Zarządzanie projektem IT z wykorzystaniem usług Microsoft Visual Studio Team Foundation Server.

Mateusz Sierzputowski

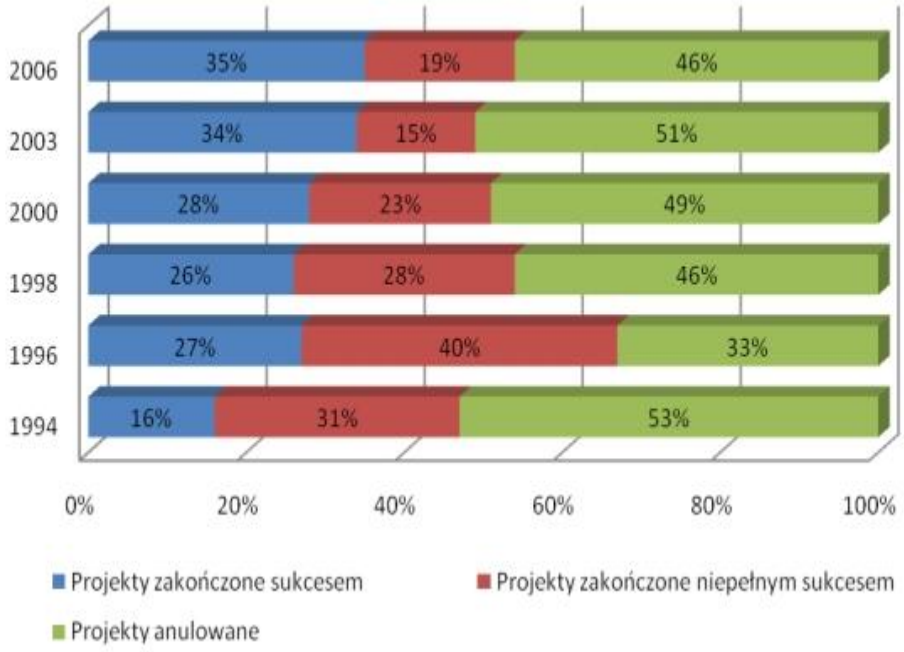
Politechnika Warszawska – Wydział elektryczny

1.Wprowadzenie

1.1 Pojęcie zarządzania projektem.

Warto zastanowić, co możemy nazwać zarządzaniem projektem. W mojej opinii jest to efektywne osiąganie zamierzonych celów projektowych, z równoczesnym radzeniem sobie z ograniczeniami i problemami (np. ryzykiem) projektowymi.

Dlaczego warto umiejętnie i efektywnie zaplanować, a później zarządzać projektem? Do wytłumaczenia tego posłużę się dwoma raportami. Pierwszy z nich obrazuje realizację projektów, z podziałem na projekty zakończone, niezakończone oraz projekty częściowo zakończone tzw. dowiezione. Drugi raport przedstawia czynniki sukcesu projektów informatycznych.



Rysunek 1 Źródło : http://pmanager.pl/

Jak widać część procentowa projektów zakończonych sukcesem jest mniejsza niż projektów anulowanych lub dowiezionych. Mimo iż według wykresu udział projektów zakończonych pozytywnie rośnie to nadal nie stanowi nawet połowy podjętych projektów. Co więc i w jakim stopniu ma wpływ na sukces projektu IT?

1.	***Zaangażowanie klienta***	**15.9%**
2.	***Wsparcie kierownictwa***	**13.9%**
3.	***Jasno określone wymagania***	**13.0%**
4.	***Właściwe planowanie***	**9.6%**
5.	***Realistyczne oczekiwania***	**8.2%**
6.	***Mniejsze odstępy pomiędzy kamieniami milowymi***	**7.7%**
7.	***Kompetencje pracowników***	**7.2%**
8.	***Odpowiedzialność***	**5.3%**
9.	***Jasno postawione cele i wymagania***	**2.9%**
10.	***Ciężko pracujący, skupieni pracownicy***	**2.4%**
11.	***Pozostałe***	**13.9%**

Analizując powyższą tabelkę można zauważyć, że wszystkie kryteria mają związek z zarządzaniem projektem. Siedem z nich bezpośrednio dotyczy planowania i zarządzania projektem. Można z tego wywnioskować, że przyczyną niepowodzeń projektów informatycznych, jest właśnie nieumiejętne zarządzanie nimi.

1.2 Pojęcie metodyki.

Metodyka to ustandaryzowane dla wybranego obszaru podejście do rozwiązywania problemów. Istnieją różne rodzaje metodyk, zanim jednak do tego przejdziemy, warto przyjrzeć się historii rozwoju metodyk.

Pierwsze systemy komputerowe były tworzone dla organizacji rządowych lub dużych korporacji. Charakteryzowały się więc zwykle ściśle określonymi normami i przepisami, które miały cechować te systemy. Z tego powodu można było sobie pozwolić na to by przeprowadzić długą i bardzo dokładną analizę wymagań biznesowych, przygotować ogromną ilość dokumentacji, a testy przeprowadzać dopiero po zakończeniu prac związanych z implementacją. To właśnie są cechy metodyk tzw. twardych, takich jak PRINCE2.

Na przełomie XX i XXI wieku, coraz większą popularność zyskiwały firmy średnie oraz małe, które również zaczęły potrzebować efektywnych systemów komputerowych.

Szybko okazało się, że metody, które sprawdzały się w korporacjach nie zawsze sprawdzały się w mniejszych przedsiębiorstwach. Długotrwałe procesy analizy i dokumentacji były nie do zaakceptowania. Zmiana już wykonanego systemu była bardzo czasochłonna, a co za tym idzie tworzyły się dodatkowe koszty. Trzeba było więc wprowadzić coś nowego, co wyjdzie naprzeciw oczekiwaniom klienta.

1.3 Manifest Agile

W 2001 roku reprezentanci nowych metodyk tworzenia oprogramowania ogłosili Manifest Zwinnego Wytwarzania Oprogramowania, który brzmi następująco:

Wytwarzając oprogramowanie i pomagając innym w tym zakresie,
odkrywamy lepsze sposoby wykonywania tej pracy.
W wyniku tych doświadczeń przedkładamy:

Ludzi i interakcje ponad procesy i narzędzia.
Działające oprogramowanie ponad obszerną dokumentację.
Współpracę z klientem ponad formalne ustalenia.
Reagowanie na zmiany ponad podążanie za planem.

Doceniamy to, co wymieniono po prawej stronie,
jednak bardziej cenimy to, co po lewej.

Do tego przedstawiono 12 zasad wytwarzania zwinnego oprogramowania.

1.Najważniejsze dla nas jest zadowolenie Klienta wynikające z wcześnie rozpoczętego i ciągłego dostarczania wartościowego oprogramowania.

2.Bądź otwarty na zmieniające się wymagania nawet na zaawansowanym etapie projektu. Zwinne procesy wykorzystują zmiany dla uzyskania przewagi konkurencyjnej Klienta.

3.Często dostarczaj działające oprogramowanie od kilku tygodni do paru miesięcy, im krócej tym lepiej z preferencją krótszych terminów.

4.Współpraca między ludźmi biznesu i programistami musi odbywać się codziennie w trakcie trwania projektu.

5.Twórz projekty wokół zmotywowanych osób.
Daj im środowisko i wsparcie,
którego potrzebują i ufaj im, ze wykonają swoją pracę.

6.Najwydajniejszym i najskuteczniejszym sposobem przekazywania informacji do i ramach zespołu jest rozmowa twarzą w twarz

7.Podstawową i najważniejszą miarą postępu jest działające oprogramowanie.

8.Zwinne procesy tworzą środowisko do równomiernego rozwijania oprogramowania. Równomierne tempo powinno być nieustannie utrzymywane poprzez sponsorów, programistów oraz użytkowników.

9.Poprzez ciągłe skupienie na technicznej doskonałości i dobremu zaprojektowaniu oprogramowania zwiększa zwinność.

10.Prostota – sztuka maksymalizacji pracy niewykonanej – jest zasadnicza.

11.Najlepsze architektury, wymagania i projekty powstają w samoorganizujących się zespołach.

12.W regularnych odstępach czasu zespół zastanawia się jak poprawić swoją efektywność, dostosowuje lub zmienia swoje zachowanie.

Te zasady stały się punktem wyjściowym dla powstania metodyk zwinnych, czyli opartych na programowaniu iteracyjnym.

1.4 SCRUM

SCRUM jest jedną z metodyk zwinnych. Jest to termin pochodzący z Rugby. Twórca metodyki uznał, że zasady gry w Rugby świetnie obrazują, jak ma wyglądać proces tworzenia oprogramowania. Po opuszczeniu przez piłkę placu gry lub naruszeniu zasad drużyny planują, jak rozegrać kolejną akcję. Następnie grupy zawodników ustawiają się naprzeciw siebie, oplątani ramionami z głowami skierowanymi w dół i usiłują przejąć kontrolę nad leżącą w centrum piłką, żeby zrealizować swój założony plan, aż do momentu kolejnego zatrzymania gry. Dokładnie w ten sposób wygląda proces wytwarzania oprogramowania w metodyce Scrum.

W SCRUMIE występują zespół tworzą tylko trzy typy ról, w przeciwieństwie do metodyk twardych. Są to:

- **Product Owner** – osoba będąca łącznikiem pomiędzy zespołem, a klientem. Określa cechy biznesowe produktu, akceptuje lub odrzuca wykonane prace. Ustala zapotrzebowania klienta.
- **Scrum Master** – osoba odpowiedzialna za projekt. Wybiera cele i określa efekty pracy. Jest odpowiedzialny za wprowadzanie w życie wartości i praktyk Scrum.
- **The Team** – osoby odpowiedzialne za utworzenie produktu. Nie obejmuje żadnej tradycyjnej roli inżynierii oprogramowania, takiej jak programisty, projektanta, testera, bądź architekta.

Cykl wytwarzania produktu podzielony jest na **Sprinty (Iteracje)**. Sprint jest najmniejszą jednostką czasową w Scrumie. W jego obrębie następuje planowanie, realizacja i podsumowanie prac projektowych. Każdy sprint jest zamkniętą całością i wnosi coś nowego do systemu. Czas każdego Sprintu powinna być stała dla całego projektu.

W obrębie Sprintu występują następujące zdarzenia:

Planowanie Sprintu(Sprint Planning) - Określenie, w jaki sposób będą przebiegały prace w czasie całej iteracji.

Codzienne spotkania (Daily Scrum Meeting) - Krótkie spotkania zespołu podczas trwania danej iteracji.

Spotkanie przeglądowe Sprintu (Sprint Review Meeting) - Omówienie stworzonej funkcjonalności.

Ostatnią rzeczą, o której warto wspomnieć to **Artefakty,** czyli elementy, które opisują pracę wykonywaną podczas trwania projektu.

- **Backlog produktu** – lista funkcjonalności, które musi posiadać produkt. Odpowiedzialny za nią jest właściciel produktu.
- **Backlog sprintu** – lista zadań, które mają zostać wykonane przez zespół w danym sprincie.
- **Przyrost** – wynik każdego ze sprintów.
- **Wykres wygaszania** – wykres przedstawiający ile pozostało jeszcze do wykonania zadań względem czasu trwania danego sprintu.

1.5 Cel dokumentu.

Po zapoznaniu się z wiedzą teoretyczną dotyczącą zarządzaniem projektami IT oraz poznaniem metodyk cyklu życia oprogramowania, wykorzystamy tą wiedzę w praktyce korzystając z narzędzi firmy Microsoft: Visual Studio Team Foundation Server wraz z Visual Studio 2013.

2. Praca z Visual Studio Online.

Zanim rozpoczniemy pracę z narzędziami MS, powiemy sobie, czym jest TFS i jak wpływa na zarządzanie projektem IT.

Duża część programistów używa Team Foundation Servera, jedynie jako repozytorium plików, który może być zarządzany przez wielu użytkowników. Oczywiście TFS nie ogranicza się tylko do tego. Team Foundation Server pozwala sprawować kontrolę nad wersjami aplikacji oraz pozwala np. na tworzenie automatycznych buildów. Trzeba tu również wspomnieć o, że TFS wspomaga zarządzanie projektem przy użyciu metodyk zwinnych.

Tytuł tego rozdziału brzmi „Praca z Visual Studio Online”, należy więc krótko wytłumaczyć co to jest. Microsoft oferuje swoim użytkownikom dwie metody planowania i zarządzania projektem poprzez tworzenie serwera TFS. Pierwsza to instalacja aplikacji Microsoft Team Foundation

Server w wersji Express(ograniczona do 5 użytkowników) lub w wersji zwykłej. Po instalacji tworzony jest nasz własny serwer lokalny, w którym możemy zarządzać naszym projektem. Drugą metodą i nią właśnie się zajmiemy to korzystanie z Visual Studio Online, czyli serwera postawionego w chmurze MS. VS Online zapewnia wszystkie funkcjonalności Team Foundation Servera bez instalowania jakichkolwiek aplikacji i konfigurowania serwera. Obie te metody zapewniają możliwość połączenia z Visual Studio, ale nie tylko. Można działać również na innych narzędziach developerskich np. Eclipse. Do połączenia z Visual Studio Online potrzebujemy konta w bazie Microsoftu. Możemy je założyć, korzystając z:

http://www.visualstudio.com/products/visual-studio-online-overview-vs

2.1 Microsoft Solution Framework

„Microsoft Solutions Framework (MSF) to zdyscyplinowane podejście do projektów technologicznych bazujące na zdefiniowanym zbiorze zasad, modeli, dyscyplin, pomysłów, wskazówek i sprawdzonych praktyk Microsoftu”[1]

„Microsoft Solutions Framework zwane jest framework’iem (strukturą) zamiast metodologią z określonych powodów. W przeciwieństwie do metodologii, MSF zapewnia giętką i skalowalną strukturę, która może być przystosowana do potrzeb wielu projektów (niezależnie od wielkości i złożoności) by planować, budować i wdrażać rozwiązania."[1]

Dlaczego mówimy o Microsoft Solutions Framework tworząc aplikację przy pomocy SCRUMA? Ponieważ planując, tworząc i śledząc proces tworzenia oprogramowania będziemy używać jednostek roboczych (Work Itemów), których definicja jest taka sama, jak artefaktów przy użyciu SCRUMA.

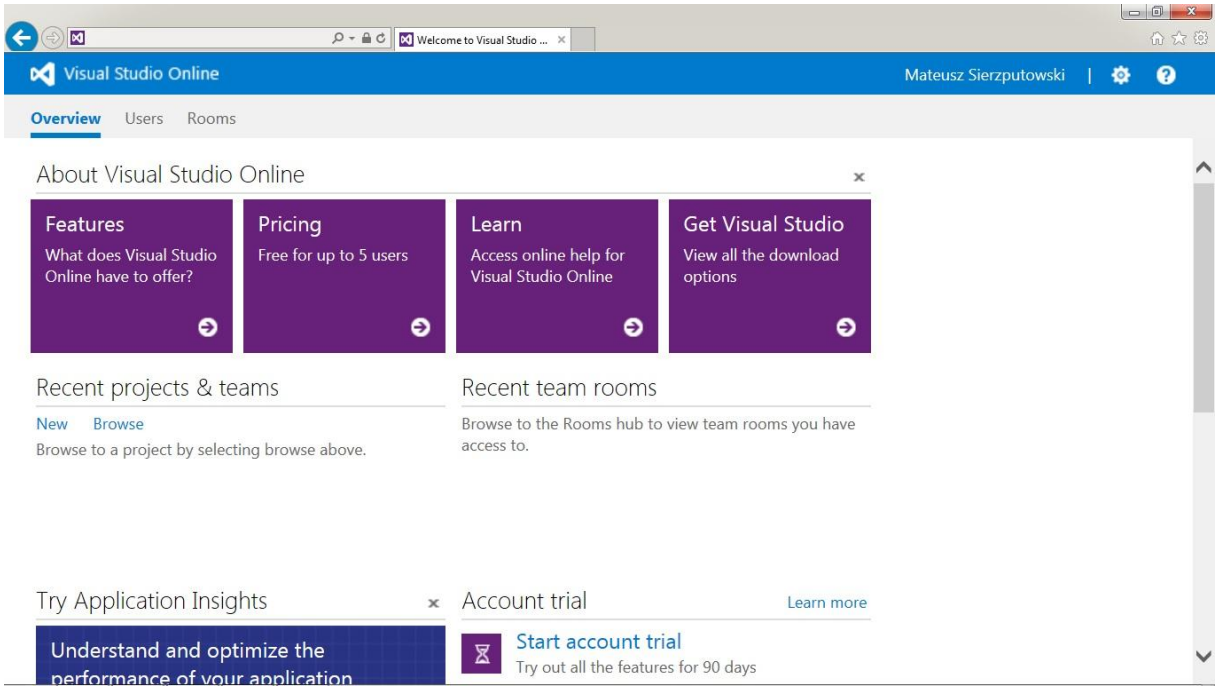
MSF w najnowszej wersji oferuje nam 6 typów jednostek roboczych:

- User Story – Scenariusz, prezentowany w postaci historii użytkownika.
- Bug – Błąd, prezentowany w postaci historii użytkownika.
- Test Case – Przypadek Testowy, powstający na bazie scenariusza lub błędu.
- Task – Zadanie, które zostaje zlecone zespołowi.

- Impediment – Zagrożenie, określające możliwe przeszkody, ryzyka obecne w projekcie.
- Shared Steps – Wspólne kroki, możliwość utworzenia i użycia tego samego zestawu kroków dla różnych przypadków testowych np. (rejestracja, logowanie).

W dalszej części zobaczymy, w jaki sposób jednostki robocze są ze sobą powiązane i jaki mają wpływ na planowanie i zarządzanie projektem IT.

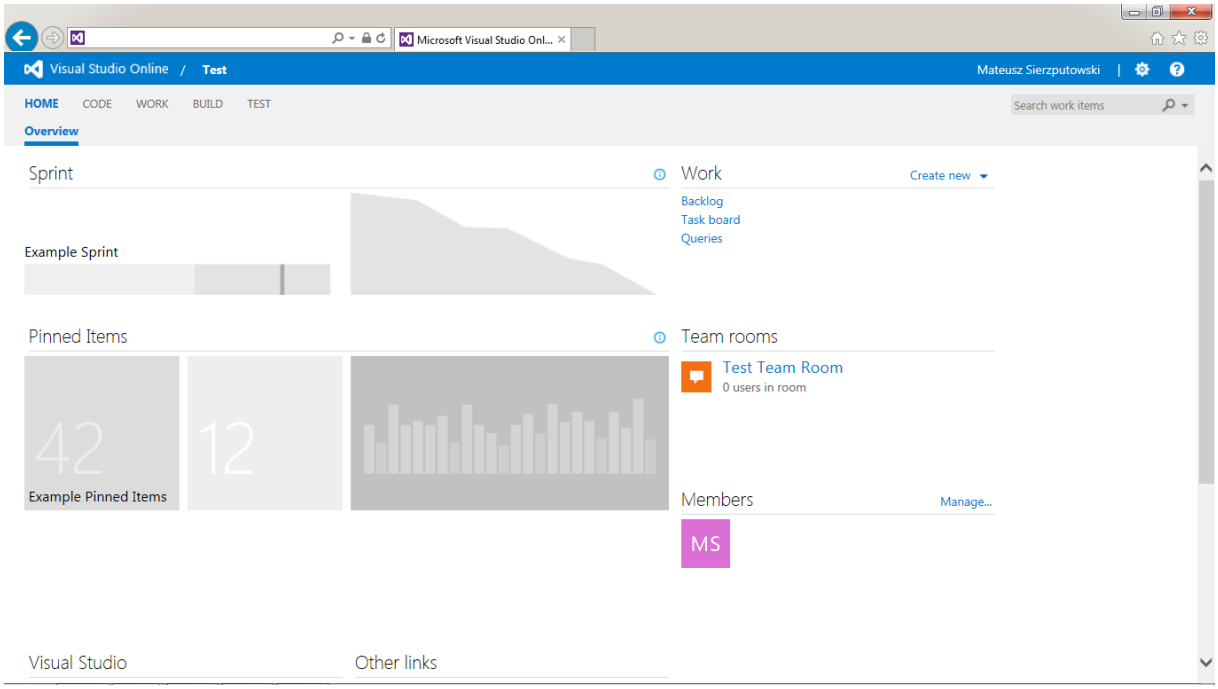
2.2 Tworzenie projektu.



Rysunek 2Okno główne VS Online

Projekt tworzymy klikając w przycisk „New”. Wpisujemy nazwę projektu, dodajemy krótki opis, pozostawiamy „Microsoft Visual Studio Scrum 2013” w Process Template i ustawiamy kontrolowanie wersji przy pomocy „Team Foundation Version Control”. Po krótkiej chwili oczekiwania klikamy Navigate to project , po czym zostaniemy przeniesieni do okna głównego naszego projektu.

2.3 Okno główne projektu.



Rysunek 3Okno główne projektu.

Pokrótce opiszę, to co widać na powyższym zrzucie.

Zaczynając od góry widzimy cztery zakładki.

Zakładka Home pokazuje nam najważniejsze rzeczy dotyczące projektu, jak czas iteracji, wskaźnik wypalenia.

Zakładka Code pozwala zobaczyć drzewo projektu oraz changelog, w którym można sprawdzić, czas i zawartość kolejnych commitów dodanych przez zespół.

Zakładka Work pozwala na planowanie i zarządzanie naszym projektem zgodnie z wybraną przez nas metodyką.

Zakładka Build pozwala nam na tworzenie definicji buildów.

Zakładka Test pozwala planować testy naszej aplikacji.

Do tego mamy dostęp do Team Roomu, w którym mamy możliwość komunikowania się z naszym zespołem. Najważniejsza jest jednak możliwość dodawania i przeglądania przypisanych do nas oraz nieprzypisanych jednostek roboczych.

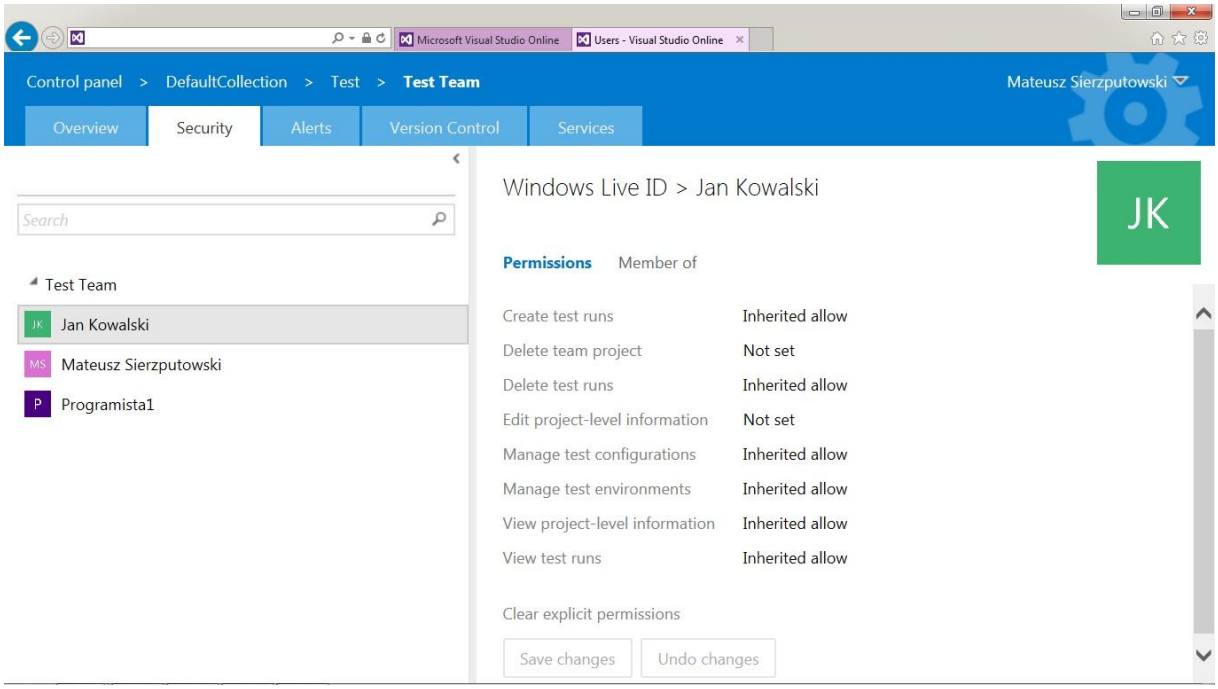
2.4 Dodawanie członków zespołu.

Dodawanie zespołu projektowego odbywa się poprzez wybranie opcji „Manage…" w oknie głównym aplikacji.

Wybieramy opcję „Add User" i podajemy adres mailowy naszego przyszłego członka zespołu. Należy pamiętać, o tym że adres mailowy powinien być powiązany z kontem Microsoft, żeby ta osoba mogła korzystać z VS Online. Na podany adres mailowy zostaje wysłana wiadomość z zaproszeniem projektu. Klikamy w podany link i to wszystko.

2.5 Administracja zespołem projektowym.

Bardzo przydatna opcją jest możliwość administracji naszym zespołem. VS Online oferuje możliwość dodawania członków zespołu do domyślnie utworzonych podgrup projektowych, jak i także określenie uprawnień dla poszczególnych członków. Dodatkową opcją jest możliwość ustawienia dni pracy.



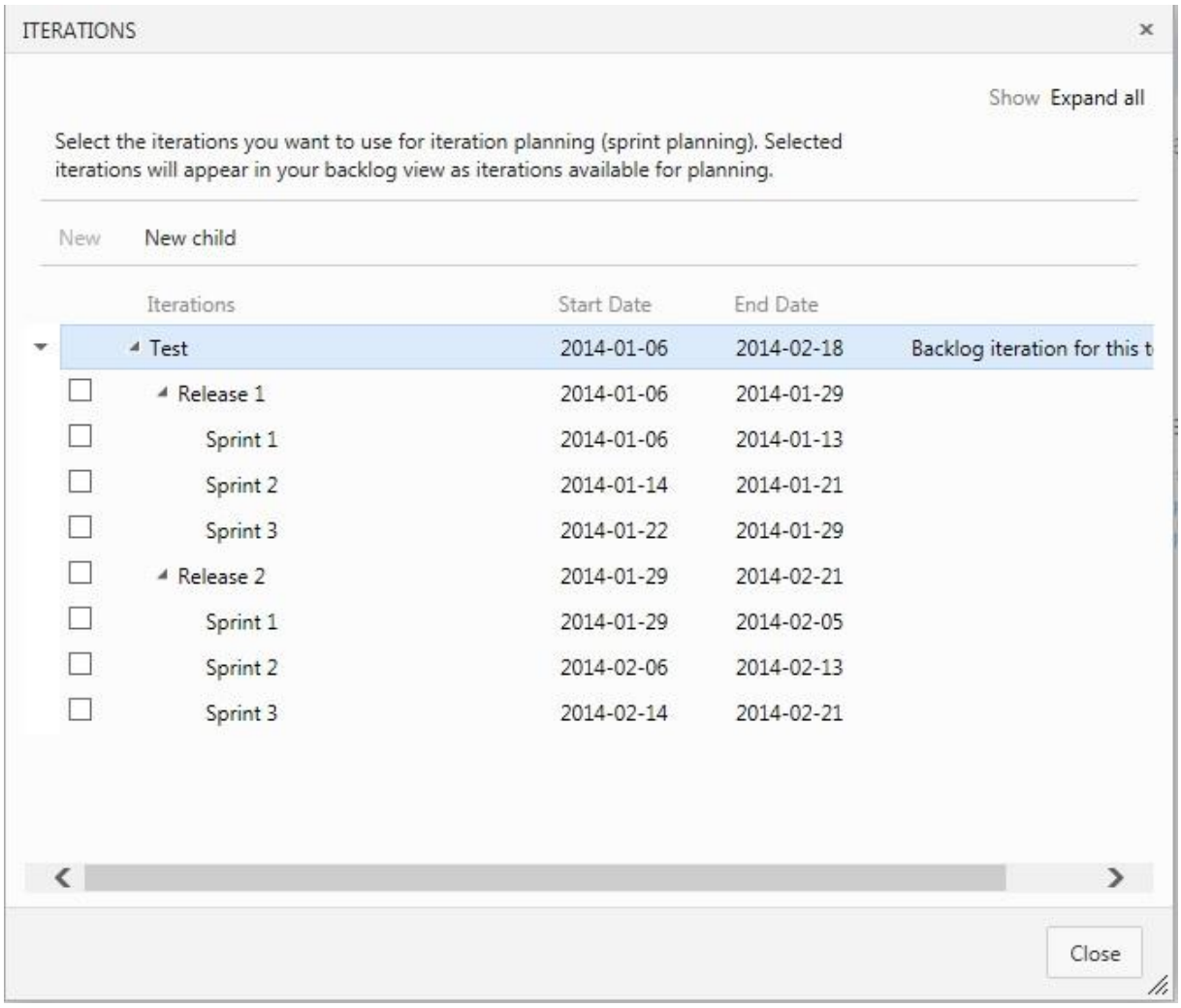
Rysunek 4Okno nadawania uprawnień.

2.6 Planowanie iteracji

Mając już utworzony projekt i dodany zespół projektowy do właściwej części projektowania projektu bazując na metodyce SCRUM. Dzielimy, więc projekt na Sprinty i grupujemy je odpowiednio w Release'y. Jest to przydatne, gdy na przykład Sprinty to poszczególne części jakiegoś modułu, a koniec Release'a oznacza ukończenie danego modułu.

W oknie głównym projektu klikamy na „Configure schedule and iterations".

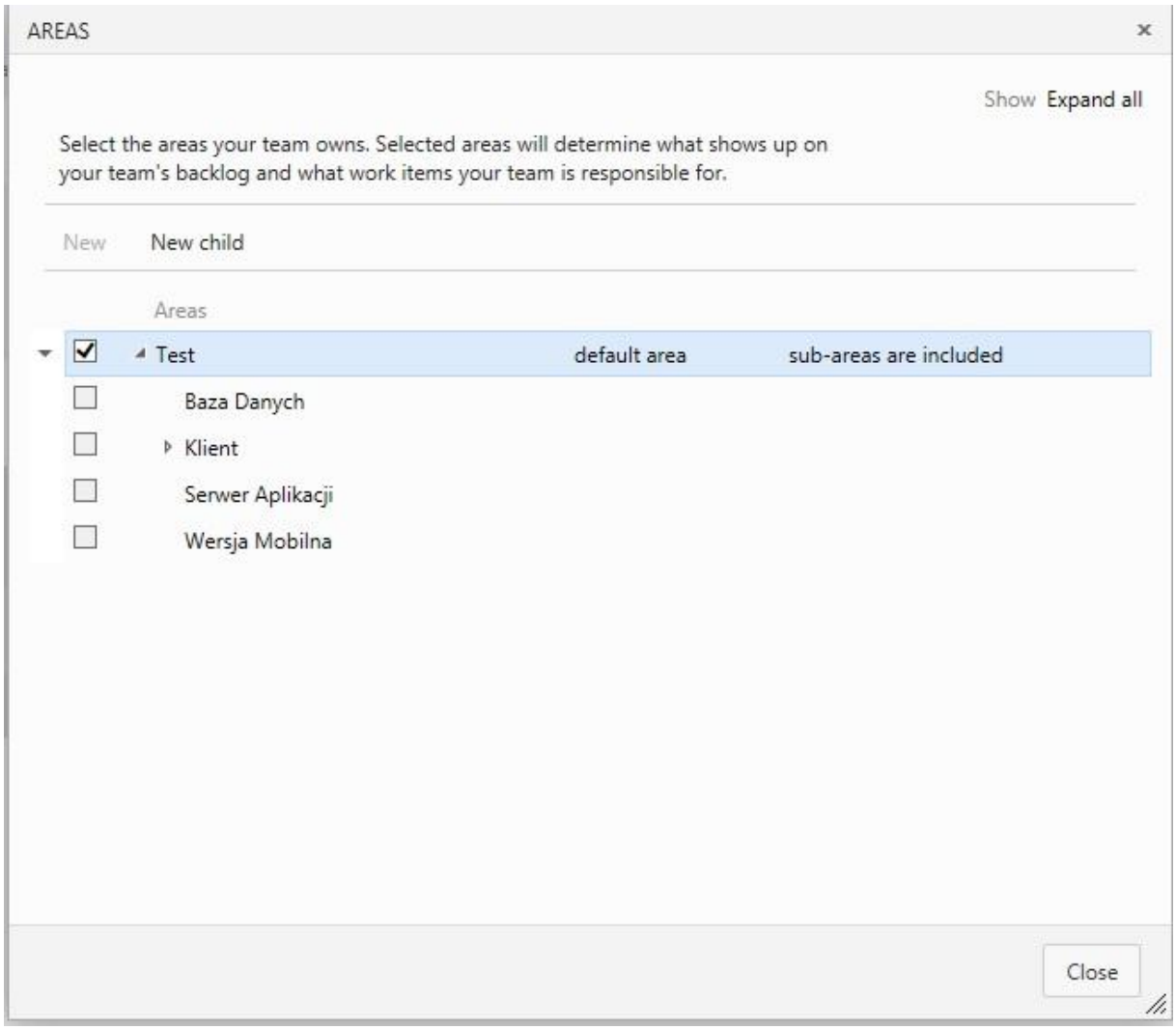
Wybieramy liczbę Sprintów oraz Release'ów i ustalamy daty początkowe i końcowe. Obsługa jest bardzo prosta i polega na naciskaniu prawym przyciskiem myszy na dany Sprint lub Release i wybieranie interesującej nas opcji.



Rysunek 5Okno podziału na iteracje

2.7 Planowanie obszarów.

Drugą rzeczą według której podzielimy nasz projekt są obszary. Obszary są to grupy, opisujące nasze funkcjonalności np. cele biznesowe albo tworzone moduły. Obszary możemy zagnieżdżać na kilka poziomów, wszystko zależy od naszego projektu. Planowanie obszarów odbywa się poprzez naciśnięcie „Configure Work Areas”. Obsługa jest identyczna, jak przy tworzeniu iteracji.



Rysunek 6 Okno planowania obszarów.

2.8 Zarządzanie Product Backlogami.

Product Backlog (User Story) jest to lista wymagań stawianych wobec oprogramowania, które tworzymy. Jest to najważniejsza jednostka robocza w TFSie. Zadania tworzymy właśnie wzorując się na historii

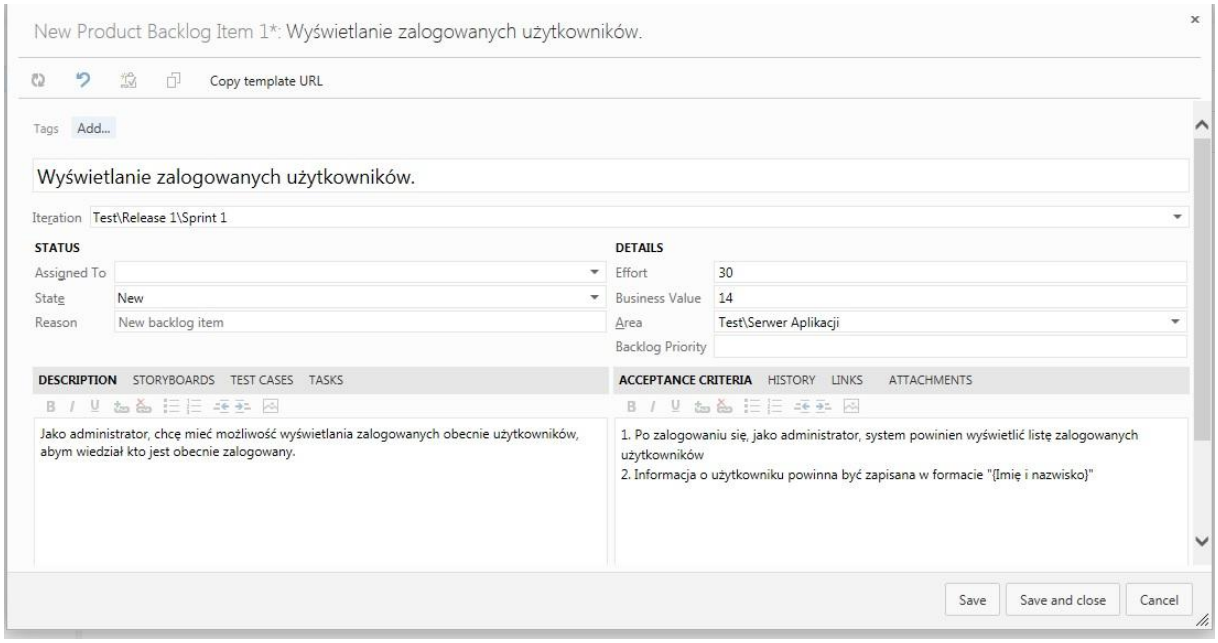
użytkownika. Opisuje ona wartość biznesową, która ma zostać dodana do projektu.

Do opisywania Product Backlogów, przyjęto odpowiednie nazewnictwo, które zapewnia prostotę zrozumienia przez klienta, jak i samego twórcę oprogramowania i wygląda następująco:

JAKO <rola, osoba> **CHCĘ** <działanie, funkcjonalność>, **ABY(W CELU)** <uzasadnienie biznesowe, wyrażone jako zaleta lub korzyść>.

Historię użytkownika tworzymy, przechodząc do zakładki Work. Wybieramy „Backlog Items" i dodajemy nowy Product Backlog, klikając w przycisk „Add".

Pojawi nam się nowe okno:



Rysunek 7Okno tworzenia Product Backlog Itemu

Omówię główne elementy:

- Iteration – Iteracja, w której wykonany dany Product Backlog.

- Assigned To – osoba odpowiedzialna za realizację historii użytkownika.

- State – aktualny status.

- Reason – powód dokonania akcji.

- Effort –wysiłek włożony w wykonanie historii użytkownika wyrażony w Story Pointach, co będą oznaczać Story Pointy to zależy tylko i wyłącznie od ustaleń w projekcie.

- Business Value – wartość biznesowa, jaką niesie dana historyjka.

- Area – obszar, do którego należy dany Product Backlog.

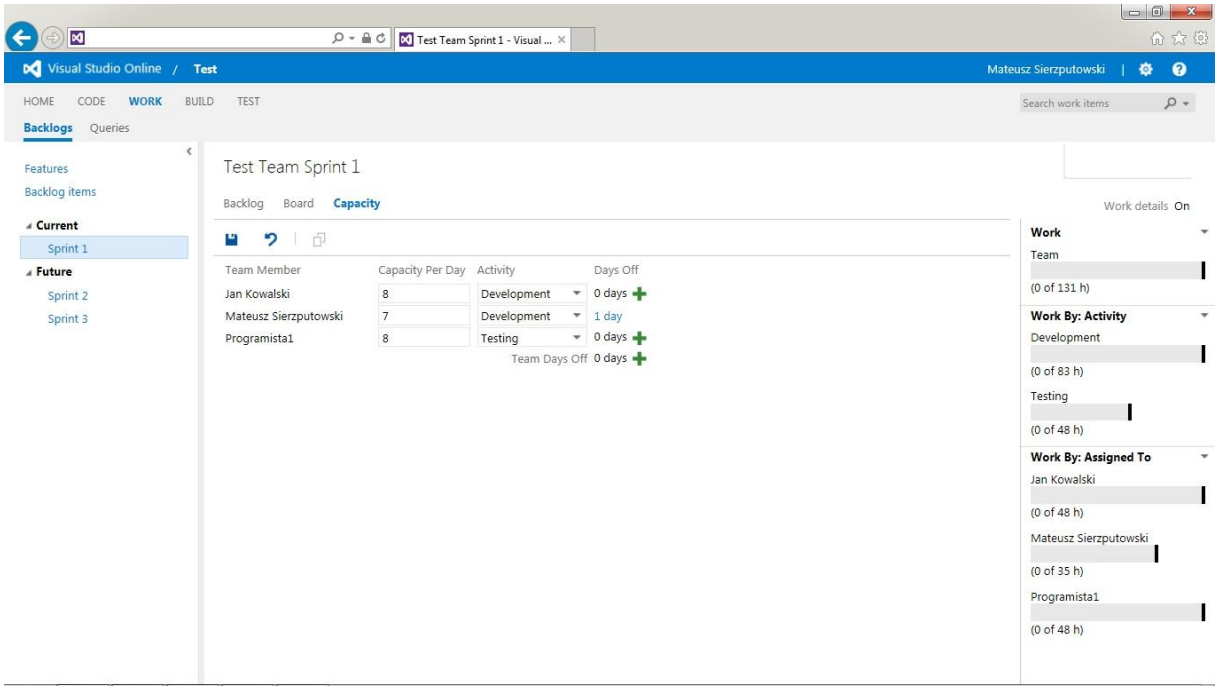
- Description – opis danej historyjki.

- Acceptance Criteria – Warunki, które muszą zostać spełnione, w celu akceptacji danego zadania.

Po utworzeniu Product Backlogu klikamy na „Save and Close" i automatycznie zostany on dodany do okna z Backlogami i przypisany do odpowiedniej iteracji.

2.9 Określanie pojemności zespołu

Pojemność zespołu, jest to ilość godzin, jaką może spędzić zespół nad projektem w ramach jednej iteracji.



Rysunek 8 Ustalanie pojemności zespołu.

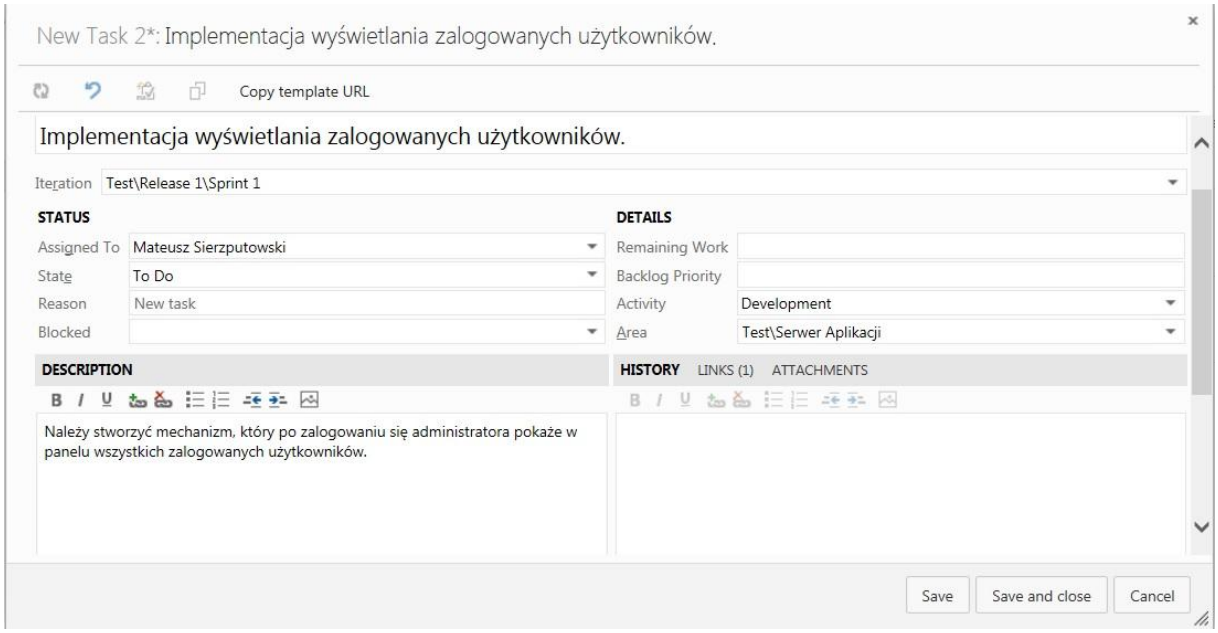
Jak widać, możemy wybrać tutaj pojemność dzienną poszczególnych członków zespołu, rodzaj aktywności oraz ilość dni, w czasie których dana osoba jest na urlopie. Po wybraniu godzin, automatycznie po prawej stronie pojawia się pojemność zespołu ze względu na różne kryteria. Ustalenie pojemności jest bardzo ważne, ponieważ dzięki temu wiemy ile

zespół może przeznaczyć czasu na daną iteracją, co trzeba brać pod uwagę przy tworzeniu i przydzielaniu zadań.

2.10 Tworzenie zadań.

W TFS przyjęto, że zadania grupowane są w historie użytkownika opisujące konkretny przypadek do zrealizowania. Co oznacza, że w celu realizacji musimy przypisać do historii przynajmniej jedno zadanie.

W celu utworzenia nowego zadania w zakładce „Work" wybieramy interesującą nas iteracją, następnie klikamy na zakładkę „Board". Pojawiają się utworzone przez nas Product Backlogi. Ostatnim krokiem jest naciśnięcie przycisku „+". Otwiera się okno nowego zadania.



Rysunek 9 Okno Tworzenia nowego zadania.

Omówię podstawowe elementy:

- Assigned To – osoba, której przypisujemy dane zadanie.

- State – aktualny Status Zadania.

- Reason – powód dokonywanej akcji.

- Blocked – określenie, czy dane zadanie jest w tym momencie zablokowane (z jakiegoś powodu nie powinno być aktualnie robione)

- Remaining Work – pozostały czas do zakończenia zadania.

- Backlog Priority – priorytet zadania.

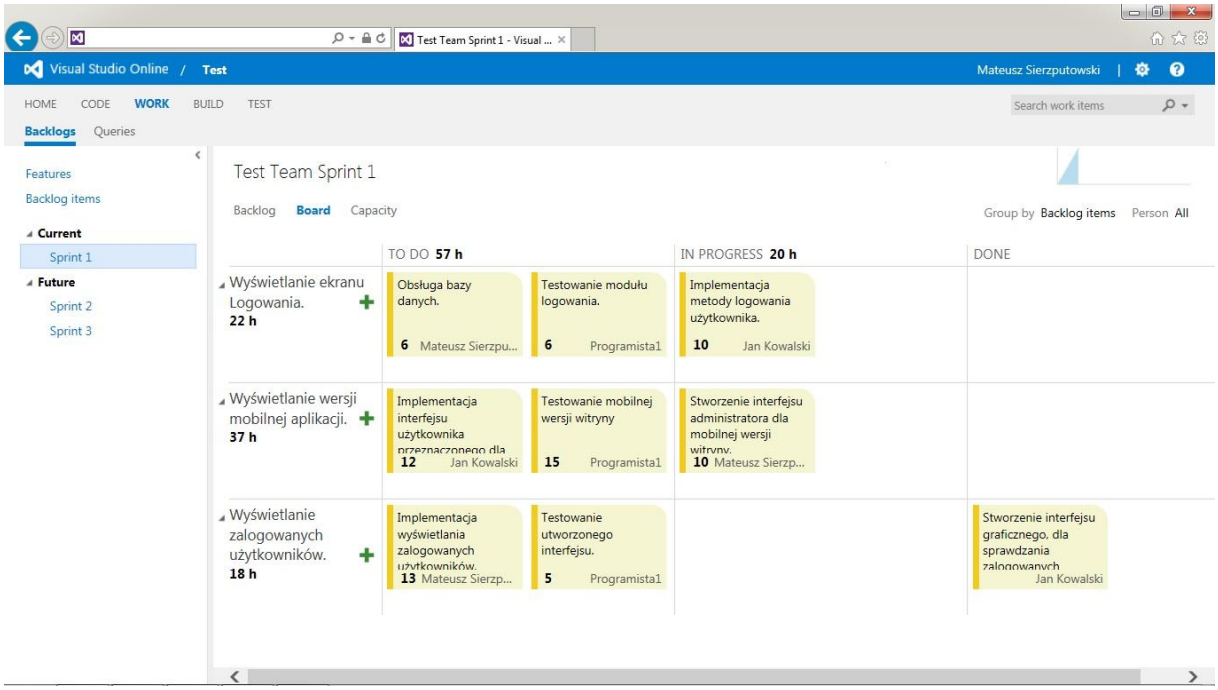
- Acitivity – rodzaj czynności wykonywanej w zadaniu.

- obszar, do którego należy dane zadanie.

Po dodaniu zadania pojawi się ono w zakładce „Board”.

2.11 Zarządzanie realizacją sprintu.

Zarządzanie realizacją sprintu odbywa się w zakładce „Board”.



Rysunek 10 Realizacja Sprintu.

Zadania dzielimy na trzy grupy:

- To Do – nie rozpoczęte.

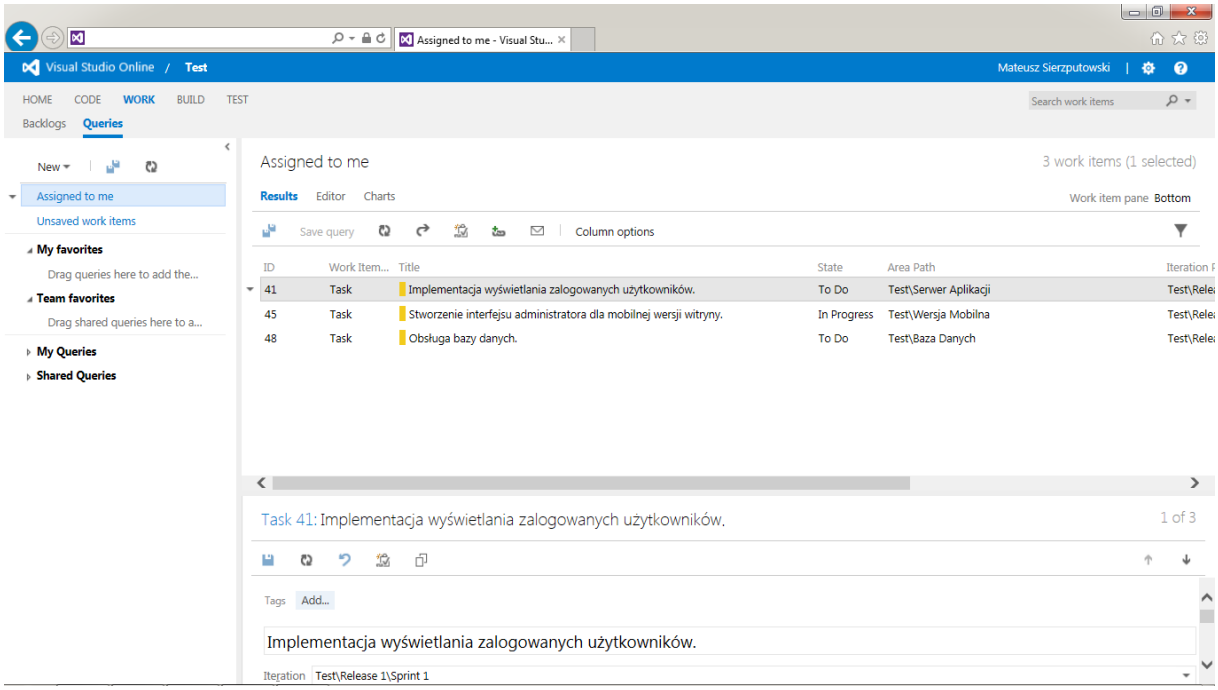
- In Progress – w trakcie robienia.

- Done – wykonane.

Obsługa jest bardzo prosta, wystarczy przeciągnąć odpowiednie zadanie do odpowiedniej grupy. Mimo, że sam wygląd i obsługa są proste, doskonale widać, które zadanie zostało wykonane, a które jest w trakcie robienia. Jest to bardzo pomocne narzędzie.

2.12 Zarządzanie jednostkami roboczymi.

Zarządzanie jednostkami roboczymi odbywa się poprzez naciśnięcie przycisku „Queries” w oknie głównym projektu.



Rysunek 11 Zakładka Queries.

Zakładka ta pozwala na szybki przegląd i edycję Work Itemów. Bardzo przydatna jest możliwość filtrowania Work Itemów, ze względu na czas wykonania, przynależność do danego członka zespołu, czy też do danej podgrupy projektowej. Można również tworzyć bardzo różne wykresy, które wizualizują nasz zaprojektowany projekt.

2.13 Zakończenie.

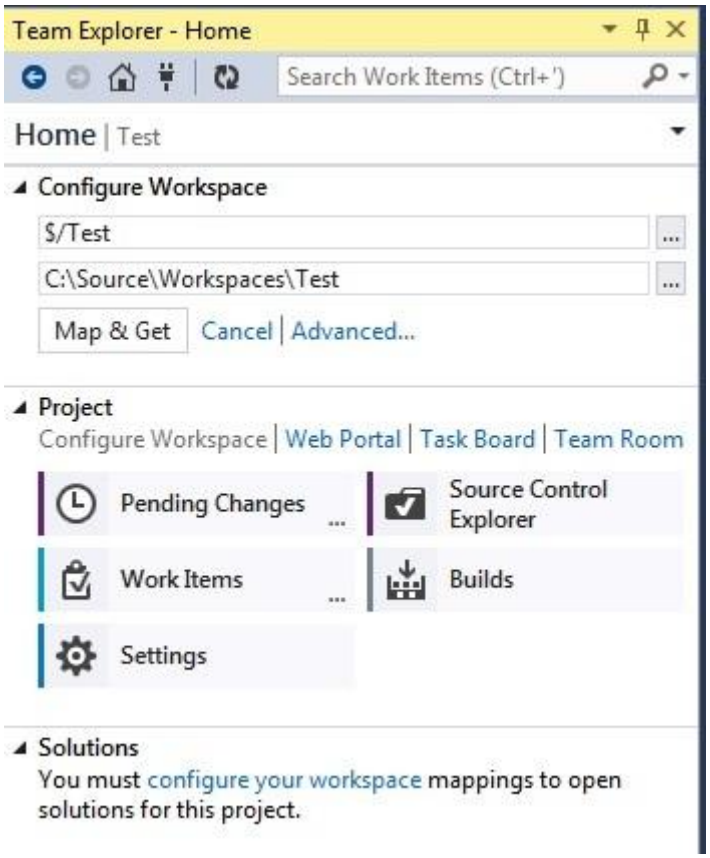
Udało nam się przejść przez zaprojektowanie naszego projektu zgodnie z metodyką SCRUM. Pokazane zostały prostota i możliwości Team Foundation Servera, akurat w tym przypadku opartego na chmurze Microsoft, czyli Visual Studio Online.

Następna część to praca z Visual Studio 2013, oparta na zaplanowanym przez nas projekcie.

3. Praca z Visual Studio 2013.

3.1 Łączenie z TFS

Żeby podłączyć się do naszego Team Foundation Servera, z listy rozwijanej „View” klikamy „Team Explorer”. Zostaniemy poproszeni o login i hasło do naszego konta Microsoft. Następnie należy kliknąć na „Configure workspace” i podać ścieżkę, w której będzie przechowywany nasz projekt.

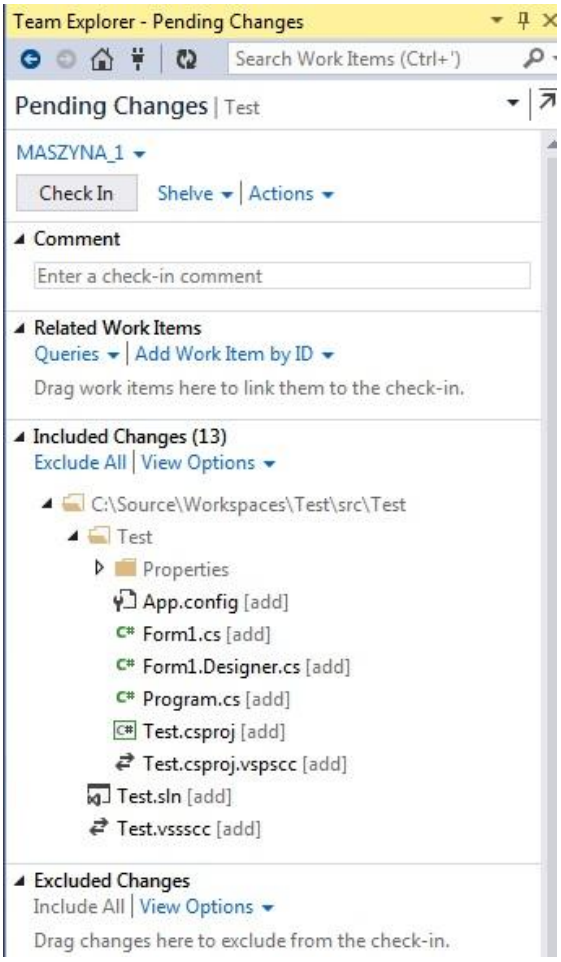


Rysunek 12 Team Explorer

Klikamy Map & Get i chwilę czekamy. W tym momencie jesteśmy na etapie tworzenia projektu, więc wybieramy z Menu głównego „New Project”. Wybieramy typ projektu, podajemy wcześniej zmapowaną

ścieżkę i najważniejsza część to zaznaczenie opcji „Add to Source Control”.

W tym momencie nasze pliki znajdują się tylko lokalnie na naszym komputerze. W celu wrzucenia zmian na serwer wybieramy w Team Explorerze zakładkę „Pending Changes” i klikamy „Check In”



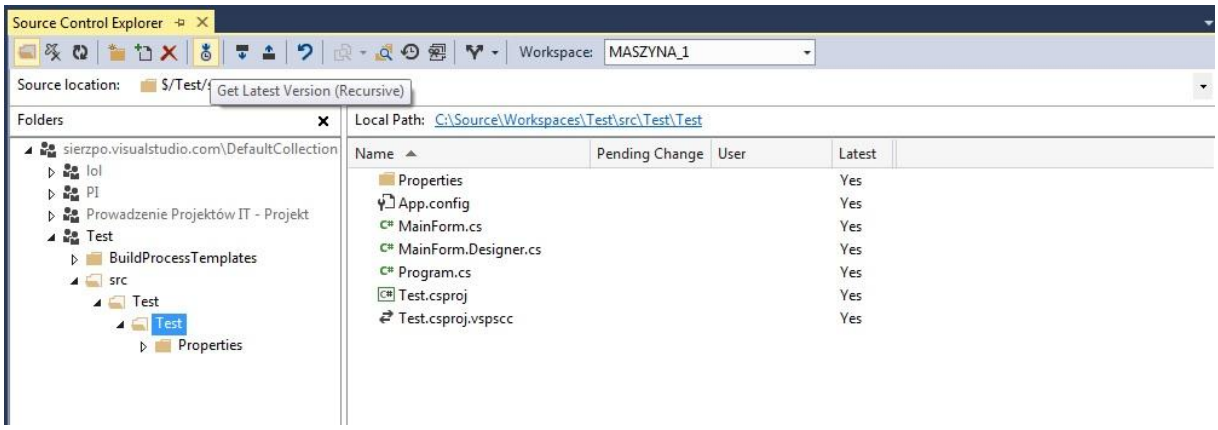
Rysunek 13 Pending Changes

Dodanie Commita możemy opatrzyć w komentarz, w którym opiszemy, co dodajemy do TFS’a. Możemy również wybrać, które pliki chcemy, a których nie chcemy dodawać wraz z naszym Commitem.

Po chwili oczekiwania dostajemy komunikat, że operacja została zakończona powodzeniem. Aktualne changesety sprawdzamy np. w Visual Studio Online w zakładce „Code”.

3.2 Pobranie zmian z serwera.

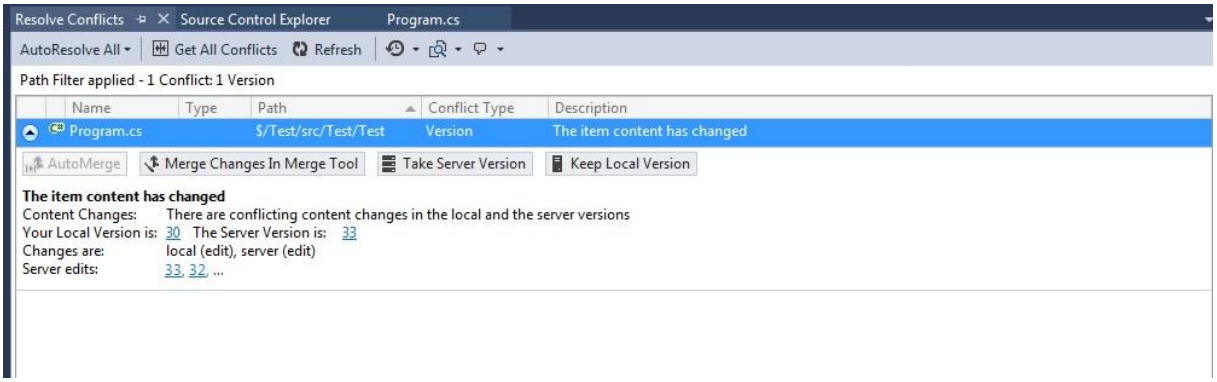
Przypuśćmy, że któryś z członków naszego zespołu dodał jakieś zmiany w projekcie. Musimy więc pobrać najnowszą wersję z serwera. W tym celu wybieramy z Team Explorera „Source Control Explorer”, po pojawieniu się nowego okna wybieramy „Get Latest Version”. Jeśli posiadamy najnowszą wersję z repozytorium otrzymamy odpowiedni komunikat, jeśli zaś nie najnowsza wersja zostanie pobrana z repozytorium.



Rysunek 14 Source Control Explorer

3.3 Rozwiązywanie konfliktów.

Wyobraźmy sobie sytuację, gdy dwóch programistów pracuje nad tym samym plikiem i nadpisują sobie wzajemnie pracę. Chcemy wysłać poprawiony kod na repozytorium, jednak dostajemy informację, że inna wersja tego pliku znajduje się już na repozytorium i automatycznie pojawia się okno rozwiązywania konfliktów.

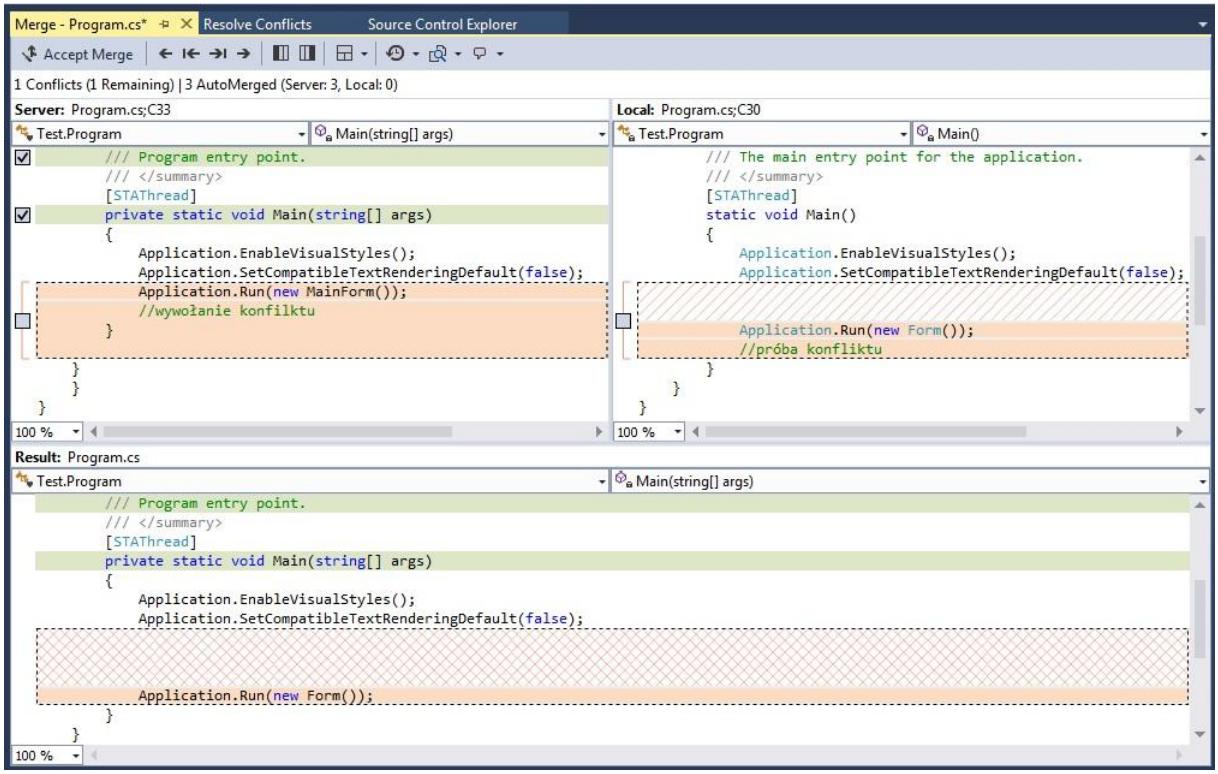


Rysunek 15 Okno Resolve Conflicts

Visual Studio oferuje nam trzy sposoby rozwiązania konfliktu:

- Take Server Version – porzucenie własnych zmian i pobranie wersji znajdującej się na serwerze.
- Keep Local Version – nadpisanie wersji znajdującej się na serwerze, zmianami które mamy zapisane lokalnie.
- Merge Changes In Merge Tool – ręczne rozwiązanie konfliktu przy pomocy narzędzia Merge Tool.

Przyjrzymy się bliżej ostatniemu sposobowi.



Rysunek 16 Merge Tool

Po lewej stronie widzimy wersję serwerową pliku, czyli zmiany wprowadzone przez członka naszego zespołu. Po prawej stronie widzimy zaś naszą lokalną wersję. W dolnej części jest wersja pliku, która jest efektem naszego scalenia. Praca z narzędziem polega na tym, że zaznaczając checkboxy przy odpowiednich linijkach kodu, powodujemy dodanie ich do wersji scalonej programu.

Po ustaleniu finalnej wersji klikamy „Accept Merge”, a następnie używając „Check-in” dodajemy nową wersję do repozytorium.

3.4 Praca z jednostkami roboczymi.

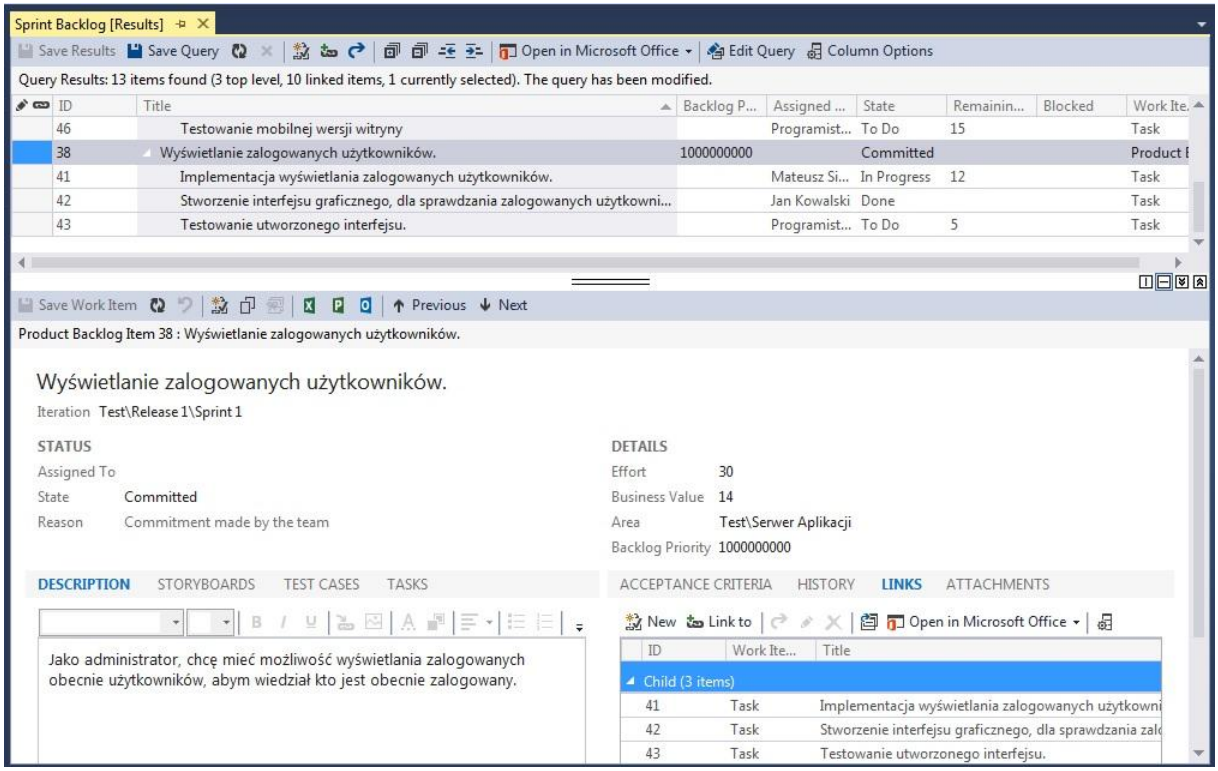
Z poziomu Visual Studio również mamy możliwość tworzenia, edycji, przeglądania jednostek roboczych. Zanim jednak zaczniemy pracę z Work

Itemami, trzeba powiedzieć kilka słów na temat najpopularniejszych typów relacji pomiędzy jednostkami roboczymi w TFS'ie.

- „Parent-Child" – Relacja tworzona np. pomiędzy User Story – Task, czy dzielenie zadania na podzadania. Tworzona jest w celu utworzenia jakiejś hierarchii.
- „Related" – Relacja tworzona w celu pokazania, że dwie jednostki robocze są ze sobą w jakiś sposób powiązane.
- „Tested by" – Relacja wykorzystywana z jednostkami roboczymi typu Test Case.
- „Shared Steps" – Relacja wykorzystywana z jednostkami roboczymi typu Shared Steps.

Podczas pracy z Visual Studio Online utworzyliśmy szereg Product Backlogów i zadań. Wykorzystamy to do utworzenia pozostałych typów jednostek roboczych, w celu zobaczenia powiązań pomiędzy nimi.

Przechodzimy do „Work Items" w Team Explorerze, a następnie wybieramy z listy „Shared Queries" opcję „Sprint Backlog". Spowoduje to otwarcie w oknie głównym Visual Studio itemów, które utworzyliśmy w VS Online.

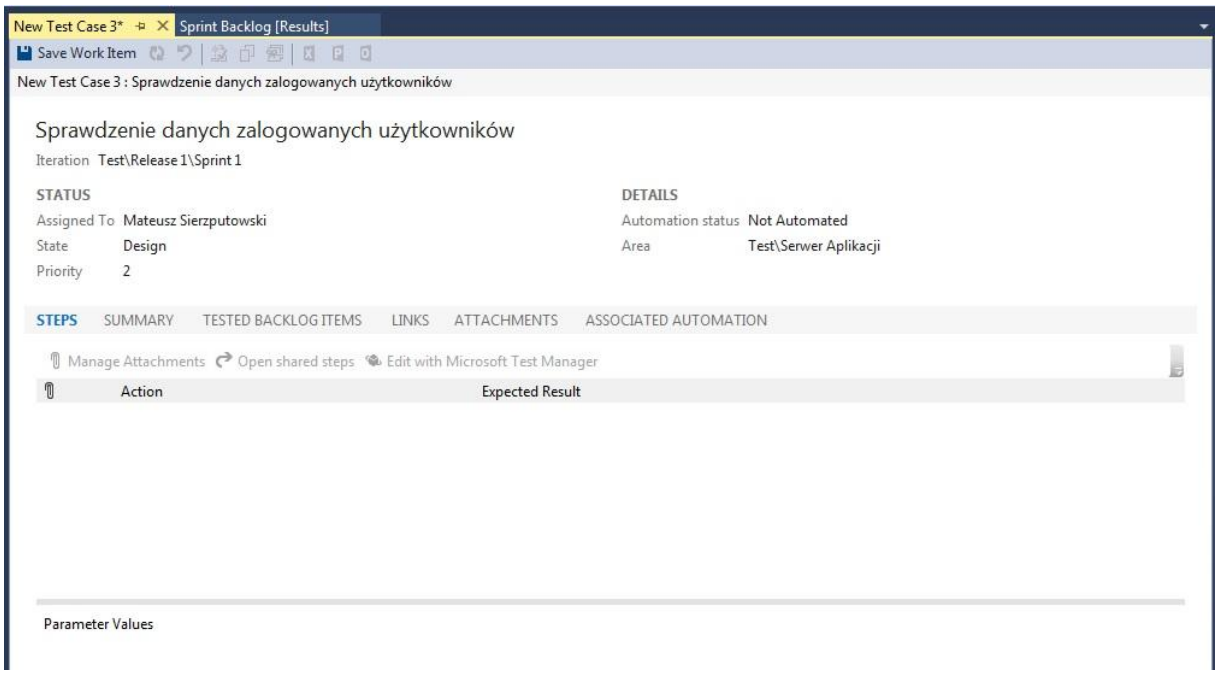


Rysunek 17 Sprint Backlog Window

Widzimy wszystkie historie użytkownika i zadania. Po wybraniu odpowiedniego Product Backlogu i kliknięciu na zakładkę „Links” widzimy wszystkie powiązane z nimi jednostki robocze. W tym przypadku są to trzy zadania. Jak widać relacja pomiędzy jednostkami to „Parent-Child”.

Teraz utworzymy Test Case dla naszego Product Backlogu.

Klikamy na zakładkę „Test Cases” i wybieramy „New”. Pojawi nam się nowe okienko, którym podajemy nazwę i ewentualny komentarz. Widzimy, że relacja pomiędzy elementami będzie relacją Tested by. Klikamy OK i automatycznie zostajemy przenoszeni, do naszego Test Case’a.



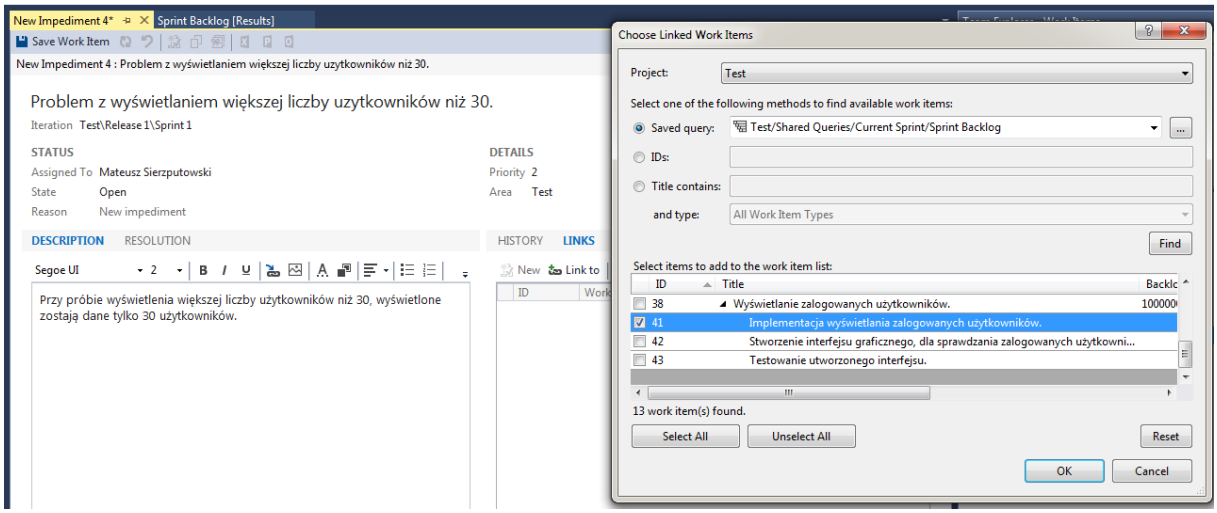
Rysunek 18 Dodawanie Test Case

Wypełniamy odpowiednie parametry i klikamy „Save Work Item”. Wracając do Sprint Backloga widzimy, że w historii użytkownika został dodany nasz Test Case, w relacji Tested by.

Przypuśćmy, że podczas wykonania zadania okaże się nie wiemy jak coś zrobić, bądź mamy jakieś problemy ze zrozumieniem problemu albo nie dostarczono nam odpowiedniej dokumentacji. W tym celu stworzymy Work Item typu Impediment, który będzie w relacji Related z naszym zadaniem.

W tym celu klikamy w Team Explorerze na listę „New Work Item” i klikamy na „Impediment”. Automatycznie pojawia się okno tworzenia

nowego Impediment'a. Wypełniamy podstawowe dane i klikamy na zakładkę „Links". Następnie wybieramy „Link To". Pojawia się okno, w którym wybieramy rodzaj relacji, w naszym przypadku jest to relacja „Related by". Jeśli pamiętamy Id naszego Taska to wpisujemy je, jeśli nie klikamy na „Browse…", wybieramy odpowiednie Query i klikamy „Find". Pojawią się wszystkie nasze Work Itemy.

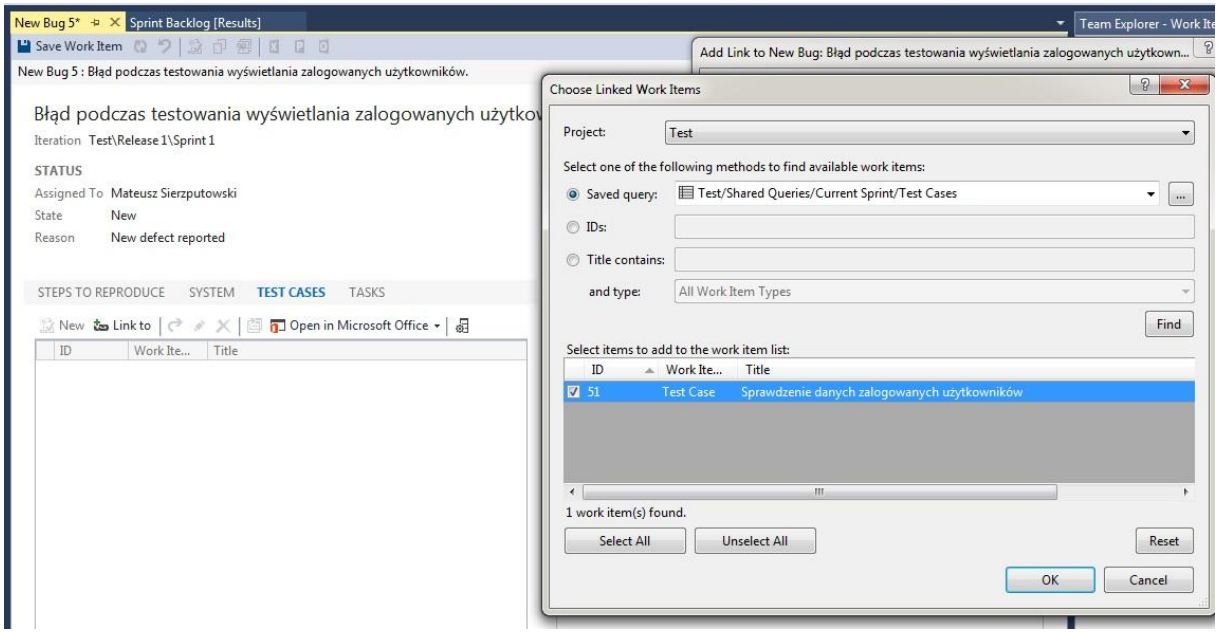


Rysunek 19 Dodawanie Zagrożenia

Po wybraniu odpowiedniego klikamy OK, następnie znów OK i na koniec Save Work Item.

Po kliknięciu w Sprint Backlogu na nasze zadanie, widzimy że w zakładce „Links" pojawiło się nowe powiązanie z dodanym przez nas Impedimentem.

Ostatnim krokiem będzie powiązanie naszego przypadku testowego, z jednostką roboczą typu Bug, w przypadku znalezienia błędu podczas fazy testowania. Klikamy w Team Explorerze na listę „New Work Item" i klikamy na „Bug". Wpisujemy potrzebne dane i w zakładce „Test Case" klikamy na „Link To". Działamy podobnie, jak w przypadku Impedimenta, ale tym razem wybieramy Query typu „Test Cases". Wybieramy odpowiedni przypadek testowy i klikamy OK, a następnie Save Work Item.



Rysunek 20 Dodawanie błędu.

Teraz osoba, do której został przypisany błąd może sprawdzić, jaki Test Case był wykonany, gdy pojawił się błąd. Od razu może sprawdzić, jaki Product Backlog był testowany tym scenariuszem oraz jakie Taski implementują dany Product Backlog. Oczywiście wiążą się z tym również Impediments, czyli zagrożenia, jakie niesie ze sobą implementacja danego zadania.

W ten sposób bardzo szybko możemy znaleźć przyczynę błędów i podjąć próbę rozwiązania ich.

3.5 Automatyczne buildy.

Tematykę automatycznych buildów rozpoczniemy od wytłumaczenia sobie terminu „Continuous Integration”. Ciągła integracja jest to praktyka wytwarzania oprogramowania polegająca na częstej i regularnej integracji bieżących zmian w kodzie do repozytorium.

Zdarza się, że programista dodając zmiany na repozytorium, nie sprawdzi czy program się kompiluje, nie wspominając już o sprawdzeniu funkcjonalności.

Programując nowe funkcjonalności należy pamiętać, aby zmiany dodawać do repozytorium często i regularnie, a nie dopiero po wykonaniu np. całego zadania, należy pamiętać też o dodawaniu jednoznacznych komentarzy.

Dobrą praktyką jest też częste pobieranie zmian z repozytorium, dzięki temu zmniejszymy możliwość ewentualnych konfliktów.

Tworzenie automatycznych buildów pozwala praktycznie na wyeliminowanie tych problemów.

Zbudujemy dwa typy buildów:

- Build, który będzie sprawdzał, czy wrzucony kod jest poprawny.
- Build, który będzie okresowo sprawdzał czy zachowana jest integracja projektu.

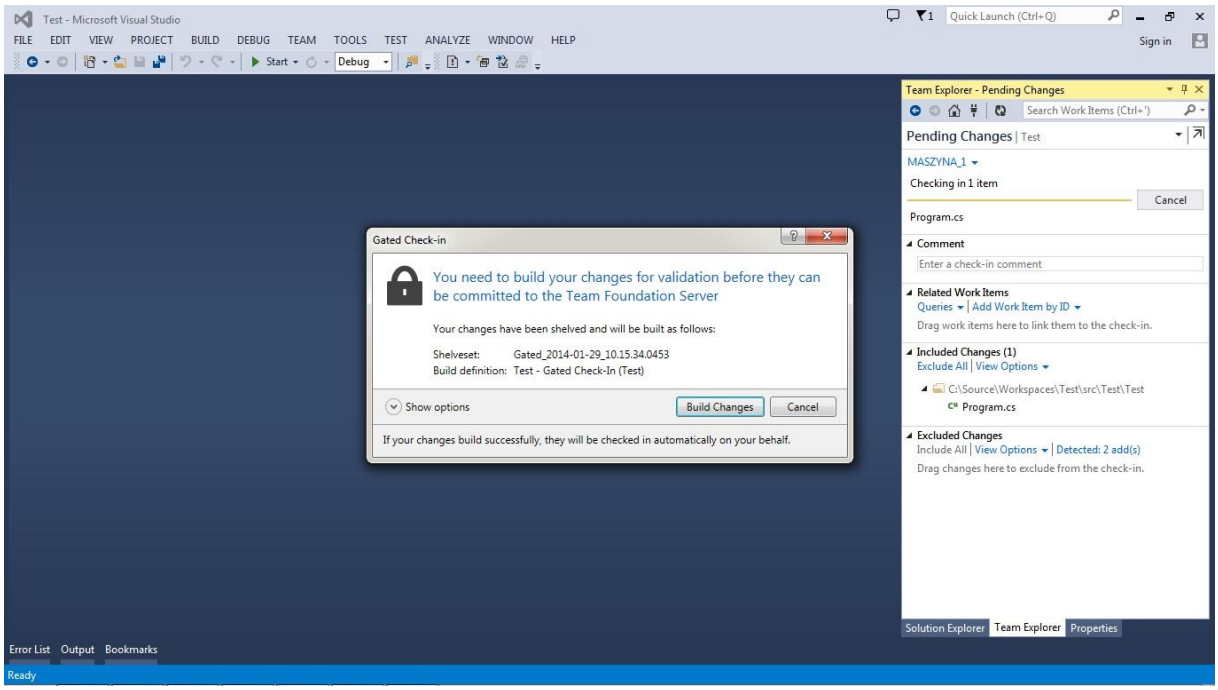
W celu utworzenia buildu, klikamy na „Builds” w Team Explorerze, a następnie „New Build Definition”. Pojawia się nowe okno, w którym tworzymy definicje builda.

Zajmiemy się teraz stworzeniem pierwszego buildu.

W zakładce „General” wpisujemy nazwę oraz ewentualny opis.

W zakładce „Trigger” wybieramy Gated Check-In.

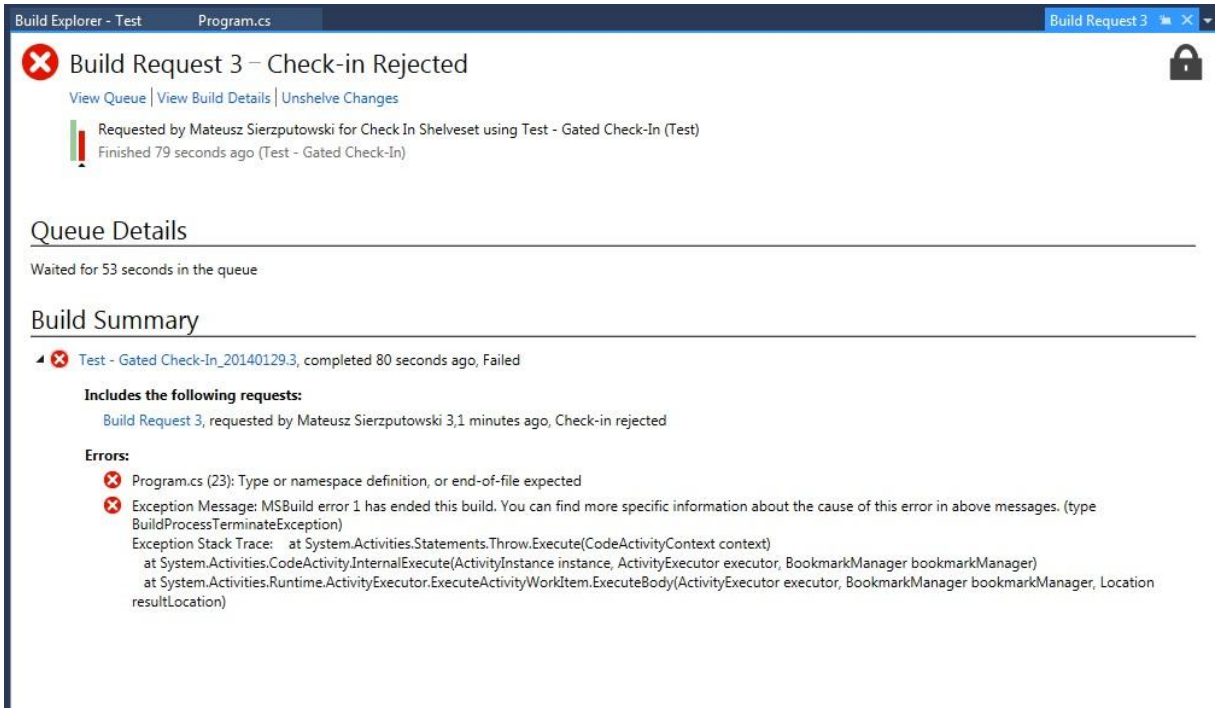
Zapisujemy build, dodajemy zmiany w naszym projekcie i klikamy przycisk „Check-in”.



Rysunek 21 Build Check-In

Pojawiło nam się nowe okno, informujące nas o tym, że TFS sprawdzi poprawność zmian i jeśli wszystko będzie poprawnie to zmiany zostaną wysłane do repozytorium. Klikamy „Build Changes”. Po dłuższej chwili kod

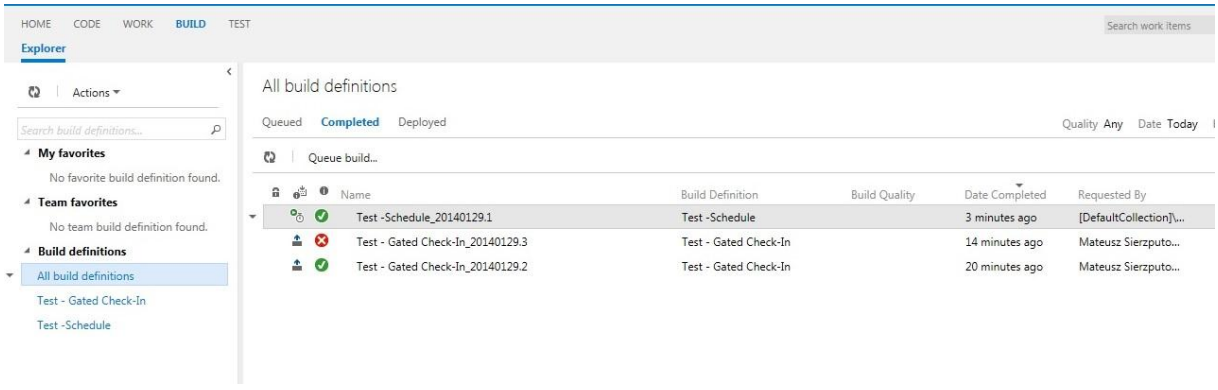
zostanie wysłany, bądź też otrzymamy błąd, że projekt nie może zostać wysłany na repozytorium. W czasie wykonywania buildu można sprawdzić postępy w Build Explorerze.



Rysunek 22 Informacja o błędzie.

Utworzymy teraz build okresowy. Podobnie, jak przy pierwszym buildzie klikamy „New Build Definition”, podajemy odpowiednią nazwę i w zakładce „Trigger” wybieramy opcję „Schedule” podając interesujące nas dni oraz godzinę i zaznaczamy opcję, aby build uruchamiał się, nawet gdy kod się nie zmienił. Zapisujemy i czekamy do odpowiedniej godziny.

Wszystkie wykonane buildy możemy sprawdzić z poziomu Visual Studio Online, klikając w zakładkę „Build”.



Rysunek 23 Buildy w VS Online

4.Podsumowanie.

Przedstawione powyżej możliwości TFS'a pokazują jego prostotę i moc w zarządzaniu projektami informatycznymi. Wszystkie najważniejsze informacje dotyczące naszego projektu mamy w zasięgu ręki. Lokalizacja i poprawa błędów stają się o wiele łatwiejsze. Jednak nie są to wszystkie możliwości Team Foundation Servera. TFS świetnie współgra z innymi aplikacjami od Microsoftu, jak Microsoft Excel, PowerPoint, czy Project.

5.Bibliografia

- Professional Team Foundation Server 2010, Ed Blankenship; Martin Woodward; Grant Holliday; Brian Keller, 2011, ISBN: 978-0-470-94332-8
- [1] Microsoft Solutions Framework Core Whitepapers, http://www.microsoft.com/msf
- http://channel9.msdn.com/
- http://msdn.microsoft.com/pl-pl/