 Komentowanie zmian w trakcie ich zatwierdzania musi być znaczące(!). Wprowadzona informacja powinna określać czego dotyczy zmiana.
2. Repozytorium musi być tak skonfigurowane, aby zmiany były sygnowane nazwą użytkownika odpowiadającą imieniu i nazwisku studenta oraz jego adresem emailowym.
3. Rezultat musi znaleźć się ostatecznie na koncie studenta w systemie GitHub.
4. Historia zmian (wraz z komentarzami) będzie kontrolowana - musi odpowiadać strukturze zadań.
5. Podczas wykonywania zadań wyświetlaj strukturę grafu wersji (TortoiseGit -> Revision graph) i **zapisuj jego stany** dla poszczególnych zadań w postaci plików graficznych (File -> Save graph as).

Wstęp:

1. Czym jest kontrola wersji

Kontrola wersji (rewizji) to aspekt zarządzania konfiguracją (oprogramowania) związany z zarządzaniem zmianami w obrębie dokumentów elektronicznych.

2. Co to jest system kontroli wersji

System kontroli wersji to narzędzie oprogramowanie wspomagające proces kontroli wersji. Do jego funkcji należy przede wszystkim: zarządzanie składem (repozytorium) przechowującym wersje dokumentów oraz kontrolowanie dostępu do przechowywanych wersji zasobów. W kontekście kontroli wersji dostęp do zasobu może oznaczać pobranie określonej jego wersji, wprowadzenie nowej wersji czy też utworzenie nowej wersji poprzez połączenie zmian występujących w innych wersjach zasobu.

Zadania:

0. Uwagi wstępne

Struktura laboratorium wydziela problem kontroli wersji (który jest związany z laboratorium) od projektu podlegającego kontroli wersji (który jest tylko przykładem). Z tego powodu zmiany, które należy wprowadzić w projekcie opisane są w osobnych plikach.

1. Zdalna inicjalizacja

Utwórz repozytorium projektu w serwisie GitHub o nazwie iislabpio_[nr indeksu].

2. Lokalna inicjalizacja repozytorium projektu

Utwórz katalog projektu i dokonaj inicjalizacji repozytorium git w jego wnętrzu (git init).
Skonfiguruj lokalnie dla repozytorium swoją nazwę użytkownika (swoje imię i nazwisko) oraz adres email (git config).

Utwórz plik o nazwie ‘README.md’ i dodaj go do lokalnego repozytorium. Wyedytuj go w edytorze tekstowym i wpisz swój numer indeksu oraz grupę laboratoryjną. Zatwierdź (git commit) zmiany pamiętając o komentarzach.
Połącz lokalne repozytorium (domyślna gałąź master) z repozytorium zdalnym (git remote).
Wyślij (git push) do zdalnego repozytorium lokalną gałąź.

Struktura projektu i gałęzie robocze

Utwórz lokalnie plik *index.html* i dodaj do niego strukturę z pliku *init.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany pamiętając o komentarzach.
Utwórz gałęzie ‘cecha1’ i ‘cecha2’.
Przełącz się na gałąź *cecha1* i wprowadź zmiany zgodnie z opisem w pliku *zmiana1.txt*.
Przetestuj poprawność wprowadzonych zmian.
Zatwierdź (git commit) zmiany.

Scalanie zmian w gałęzi master

W trakcie testowania zorientowałeś się że podstawowa struktura projektu zawiera błąd. Informacje te powinny znaleźć się w głównej gałęzi projektu, zanim ukończone zostaną prace nad poszczególnymi cechami. W związku z tym należy w pierwszym rzędzie wprowadzić zmiany w gałęzi master. Aby uniknąć rozsynchronizowania gałęzi należy również scalić zmiany z gałęzi master z zawartością pozostałych gałęzi.

Przełącz się do gałęzi master i wprowadź zmianę zgodnie z opisem w pliku *zmiana2.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany.

Przełącz się do gałęzi *cecha1* i scal zmiany z gałęzi master (zapoznaj się z poleceniami merge oraz rebase - wybierz jedną z nich). Pamiętaj o zapisaniu obrazu grafu wersji (rewizji).
Wykonaj scalanie dla gałęzi *cecha2* (wykonaj operację scalania drugą metodą - nie wybraną przy poprzednim scalaniu). Pamiętaj o zapisaniu obrazu grafu wersji (rewizji).
Jakie są różnice w działaniu polecenia *merge* i *rebase*?

Wprowadź zmiany w gałęzi *cecha2* zgodnie z opisem w pliku *zmiana3.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany.

Scalanie zmian do gałęzi master

Praca nad sekcjami została zakończona i należy dokonać scalenia głównej gałęzi projektu (master) z zawartościami gałęzi *cecha1* oraz *cecha2*.
Przetestuj poprawność scalonej struktury. Pamiętaj o zapisaniu obrazu grafu wersji (rewizji).

Finalizowanie i upublicznianie zmian

Wprowadź (w gałęzi master) zmiany zgodnie z opisem w pliku *zmiana4.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany.
Wyślij rezultat do repozytorium zdalnego.
W ostatnim kroku wprowadź zmianę zgodnie z opisem w pliku *zmiana5.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany.
Ponownie wyślij poprawioną wersję do repozytorium zdalnego.
Za pomocą funkcji systemu GitHub utwórz wydanie (release) oznaczając go jako v0.1

Praca zespołowa

[*Ćwiczenie wymaga wcześniejszego wykonania powyższych zadań*]
Utwórz projekt na swoim koncie GitHub na podstawie projektu (komenda systemu GitHub: fork - https://help.github.com/articles/fork-a-repo) kolegi/koleżanki po twojej prawej stronie (jeżeli jesteś ostatnią osobą w rzędzie - osobą siedzącą po twojej prawej stronie jest pierwsza osobą w drugim rzędzie). Jeżeli osoba nie ukończyła jeszcze poprzednich zadań - najpierw jej pomóż. Następnie dokonaj operacji klonowania tak utworzonego repozytorium zdalnego na komputer lokalny. Wprowadź w projekcie zmiany zgodnie z opisem w pliku *zmiana6.txt*. Przetestuj poprawność wprowadzonych zmian. Zatwierdź zmiany. Wyślij rezultat do repozytorium zdalnego. Wyślij do kolegi/koleżanki informacje o zmianach (pull request).
Na podstawie otrzymanego “pull request” wprowadź zmiany, dokonane przez kolegę/koleżankę, do swojego repozytorium. Utwórz nowe wydanie oznaczając je jako v0.2.

Pytania uzupełniające

1. Jaką komendę systemu git wykorzystuje platforma GitHub do tworzenia wydań (release)?
2. W jaki sposób wycofać zmiany wprowadzone w pliku w kopii roboczej (czyli przywrócić ostatnią wersję z repozytorium)?
3. W jaki sposób cofnąć się do poprzedniej wersji w repozytorium? W jaki sposób cofnąć się o N wersji?
4. W jaki sposób wycofać zmiany w repozytorium tak aby najnowsza wersja gałęzi odpowiadała zmianom wprowadzonym N commitów wcześniej? Podaj dwie metody, różnice pomiędzy nimi oraz, wynikający z różnic, zakres stosowalności.

Interaktywne kursy:

https://commons.lbl.gov/display/pbddocs/Learn+Git
http://pcottle.github.io/learnGitBranching/

Materiały:

http://git-scm.com/book/pl/
http://githubtraining.s3.amazonaws.com/github-git-training-slides.pdf
http://ndpsoftware.com/git-cheatsheet.html