Szymon SUPERNAK[1]

PRACA GRUPOWA Z WYKORZYSTANIEM REPOZYTORIUM PROJEKTU

Streszczenie

Celem artykułu jest przedstawienie aspektów pracy grupowej w projekcie informatycznym z wykorzystaniem repozytorium w procesie tworzenia oprogramowania. Omówiono narzędzia umożliwiające swobodny przepływ informacji o projekcie i dzielenie się nią. W narzędziach typu CASE artefakty projektowe gromadzi się w repozytoriach pracy grupowej, które wspomagają organizację pracy zespołu, umożliwiają śledzenie przebiegu pracy oraz kontrolę wersji artefaktów. Zaprezentowano zintegrowane platformy programistyczne systematyzujące pracę grupową zespołu projektowego z wykorzystaniem dedykowanych metodologii projektowych.

Abstract

The goal of the following study is to show the features of group work in a computer science project incorporating a repository in the process of software development. The tools of information sharing and unrestricted information flow were put up for discussion. In case of tools such as CASE, project artifacts are gathered in group work repositories which enhance organization of the team work, enable the progress monitoring and control of the artifact versions. The integrated programming platforms regularizing group work of project team with the use of dedicated project methodologies were presented.

1 WPROWADZENIE

Zarządzanie projektami i koordynowanie pracy grupowej to podstawowe tematy dobrze działającego zespołu informatycznego. Odpowiednie zaplanowanie pracy, zbudowanie zespołu, przydzielenie zadań, dyskusje czy raportowanie stanowią nieodłączną część każdego prowadzonego projektu. Z pomocą każdemu zespołowi przychodzą zaawansowane programy komputerowe – dzisiaj dostępne również przez przeglądarkę jako aplikacje internetowe.

Różnorodne zadania składające się na realizację projektu programistycznego spowodowały naturalny podział twórców oprogramowania na analityków, projektantów i programistów. Konieczność pracy zespołowej nie wynika tylko z tego, że do wykonania są różne grupy zadań. Każdy z wymienionych rodzajów zadań wymaga w projekcie więcej niż jednej osoby, aby zdążyć z wykonaniem całości na czas. Istnieje, zatem nie tylko pot-

[1] Dr inż. Szymon Supernak jest wykładowcą Warszawskiej Wyższej Szkoły Informatyki.

rzeba współpracy pomiędzy informatykami zajmującymi się różnymi zadaniami, ale także pomiędzy pracownikami zajmującymi się tymi samymi bądź podobnymi zadaniami. Ten drugi przypadek oznacza zwykle znacznie bliższą, intensywniejszą i dłużej trwającą współpracę w projekcie.

Specjaliści w oparciu o elementy metodyki obiektowej lub strukturalnej, wykonują projekt logiczny i techniczny systemu informatycznego. Tworzą adekwatny model relacyjnej bazy danych, opracowują skrypty tworzące strukturę bazy danych oraz przykładowe dane testowe. Realizują opis systemu z wykorzystaniem wybranych diagramów UML (klas, przypadków użycia, stanu, czynności, wdrożenia), tworzą kod właściwej aplikacji z zachowaniem ustalonych wcześniej standardów interfejsów GUI. Programują tzw. logikę działania, zgodnie z zasadami inżynierii programowania oraz standardami kodowania przyjętymi dla wybranego języka i środowiska programowania. Implementują funkcjonalność raportującą, dokonują strojenia bazy danych pod kątem szybkości realizacji wybranych, krytycznych zapytań generowanych z aplikacji poprzez analizę planu zapytania. Tworzą kontekstowy, hipertekstowy system pomocy przyłączany do aplikacji, wykonują wersje instalacyjne produktu końcowego [1].

Za każdym projektem kryją się ludzie, którzy biorą udział w jego realizacji. Dodawanie członków do zespołów wykonujących poszczególne projekty, to również podstawowa funkcjonalność aplikacji do pracy grupowej. W projektach informatycznych aplikacje do pracy grupowej uwzględniają np. wersjonowanie oprogramowania – to umożliwia definiowanie poszczególnych funkcji i przypisywanie ich do kolejnych wydań. Często określanych mianem faz czy kroków milowych, a więc swego rodzaju kategorii, grupujących poszczególne zadania. Oprogramowanie koordynujące pracę grupową służy przede wszystkim celowi, którym jest wykonanie projektu na czas. Skuteczność projektu zależy od dobrego rozplanowania pracy, dobrania odpowiednich osób i sprawnej realizacji przydzielonych zadań poprzez właściwe wykorzystanie repozytorium projektu.

2 INTEGRACJA I ŁĄCZENIE MODELI RÓŻNYCH TYPÓW

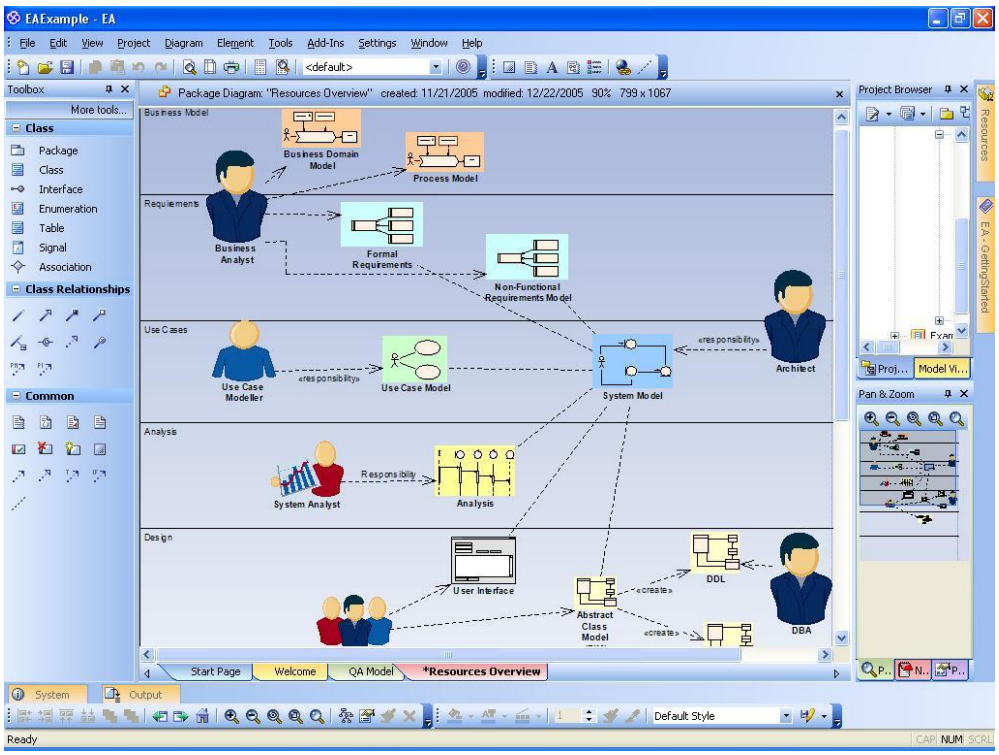
Niezwykle szybki rozwój wymagań stawianych informatykom oraz ciągłe poszerzanie obszaru zastosowań informatyki w naszym codziennym życiu powodują, że ciągle powstaje dużo nowych aplikacji, ale również wielkość i złożoność tych programów stale wzrasta. Typowa współczesna aplikacja powinna:

1) mieć przyjazny, wygodny w użyciu graficzny interfejs (GUI),
2) mieć zaawansowany, kontekstowy system podpowiedzi, często z różnorodnymi kreatorami dla typowych zadań,
3) móc pracować w środowisku sieciowym i rozproszonym,

4) zapewniać ochronę danych (od poziomu prostego systemu kont użytkowników z różnymi zakresami praw działania, a kończąc na zaawansowanych protokołach szyfrowania informacji),
5) posiadać strukturę profesjonalnie udokumentowaną.

Żądania te wymagają dużego wkładu pracy na poziomie analizy, projektowania i programowania. Np. stworzenie dobrego interfejsu użytkownika, który byłby intuicyjny, wygodny i bezpieczny - wymaga bardzo starannego zaprojektowania.

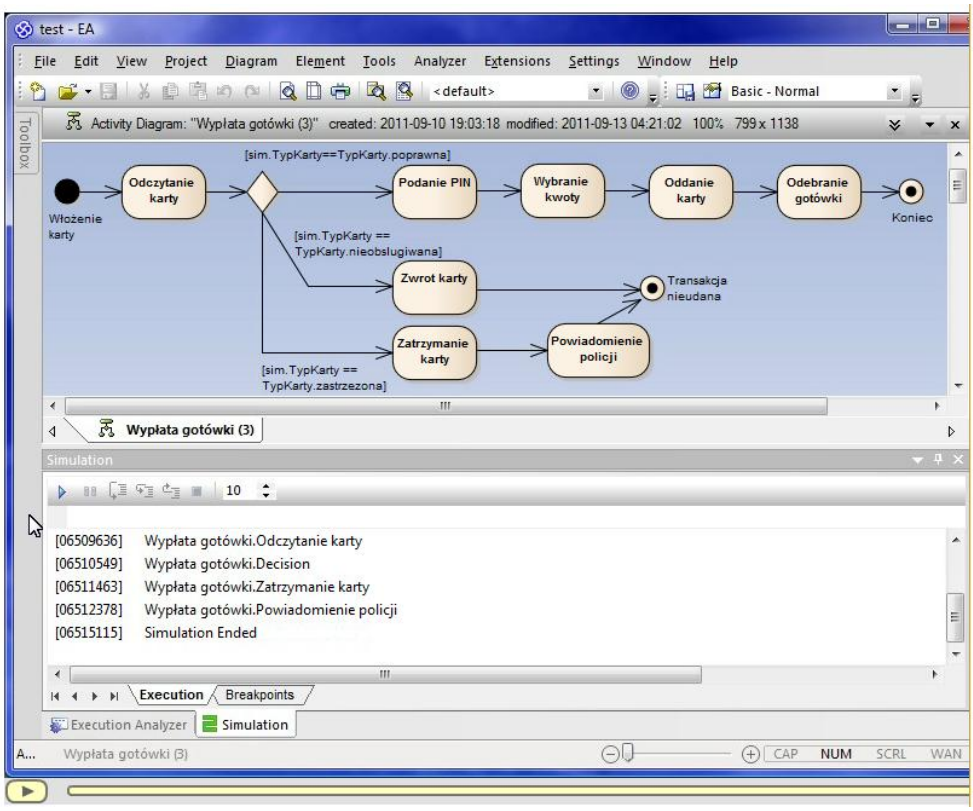
Do zrealizowania wszystkich tych zadań niezbędne jest stosowanie profesjonalnych narzędzi wspomagania projektowania. Można między innymi wykorzystać Enterprise Architect.



Rys. 1. Widok modeli w języku UML w pakiecie EA [2]

Standardowy system informatyczny modelujemy pokazując jego strukturę i jego zachowania. Modele struktury (klas, komponentów, pakietów) są modelami statycznymi i pokazują to, co nie zmienia się w czasie. Modele zachowań (czynności, sekwencji, stanów) pokazują zmianę. Problem w tym, że raz wykonany diagram sam się nie zmienia i pokazuje wszystkie zmiany od początku do końca. Dobre zrozumienie, a szczególnie uzgodnienie tego zrozumienia jest łatwiejsze, jeśli diagram zachowań możemy omówić

krok po kroku. Współczesne systemy wspomagania projektowania dostarczają nam również narzędzia pozwalającego śledzić dynamikę z diagramu – symulację modeli.

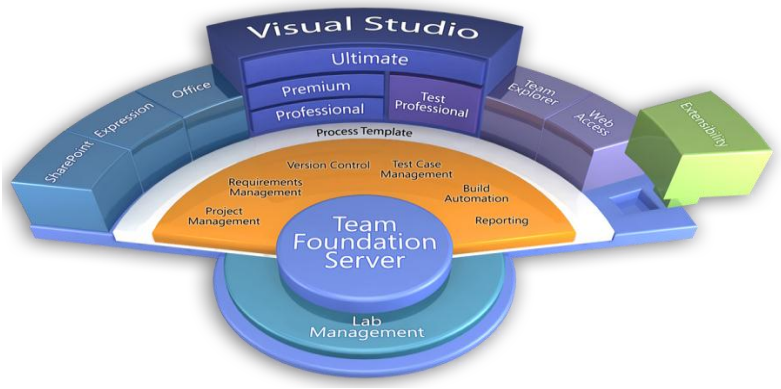


Rys. 2. Symulacja modelu dla wybranego przykładu [2]

W przypadku Enterprise Architect można zarządzać złożonością za pomocą narzędzi do monitorowania zależności, wsparciem dla bardzo dużych modeli, kontrolą wersji (CVS lub SCC), porównaniem narzędzi do monitorowania zmian w modelu, intuicyjnemu interfejsowi z wyglądem przypominającym przeglądarkę Internet Explorer. Dla pracujących w Eclipse czy Visual Studio .NET, Sparx Systems dostarcza integrację z tymi IDE. Pozwala to na modelowanie i przechodzenie bezpośrednio do kodu źródłowego w preferowanym edytorze.

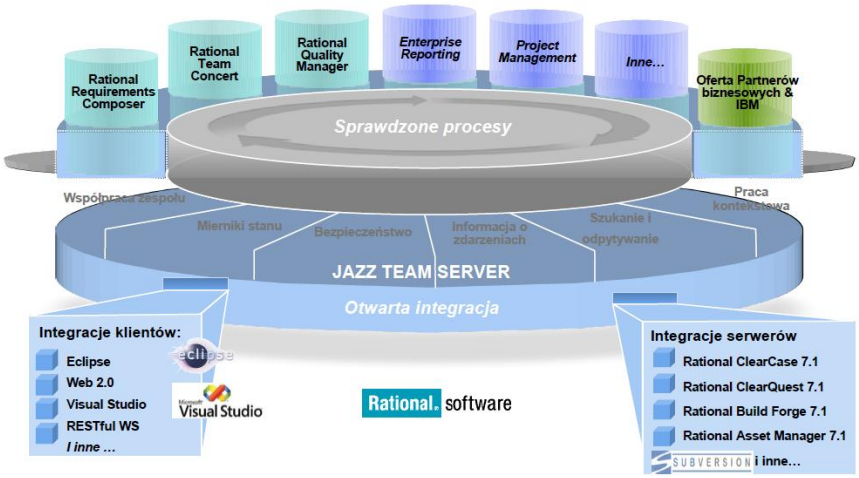
Obecnie do realizacji projektów stosuje się również zintegrowane platformy programistyczne. Jedna z nich do pracy grupowej wykorzystuje Microsoft Visual Studio 2010 Team Foundation Server, który wchodzi w skład platformy programistycznej Visual Studio 2010 Premium oraz Ultimate. Jego głównym celem jest prowadzenie biblioteki kodu, do której dostęp mają wszyscy członkowie zespołu. W trakcie pracy prowadzona

jest historia wersji, możliwe jest także tworzenie raportów i śledzenie obiektów. Team Foundation Server zarządza też komunikacją pomiędzy deweloperami.



Rys. 3. Platforma Visual Studio 2010 [3]

Druga platforma programistyczna opracowana przez firmę IBM usprawnia pracę zespołową poprzez automatyzację, przejrzystość i przewidywalność w produkcji oprogramowania.



Rys. 4. Efektywność pracy z platformą JAZZ [4]

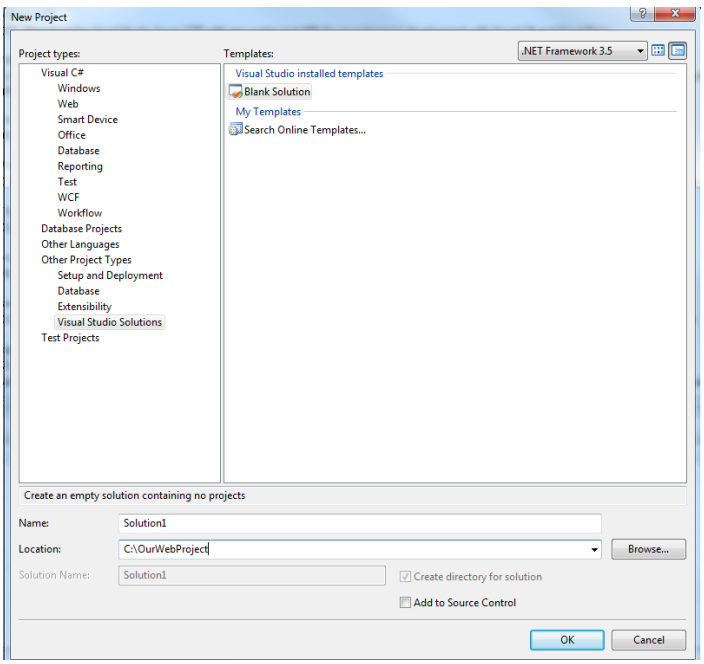
Rational Team Concert oferuje rozbudowane funkcje zarządzania tworzeniem wersji, które pozwalają zespołom wydajnie planować i realizować budowanie wersji oprogramowania. Wbudowane mechanizmy śledzenia zadań i zbiorów zmian ułatwiają dokładną identyfikację miejsc występowania błędów przy budowie, zaś rozszerzone funkcje raportowania przy budowaniu automatycznie udostępniają szczegółowy rejestr prac.

3 WSPARCIE PRACY GRUPOWEJ I DOSTĘP DO MODELI W REPOZYTORIUM

W celu przechowywania i łatwego udostępniania kodu członkom zespołu, można wykorzystać Visual Studio Team Foundation Server który jest wbudowany w Visual Studio 2010 Premium oraz Ultimate [3].

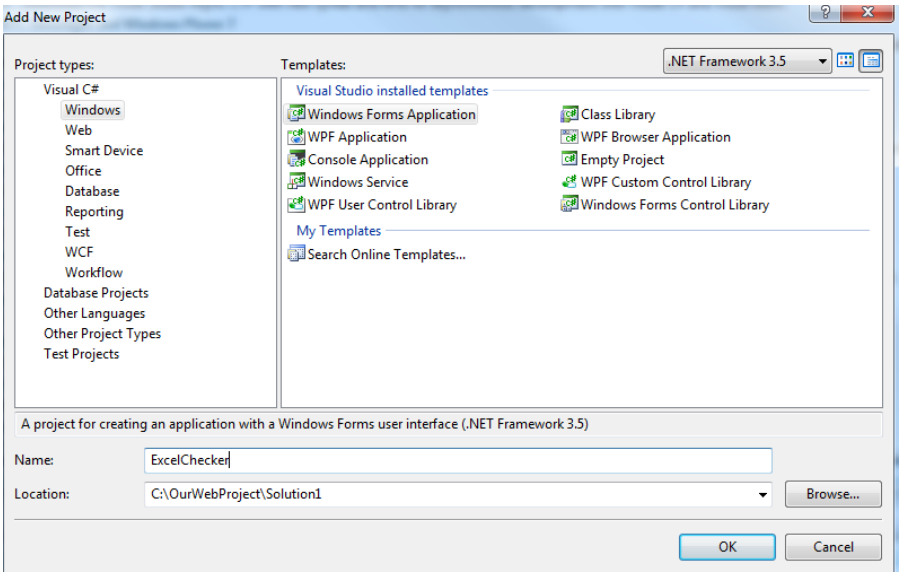
Opis przygotowania repozytorium, z którego będą mogły korzystać osoby pracujące nad projektem:

- **Stworzenie folderu lokalnego, w którym będzie przechowywany projekt.**
  Nasz folder bazowy znajduje się na komputerze jednego z członków zespołu, który posłuży za nasz server przechowujący projekt. Folder utworzyliśmy na dysku C pod nazwą „OurProject" (C:\OurWebProject).
- **Tworzenie nowego rozwiązania 'Blank Solution'**.
  Warto je tworzyć, gdyż, jeśli w przyszłości będziemy chcieli stworzyć nowy projekt będzie można go umieścić w istniejącym rozwiązaniu, co umożliwi nam łatwiejsze korzystanie z komponentów. W Visual Studio wybieramy z menu File→New→ Project. Następnie w 'Other Project Types' wybieramy Visual Studio Solutions, tam wybieramy 'Blank Solution' i nazywamy „WindowsFormAplikaction", na koniec umieszczamy ją we wcześniej utworzonej lokalizacji czyli C:\OurWebProject.



Rys. 5. Nowy projekt [5]

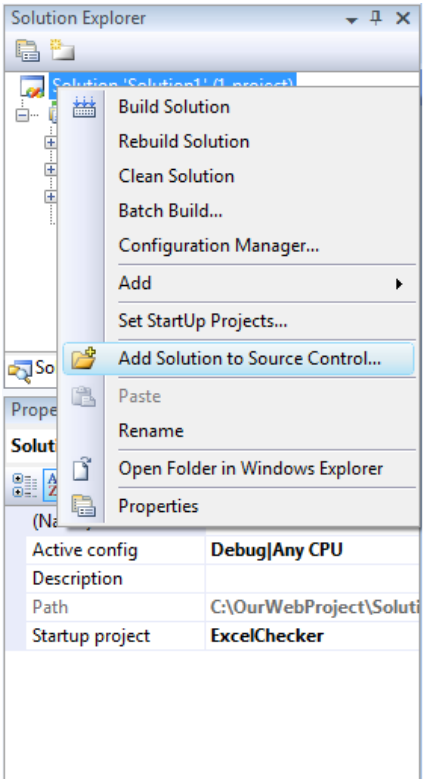
- **Tworzenie nowego projektu i dodanie go do rozwiązania.**
  W ‘Solution Explorer’ klikamy prawym przyciskiem nazwę rozwiązania, po czym wybieramy ‘Add’→New Project. Wybieramy język programowania (w naszym przypadku jest to C#) i typ projektu ‘WindowsForms-Application’.



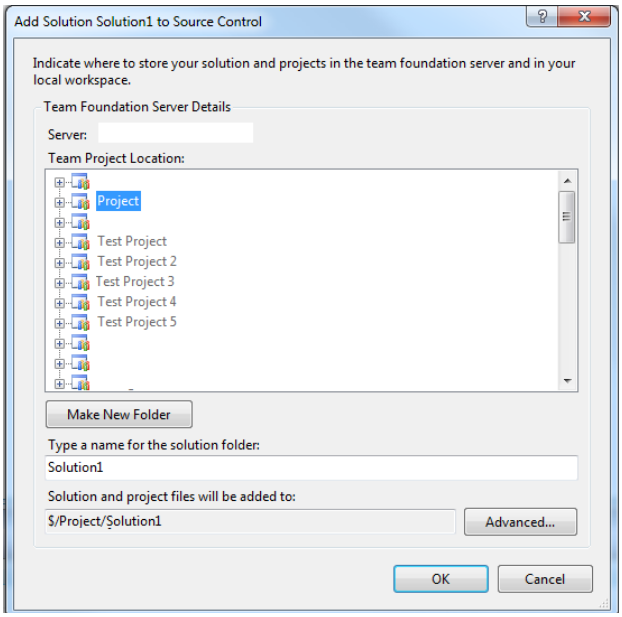
Rys. 6. Dodawanie nowego projektu [5]

- **Dodawanie strony do “Solution”.**
  W ‘Solution Explorer’, prawym przyciskiem myszy wybieramy nazwę naszego rozwiązania i wybieramy ‘Add’→’New Web Site’. W oknie dialogowym ‘Add New Web Site’ ustawiamy ‘Location’ na ‘HTTP’ i zostawiamy domyślnie ‘Language’ na Visual C#’. Ustawiamy ‘Location URL’ na http://localhost/WindowsFormsApplication2.

- **Dodanie rozwiązania do kontroli kodu.**
  Prawym przyciskiem myszy zaznaczamy nowe rozwiązanie i wybieramy ‘Add Solution to Source Control’, następnie wybieramy nasz projekt i tworzymy folder ‘Make New Folder’ po czym tworzymy jeszcze jeden folder Source, który będzie widoczny w Team Explorer. Po czym klikamy OK.

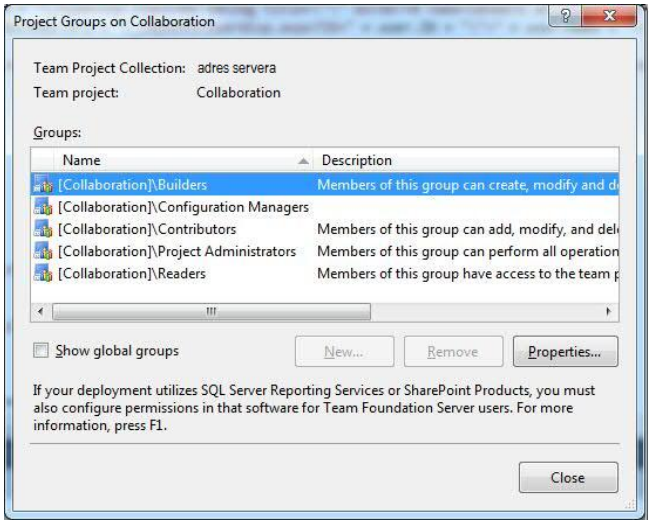


Rys. 7. Dodawanie rozwiązania [5]



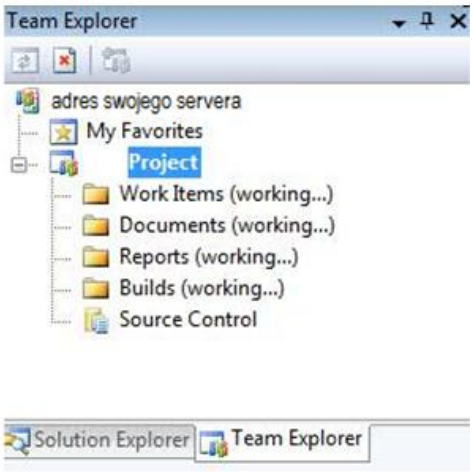
Rys. 8. Dodawanie użytkowników [5]

- **Dodawanie użytkowników do projektu.**
  W Team Explorer, wybieramy nasz projekt, wciskamy ‘Team Project Settings’, a następnie ‘Group Membership’. W oknie dialogowym ‘Project Groups’ wybieramy grupę, do której dodajemy użytkowników i klikamy “Properties”. W odpowiednim miejscu wpisujemy użytkowników, jeśli dodajemy więcej niż jednego, oddzielamy ich średnikiem.



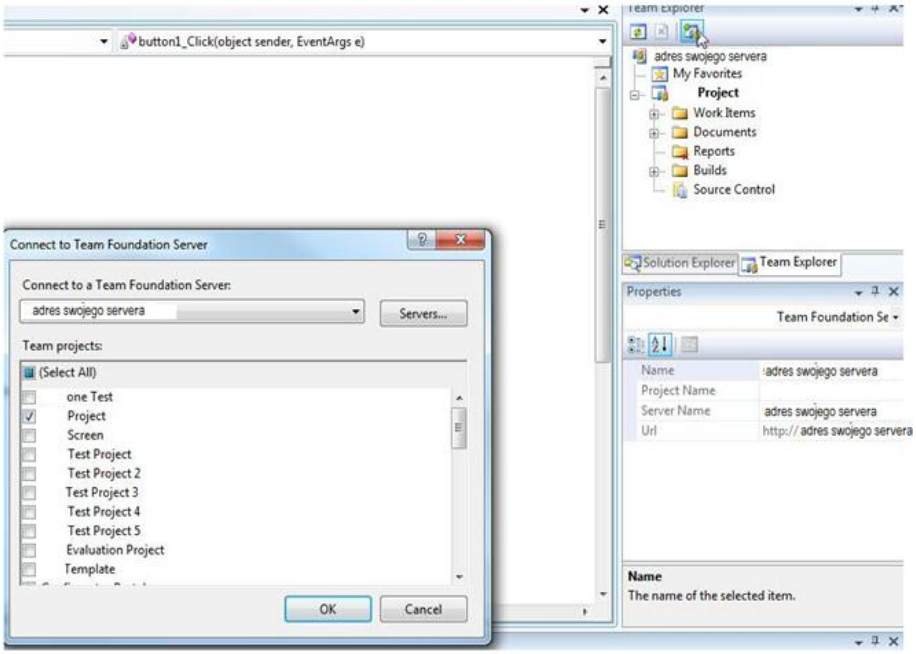
Rys. 9. Dodawanie użytkowników 2 [5]

- **Pobieranie kopi projektu na swój dysk**.
  Klikamy w zakładkę Team Explorer.



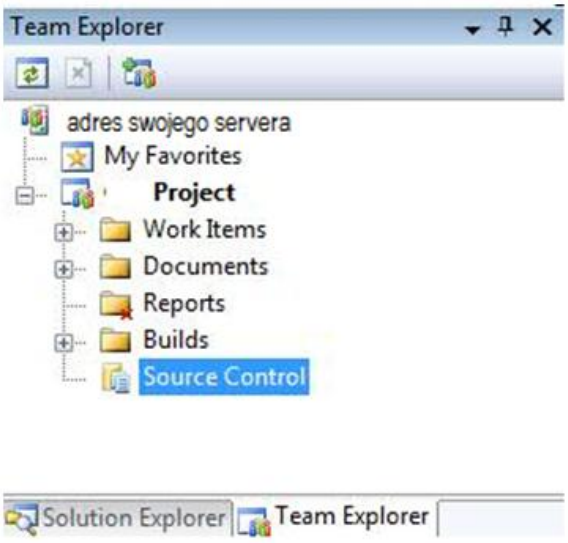
Rys. 10. Widok Team Explorera [5]

Wybieramy „Add Existing Team Project” (wybieramy serwer i projekt),



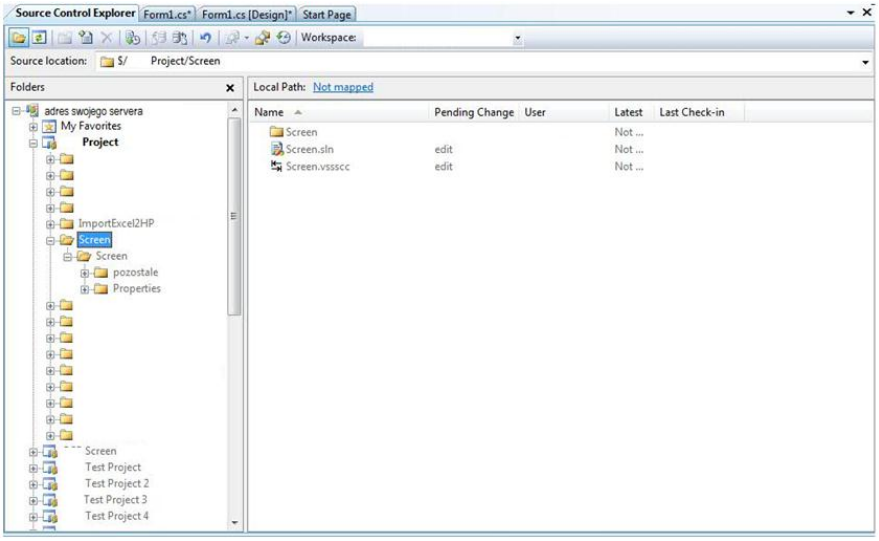
Rys. 11. Dodawanie projektu [5]

klikamy dwa razy w source control:



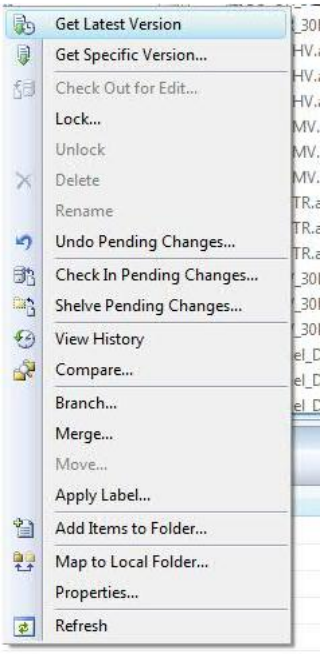
Rys. 12. Source Control [5]

Szukamy projektu, nad którym będziemy pracować i wybieramy go:



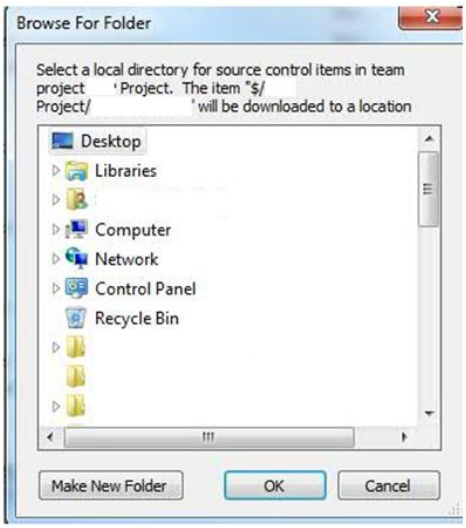
Rys. 13. Okno wyboru projektu [5]

Za każdym razem, gdy chcemy zaktualizować projekt do najnowszej wersji klikamy prawym przyciskiem myszy na projekcie i wybieramy „Get Latest Version”:



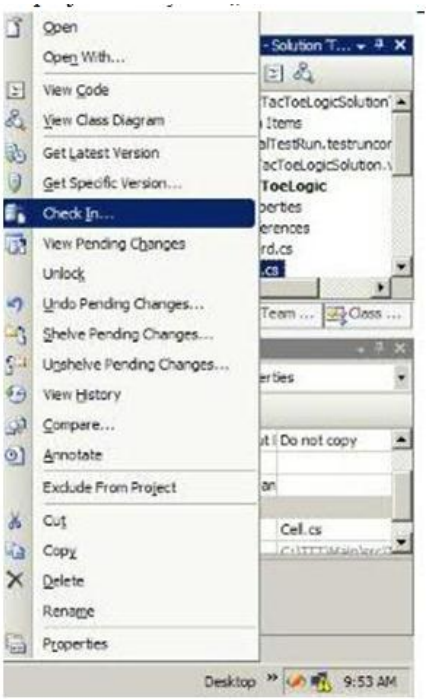
Rys. 14. Aktualizacja projektu [5]

Gdy robimy to pierwszy raz, konieczne jest wybranie miejsce na dysku, w którym chcemy zapisać nasz projekt:



Rys. 15. Miejsce na dysku [5]

Aby zapisać naszą zmienioną wersję projektu na serwer ogólny, należy kliknąć prawym przyciskiem myszy na projekcie i wybrać „Check In”:



Rys. 16. Zapis nowego projektu [5]

4 IMPORT MODELI Z REPOZYTORIÓW INNYCH NARZĘDZI

Projekty informatyczne stają się coraz bardziej skomplikowane, trwają coraz dłużej, wymagają efektywnej i dobrze skoordynowanej współpracy zespołu specjalistów. Zarządzanie zmianą składa się z dwóch podstawowych elementów:

- kontroli wersji,
- śledzenia błędów i zmian.

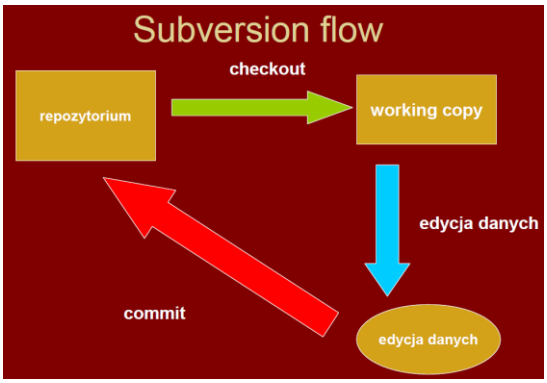
Jednym z najbardziej podstawowych sposobów na usprawnienie komunikacji w zespole jest użycie narzędzi do wersjonowania kodu źródłowego programów, dokumentacji oraz wszelkich innych plików powstających przy wytwarzaniu produktu (np. grafika, podręcznik użytkownika, itp.).

Kontrola wersji pozwala na śledzenie zmian wnoszonych przez poszczególnych członków zespołu, umożliwia, w razie potrzeby, przywrócenie poprzednich wersji oprogramowania, pozwala zablokować dostęp do wybranych zasobów tak, aby dwie lub więcej osób nie modyfikowały jednocześnie tego samego pliku, co może mieć niepożądane konsekwencje.

Kolejnym sposobem usprawnienia pracy zespołu jest wykorzystanie narzędzia do śledzenia błędów i zmian w projekcie. Jest rzeczą oczywistą, że oprogramowanie w czasie swojego cyklu życia, począwszy od wczesnej fazy projektowania systemu, poprzez etap budowania system, aż do wdrożenia kolejnych wersji u klienta będzie się zmieniało. Zmiany te mogą wynikać z pojawiających się nowych wymagań lub modyfikacji tych, które były już uwzględnione, albo być wymuszone przez wykrycie błędów.

Kluczową sprawą jest takie zarządzanie cyklem życia aplikacji, które gwarantuje, że zgłoszony błąd, czy zgłoszone nowe wymaganie, nie zostanie „zgubione”. Powinno ono być przydzielone odpowiednim osobom, które albo zajmą się poprawieniem błędów, albo, zanim problem zostanie przekazany programistom, ocenią jak bardzo kosztowne będzie wprowadzenie żądanych zmian.

Takie określenie trybu postępowania ze zgłoszonymi wymaganiami nazywa się często zdefiniowaniem reguł przebiegu pracy (ang. *workflows*).



Rys. 17. Przepływ pracy w repozytorium [6]

W systemach kontroli wersji używa się kilku zwrotów. W zasadzie są one wspólne dla większości systemów.

- **Repository** (repozytorium) – miejsce przechowywania plików w systemie kontroli wersji, niekoniecznie baza danych, często jest to pewna specyficzna struktura katalogów z opisami.
- **Working copy** (katalog roboczy) – miejsce, w którym znajdują się kopie lokalne plików, nad którymi pracujemy. Najczęściej katalog na naszym komputerze.
- **Get** (pobranie) – pobranie plików z repozytorium np.: z serwera.
- **Check-in** (wstawienie) – gdy skończymy zmieniać pliki to wstawiamy je na serwer, tak aby każdy mógł je pobrać w najnowszej wersji.
- **Check-out** (wybranie) – pobranie plików z serwera z zamiarem zmiany. Może być związane z jednoczesną blokadą.
- **Lock** (blokada) – zablokowanie innym użytkownikom możliwości wybrania.
- **Label** (etykieta) – oznaczenie nazwą grupy plików w celu łatwej ich identyfikacji.
- **Branch** (gałąź) – stworzenie alternatywnej linii wersjonowania. Pozwala to na rozwój tego samego pliku w różnych kierunkach, np.: wprowadzanie poprawek do starej wersji i dodawanie funkcji do nowej wersji.
- **Merge** (połączenie) – łączenie dwóch kopii pliku w jeden. Może się odnosić do dwóch sytuacji: łączenia gałęzi i rozwiązywania konfliktów.

Powyższe funkcje opisują podstawową funkcjonalność kontroli wersji oprogramowania.

Gdy edytujemy tekstowe pliki kodu źródłowego to blokada zazwyczaj nie jest konieczna. Jeżeli dwie osoby zmienią ten sam fragment kodu to później muszą połączyć

swoje zmiany do nowej wersji. Korzystając z powyższej terminologii pracę z systemem kontroli wersji możemy opisać przez taki ciąg działań:

- tworzymy repozytorium,
- dodajemy do niego pliki projektu,
- pobieramy (Get) pliki z repozytorium do katalogu roboczego,
- wybieramy (Check-out) pliki do edycji,
- zmieniamy to co należy zmienić,
- sprawdzamy, czy wszystko działa,
- wstawiamy (Check-in) zmienione pliki do repozytorium.

Jeżeli okaże się, że ktoś zmienił plik, który wstawiamy to prawdopodobnie będziemy musieli połączyć (Merge) wersje.

System kontroli wersji nie rozwiązuje jednak wszystkich problemów w zespole programistycznym. Jest tak naprawdę częścią większej całości – systemu zarządzania zmianą. System zarządzania zmianą powinien pozwolić na łączenie zmian w kodzie z żądaniami zmiany i śledzeniem błędów. Takie podejście pozwala na dokładniejsze zarządzanie projektem i pracą. Samo podejście do projektu przez pryzmat żądań zmiany (ang. *change request*) i próśb o funkcjonalność (ang. *feature request*) porządkuje pracę, która musi być wykonana i pozwala na jej właściwą organizację.

5 PODSUMOWANIE

Za każdym projektem kryją się ludzie, którzy biorą udział w jego realizacji. Dodawanie członków do zespołów wykonujących poszczególne projekty to również jedna z podstawowych funkcji aplikacji do pracy grupowej**.**

Głównymi problemami w pracy grupowej z wykorzystaniem repozytorium są:

- równoległa praca wielu osób nad tym samym projektem;
- ryzyko nadpisywania zmian;
- problem najnowszej wersji;
- potrzeba odwoływania się do poprzednich wersji;
- monitorowanie prac.

W zależności od wybranego języka i środowiska programistycznego konieczne jest stosowanie podstawowego zestawu narzędzi:

- Narzędzie do zarządzania wersjami – przy programowaniu w zespole cenna jest możliwość odtwarzania poprzednich wersji (chociażby po to, żeby się przekonać, czy nowy błąd w programie pochodzi z nowego własnego modułu, czy też z modułu kolegi).

- Środowisko programistyczne – jego ważnym elementem musi być wsparcie programowania wizualnego, służącego do łatwego tworzenia interfejsów użytkownika wraz ze szkieletem aplikacji.
- Narzędzia ułatwiające tworzenie dokumentacji – na tworzenie dokumentacji istnieje bardzo duży nacisk.
- Narzędzia do zarządzania informacjami o zgłaszanych błędach i uwagach (kto zgłosił, opis, kto jest odpowiedzialny za obsłużenie, jaki jest obecny stan, itp.) – utrzymywanie i pielęgnowanie kodu jest szczególnie istotne.

Oczywiście należy stosować dodatkowe narzędzia (np. do tworzenia dokumentacji projektowej) korzystając z funkcjonalności wybranego pakietu wspomagania projektowania systemów informatycznych.

Literatura

1. W. Dąbrowski, A. Stasiak, M. Wolski: *Modelowanie systemów informatycznych w ję-zyku UML 2.1*; PWN 2007 (dodruk 2009).
2. Dokumentacja techniczna i użytkownika narzędzia CASE SPARX. Enterprise Architect: http://www.sparxsystems.com.au.
3. R. Pagels: *Materiały szkoleniowe Microsoft – Visual Studio.* 2010, Ultimate.
4. Dokumentacja techniczna Platforma JAZZ; http://jazz.net.
5. S. Supernak: Materiały dydaktyczne *Projekt Indywidualny*. WWSI, 2011.
6. Dokumentacja techniczna SUBVERSION; http://subversion.apache.org.