POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

Konrad Jędrzejewski
Włodzimierz Dąbrowski

ZARZĄDZANIE PROJEKTAMI INFORMATYCZNYMI W METODYCE SCRUM

Draft

Warszawa 2012

1 Warszawa, 2012

STRESZCZENIE

Rozwój oprogramowania jest skomplikowanym zajęciem. Aby oprogramowanie to powstawało zgodnie z potrzebami klienta przy zastosowaniu odpowiedniej jakości stosuje się metodyki wspomagające zarządzanie projektami. Ich rozwój doprowadził do powstania tak zwanych zwinnych metodyk zarządzania projektami które są odpowiedzią na problemy dotyczące zwiększenia efektywności pracy, dostarczania produktu który klient naprawdę potrzebuje oraz dostarczania produktów o wysokiej jakości. Wśród takich metodyk jest metodyka Scrum która jest zbiorem zasad opisanych na zaledwie kilku stronach a które jak pokazują jej twórcy powoduje zwiększenie komunikacji w zespole a także między klientem. Została omówiona metodyka Scrum oraz jej praktyczne zastosowanie.

.

Spis treści

1. Wprowadzenie do zarządzania projektami

1.1. Czym jest zarządzanie projektem?

Oprogramowanie w dzisiejszych czasach wymaga od producentów prześcigania się w rozwiązaniach technologicznych, stabilności produktu oraz innowacyjności. Czasy, kiedy systemy informatyczne były małe i proste, należą do przeszłości. Obecnie systemy są złożone, a ich budowa często trudna, długa i nieuporządkowana. Przedsięwzięcie informatyczne często nie jest tylko wytworzeniem produktu jest to także jego utrzymanie co powoduje, że może ono trwać nawet wiele lat, można je porównywać do przedsięwzięcia wielkiej budowy, np. mostów. Statystyki pokazują [1] że w około 33% przypadków przedsięwzięcia informatyczne kończą się niepowodzeniem, natomiast następne 33% wymagają zwiększonego wykorzystania zasobów przeznaczonych na projekt. Dlatego też prowadzenie projektów wymaga odpowiedniego zarządzania.

Zadajmy sobie pytanie czym jest więc zarządzanie projektami?

Aby odpowiedzieć sobie na to pytanie najpierw powinniśmy się zastanowić czym jest projekt. Pozwoli nam to lepiej zrozumieć pojęcie zarządzania projektami oraz ułatwi zrozumienie pojęć zawartych w dalszej mojej pracy. Definicja mówi że:

> *Projekt to sekwencja niepowtarzalnych, złożonych i związanych ze sobą zadań, mających wspólny cel, przeznaczonych do wykonania w określonym terminie bez przekraczania ustalonego budżetu, zgodnie z założonymi wymaganiami". [2]*

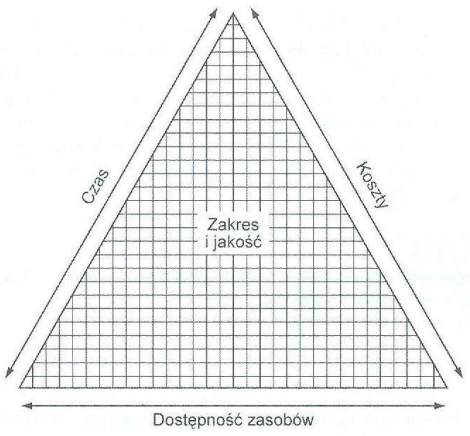
"

Interpretując powyższą definicję można stwierdzić, że projekt posiada pewne swoje atrybuty, które go charakteryzują. Tak więc zarządzanie projektem polega na takim kierowaniu przebiegiem projektu aby nie przekroczyć atrybutów które posiada dany projekt i aby cel projektu został osiągnięty. Niestety jak to w życiu bywa nie zawsze się to udaje. Opierając się na wynikach badań grupy The Standlish Group [1] projektów zakończonych

powodzeniem jest dość niewiele, porównaniu do ogólnej liczby projektów, dlatego też warto się zastanowić jakie są przyczyny niepowodzenia projektów.

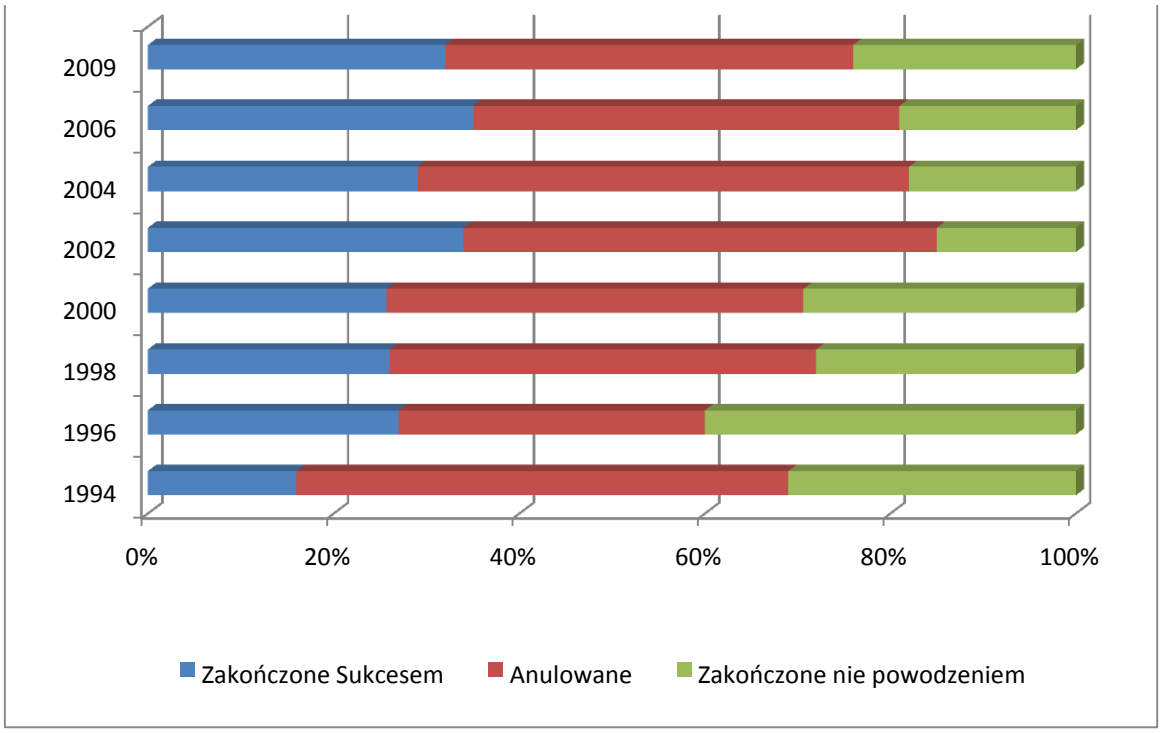
1.2. Przyczyny niepowodzenia projektów

Zarządzanie projektami nie jest zadaniem łatwym, spotykamy się podczas tej funkcji z wieloma przeciwnościami losu, które jak się okazują powodują, że realizacja projektu może zakończyć się niepowodzeniem. Zapewne każdy spotkał się podczas realizacji projektów z problemami przekroczenia czasu czy też budżetu. Odnosząc się do wyżej wymienionej definicji celem projektu jest dostarczenie specyficznego produktu spełniającego odpowiednie kryteria. Nie stosowanie się do ograniczeń, które są przypisane do danego projektu może spowodować niepowodzenie projektu lub też znaczne przekroczenie terminu czy też budżetu. Ograniczenie te można też zobrazować za pomocą trójkąta ograniczeń (Rys. 1.1.), który pokazuje możliwość kontrolowania projektu tak, aby równowaga trójkąta została zachowana. [2]



Rys. 1.1. Trójkąt zakresu projektu [2]

Utrzymanie trójkąta w równowadze jest niezwykle trudnym zadaniem, które wymaga poświęcenia dużej ilości czasu, w przeciwnym razie, gdy jego równowaga zostanie zachwiana powinniśmy zastanowić się nad przyczyną takiego stanu. Na świecie jest kilka organizacji badających uwarunkowania projektów, które zakończyły się sukcesem i niepowiedzeniem. Jedną z nich jest wspomniana wcześniej *The Standlish Group*, która przedstawiła takie uwarunkowania oraz statystyki realizacji projektów (Rys. 1.2) z branży IT w raporcie o tytule *„Chaos Report”* z 1995 roku. [1]



Rys. 1.2. Wyniki powodzenia projektów w latach 1995-2009 prowadzonych przez *The Standish Group*

Z powyższej statystyki wynika, że na przełomie lat 1994-2006 procent projektów zakończonych całkowicie sukcesem wzrósł, co może świadczyć o większym stopniu świadomości swojej roli w projekcie przez kierowników projektu oraz o lepszym wykonywaniu obowiązków. Fakt, że w ciągu tego okresu nieznacznie zmalał procent projektów zakończonych niepowodzeniem jest wynikiem niezadowalającym. Warto się zastanowić nad przyczynami tak dużego niepowodzenia projektów. Wyżej wymieniony raport opisuje najczęstsze przyczyny porażek oraz sukcesów projektów. Wśród sukcesów najistotniejsze są [1]:

- Zaangażowanie klienta – 15.9%
- Wsparcie kierownictwa – 13.9%
- Jasno określone wymagania – 13.0%
- Właściwe planowanie – 9.6%
- Realistyczne oczekiwania – 8.2%
- Mniejsze odstępy pomiędzy kamieniami milowymi – 7.7%
- Kompetencje pracowników – 7.2%
- Odpowiedzialność – 5.3%

- Jasno postawione cele i wymagania – 2.9%
- Ciężko pracujący, skupieni pracownicy – 2.4%
- Pozostałe – 13,9%

Najczęściej występujące są przyczyny związane z odpowiednim zarządzaniem (ptk. 1-6) natomiast pozostałe związane są z kompetencjami czy zaangażowaniem pracowników. *The Standish Group* wymienia także przyczyny niepowodzeń projektów lub niepełnych powodzeń [1]:

- Brak informacji wejściowych od klienta – 12,8 %
- Niekompletne wymagania i specyfikacje projektu – 12,3
- Zmiana wymagań i specyfikacji projektu –11,8
- Brak wsparcia ze strony kierownictwa –7,5
- Brak kompetencji w danej dziedzinie – 7,0
- Brak zasobów ludzkich – 6,4
- Nierealne oczekiwania klienta – 5,9
- Niejasne cele – 5,3
- Nierealne ramy czasowe projektu – 4,3
- Nowe technologie –3,7
- Pozostałe – 23%

Badania przeprowadzone przez *The Standish Group* opierają się na projektach z branży IT aczkolwiek można odnieść wyniki badań do innych branży, w których wykorzystuje się nowe technologie. W innym raporcie *The Slience Fails* sporządzonym przez firmę *VirtualSmarts* oraz *The Concours Group* przedstawionych jest pięć obszarów, które mają znaczny wpływ na powodzenie lub niepowodzenie przedsięwzięć biznesowych. Do obszarów tych zalicza się [3]:

- Wstępne planowanie projektu – zbyt szczegółowe planowanie harmonogramu, z datami i zasobami, które nie przystają do rzeczywistości, „skazując” projekt na niepowodzenie
- Brak zaangażowania sponsora projektu – sponsorzy projektów nie zapewniają odpowiedniego wsparcia grupie realizującej projekt, poświęcają za mało czasu i energii, aby doprowadzić projekt do końca
- Ignorowanie priorytetów – ignorowanie priorytetów zadań w projektach przez członków projektu, do którego zostali przypisani

- Ukrywanie faktycznego stanu projektu – lider projektu i członkowie zespołu nie sygnalizują występujących w projekcie problemów – czekają, aż ktoś inny to powie lub zapyta
- Porażka zespołu – członkowie zespołu nie posiadają odpowiedniej wiedzy, która wymagana jest do realizacji projektu – nie chcą lub nie są w stanie zaangażować się w efektywną realizację projektu

Przeprowadzone badania wykazały, że jeżeli jedna z powyższych sytuacji nie zostanie odpowiednio przeanalizowana i nie zostaną wyciągnięte z niej odpowiednie wnioski w stosunku do prowadzonego projektu, to prawdopodobieństwo zakończenia tego projektu porażką (zdefiniowaną jako przekroczenie zaplanowanego budżetu i czasu oraz niespełnienie wszystkich wymagań klienta co do jakości i funkcjonalności wytworzonego produktu) wzrasta do 85%. Według większości ankietowanych, jeśli zarządzanie krytycznymi obszarami projektu jest realizowane w sposób prawidłowy, to prawdopodobieństwo porażki spada o 50-70% [3].

1.3. Stosowanie metodyk projektowych

Pojawiające się projektach problemy zaczęły być uciążliwe co zmusiło firmy i instytucje do odpowiedniego zarządzania. Zaczęto tworzyć metodyki, które pomagały uporządkować realizację zadań a pierwsze metodyki powstały tuż przed II wojną światową. Dziś na temat zarządzania projektami powstało już wiele publikacji a także wyodrębniły się najbardziej tradycyjne oraz powszechne: PMBoK® oraz PRINCE2®. Tradycyjne podejście do zarządzania projektami opiera się na szczegółowym planowaniu i szacowaniu projektu na podstawie dostarczonych przez klienta wymagań, które jasno określają co dokładnie klient chce, na kiedy projekt ma być gotowy oraz jaki jest budżet przeznaczony na określony projekt a te zasady były podstawą do utworzenia sprawdzonego zestawu narzędzi oraz technik w metodyce PMBOK®. Zupełnie inne podejście jest wykorzystywane w metodyce PRINCE2® gdzie bazuje się na opisie procesów które w ogólnym kontekście pokazują jakie czynności trzeba wykonać aby np. rozpocząć projekt czy przedwcześnie go przerwać. Oprócz tradycyjnego podejścia do projektów zrodziła się też potrzeba znalezienia metodyki która pozwoliła by na szybkie wprowadzanie zmian w

projekcie. Ostatnie lata doprowadziły do określenia zwinnych metod zarządzania projektami (agile project management), które spełniają takie wymagania . Do tego właśnie typu zarządzania projektami zalicza się metodyka SCRUM o której będzie właśnie ta praca. Czym, że jest więc sam SCRUM? Jest to niezbyt skomplikowana metodyka pozwalająca zarządzać projektem przy wykorzystaniu prostego procesu, który jednak nie opisuje dokładnie co powinniśmy w każdej sytuacji wykonać przez co prostota tej metodyki może być złudna a poprawne jej zastosowanie pozwala na dostosowanie projektu w ten sposób aby osiągnąć założone wcześniej cele.

Czym jest Scrum

1.1. Manifest Agile podstawą Scrum

Termin *Scrum* (pl. młyn) zaczerpnięty został ze sportu rugby i jest rodzajem rzutu wolnego w którym dwie splecione drużyny przepychają się ze sobą nad leżącą na ziemi piłką [4].

W zarządzaniu projektami mianem Scrum określamy lekką metodyką zarządzania projektami opartą na Agile skupiającą się głownie na dostarczaniu klientowi funkcjonalności w sposób przyrostowy. Samo pojęcie Agile (*The Agile Manifesto*) wiąże się z deklaracją zasad tworzenia oprogramowania która została sformułowana w lutym 2010 roku [5] przez zwolenników alternatywnych metod tworzenia oprogramowania a wśród nich był uważany za jednego z twórców i propagatora Ken Schwaber. Sam manifest spisany przez uczestników został udostępniony na stronie manifestu [5]:

„Ludzi i ich wzajemne interakcje (współdziałanie) ponad procedury i narzędzia.

Działające oprogramowanie nad wyczerpującą dokumentację.

Współpracę z klientem nad negocjację umów.

Reagowanie na zmiany nad realizowanie planu.”

Oprócz manifestu agile wprowadza także zasady które mają na celu zdefiniowanie filozofii działania. Zasady te tak samo jak manifest zostały przetłumaczone na wiele języków i są dostępne na oficjalnej stronie agile [5]:

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

Bądź otwarty na zmieniające się wymagania nawet na zaawansowanym etapie projektu.

Zwinne procesy wykorzystują zmiany dla uzyskania przewagi konkurencyjnej Klienta.

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

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

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

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

Podstawową i najważniejszą miarą postępu jest działające oprogramowanie.

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

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

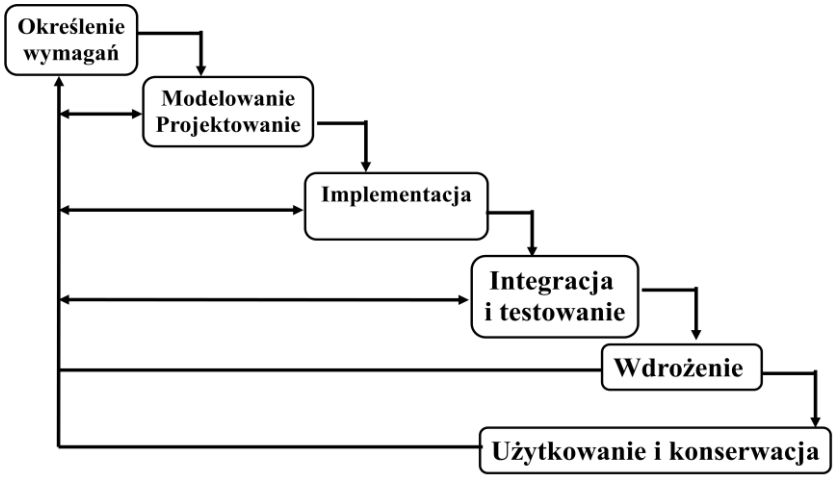
Prostota – sztuka maksymalizacji pracy niewykonanej – jest zasadnicza.

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

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

Twórcy definiując zasady skupili się na wypunktowaniu w jaki sposób kierować projektem, lub wytwarzaniem oprogramowania, aby dostarczyć oprogramowanie dopasowane do rzeczywistych potrzeb klienta [5]. W standardowych metodach kierowania projektami informatycznymi wprowadzenie zmiany przez klienta łączy się ze zmianą

planowania projektu ze względu na kaskadowy model prowadzenia projektu jak to jest na Rys. 0.1.



Rys. 0.1. Proces kaskadowy [6]

Taki przypadek wymaga dodatkowych nakładów ze strony zarządzającej i uwzględnienia ryzyka zmiany wymagań przez klienta. Metodyki oparte na agile skupiają się na dostarczeniu oprogramowania, które klient rzeczywiście potrzebuje a proces tworzenia oprogramowania podąża za zmianami potrzeb klienta. W przypadku zmian w późnej fazie projektu zmiana wymagań jest odpowiednio wpisana w założenia metodyki i nie należy się jej z tego względu obawiać. Takie działanie jest możliwe dzięki częstemu dostarczaniu fragmentów produktu do klienta, w częstości od kilku tygodni do kilku miesięcy, co umożliwia klientowi udzielenie uwag lub zmiany jego wymagań. Według manifestu ważnym aspektem jest komunikacja podczas prowadzenia projektu ponieważ dzięki interakcji klienta i zespołu tworzącego jesteśmy w stanie zrozumieć potrzeby odbiorcy [5]. Najefektywniejszą metodą komunikacyjną jest spotkanie twarzą w twarz dzięki, której jesteśmy w stanie wyjaśnić wszystkie wątpliwości lub problemy napotkane podczas realizacji projektu. Taka rozmowa możne prowadzić do wspólnego rozwiązania lub zasugerować nowe możliwości funkcjonalne a w przypadku problemu z określeniem wymagań przez klienta, nakierować go na jasne zdefiniowanie potrzeb. Klient w takim wypadku jest angażowany do podjęcia współpracy której owocem jest system dopasowany do jego potrzeb a dla strony wykonującej zakończenie projektu sukcesem, jednak sama odpowiednia komunikacja nie jest jedynym czynnikiem, który będzie warunkował o

sukcesie lub porażce projektu. W celu zakończenia projektu, oprócz spełnienia satysfakcji klienta z dostarczonej funkcjonalności, potrzebne jest też odpowiednie zaangażowanie zespołu a możemy tego dokonać dobierając grupę zmotywowanych osób oraz dostarczając im wszystkie potrzebne czynniki środowiskowe. Wartość dodatnia jaką otrzymamy z takiego zespołu to pewność, że zadania, które zostaną powierzone będą wykonane na czas lub w przypadku zdarzeń losowych z niewielkim opóźnieniem. Takie podejście pozwala na zachowanie zespołowi swojego stałego tempa pracy nad oprogramowaniem a spotkania z klientem na zachowanie ciągłej komunikacji w celu zrozumienia wymagań. Monitorowanie postępu prac powierzonych zespołowi powinno się odbywać poprzez ciągłe dostarczenie działających części zamówionego produktu czyli takich, które zostały odpowiednio przetestowane oraz spełniają ustalone wcześniej wymagania. Samoorganizacja zespołu ma wpływ na wiele czynników w wśród nich na architekturę, jakość czy projekt systemu. Zasady agile zobowiązują zespół do podejmowania wspólnie jak najlepszych decyzji dotyczących produktu oraz do jak najlepszego wykonania technicznego co później zmniejsza nieprawidłowości w działaniu systemu. Łatwość reagowania na zmiany jest głównym atutem agile jednak aby reakcja rzeczywiście była łatwa ilość wykonywanych zadań musi być ciągle optymalizowana tak aby zespół mógł określać ilość pracy potrzebnej na wykonanie danej funkcjonalności. Zespół powinien możliwość usprawnienia swojego działania lub ilości wykonywanych przez siebie zadań. Efektem prac całego zespołu jest w pełni zaprojektowane, zgodne z wymaganiami i jak najlepiej wykonane przez zespół oprogramowanie.

Cała filozofia agile sprowadza się do dwunastu zasad które wyżej zostały opisane a które są jej teoretyczną podstawą [7]. W praktyce zostało wypracowanych kilka metodyk zarządzania projektami, które swe podstawy mają w filozofii Agile a wśród nich jest między innymi metodyka Scrum, która jest jedną z popularniejszych metodyk zwinnych i posiada zarówno całe rzesze zwolenników jak i przeciwników, a ponadto w pełni uwzględnia wszystkie dwanaście zasad Agile.

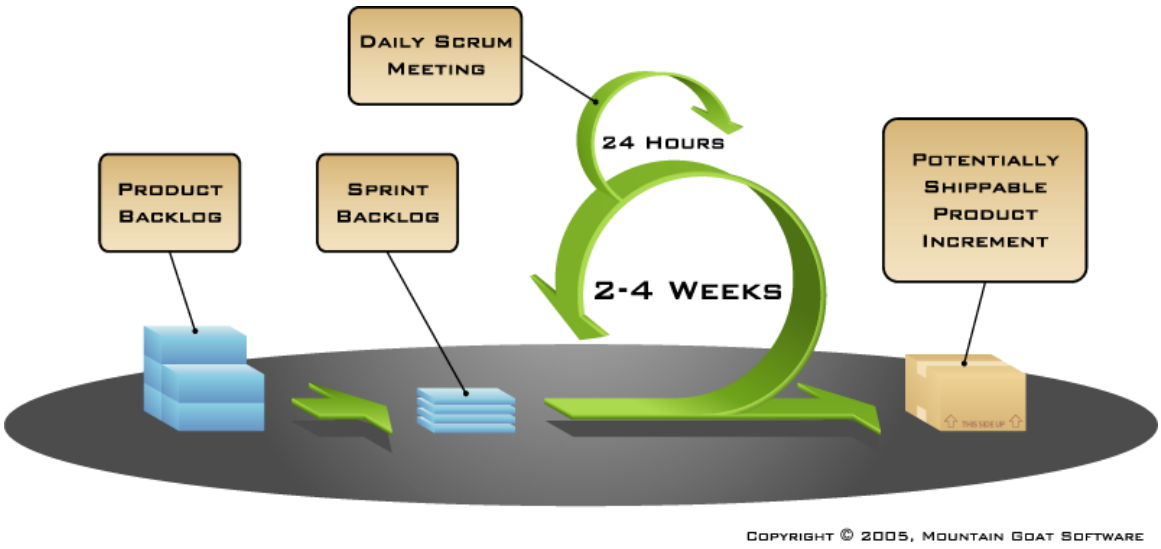
1.2. Scrum jako praktyczne podejście do zasad Agile

Praktycznym podejściem które odwołuje się do zasad agile jest metodyka Scrum. Historia metodyki rozpoczęła się od opublikowania artykułu „*The New New Product Development Game"*, który został napisany przez Hirotaka Takeuchi i Ikujiro Nonaka, w czasopiśmie *Harward Bussines Review* gdzie zostały przedstawione ogólne założenia metodyki [8]. Następnym ważnym wydarzeniem w dziejach metodyki było opublikowanie przez Kena Schwabera i Jeffa Sutherlanda artykułu „Agile Development: Lessons Learned from the First Scrum" w, którym został zawarty pierwszy formalny opis metodyki. Jak zostało to opisane na początku tego rozdziału Scrum wolnym tłumaczeniu oznacza młyn a wywodzi się ze sportu Rugby w którym oznacza rodzaj rzutu wolnego, gdzie dwie splecione drużyny przepychają się nad leżącą na ziemi piłką. Układ ten wymaga od zawodników szybkiej reakcji na poczynania przeciwnika aby zdobyć piłkę. W Agile oznacza praktyczną implementacją jej zasad. Podstawą Scrum jest zdefiniowany proces, który jest wykonywany iteracyjnie, aż do osiągnięcia celu przy zachowaniu odpowiedniej jakości co możemy to porównać do przyrostowego modelu tworzenia oprogramowania. Ken Schwaber w swojej książce „Sprawne zarządzanie projektami metodą Scrum" porównuje proces Scrum do empirycznego procesu w którym istnieją trzy filary [7]:

- I Filar – Widoczność
- II Filar – Inspekcja
- III Filar – Adaptacja

W empirycznym procesie kontroli jakości widoczność oznacza że rezultaty projektu wynikające z zastosowania procesu są widoczne dla zarządzających a także określenie co jest uwzględnione podczas obserwacji. Dla wykonujących zadania oznaczenie go jako wykonane wiąże się z zastosowaniem do ogólnie przyjętej definicji słowa „wykonane". Drugi filar, inspekcja, w ujęciu procesu scrumowego oznacza odpowiednią kontrole podczas procesu aby wykryć nieoczekiwane odchylenia. Proces empiryczny pozostawia wolną rękę co do częstości przeprowadzanych inspekcji. Wykrycie niepożądanych odchyleń pozwala na reakcję ze strony zarządzających, które wiąże się z III filarem, Adaptacją, której zastosowanie wiąże się z podjęciem działań które mają zapobiec dalszemu pogłębianiu się wykrytego problemu przez, które było by nieakceptowane [9].

Scrum jako praktyczne podejście do zasad Agile opiera się na iteracyjnym (przyrostowym) procesie kontroli projektu jak to zostało pokazane na Rys. 0.2.
Pierwszym elementem który podawany jest na wejściu procesu jest nazywany zaległościami produktowymi (*Product Backlog*) z, których następuje wybranie poszczególnych funkcjonalności, które zostaną zrealizowane w danej iteracji (*Sprint Backlog*). Następnym elementem procesu jest realizacja iteracji (zwanej też sprintem i reprezentującej na Rys. 0.2 większy okrąg) oraz codziennej inspekcji (mniejszy okrąg nad symbolem iteracji opisanej jako *Daily Scrum meeting*) w, której członkowie spotykają się by przeprowadzić inspekcję innych członków zespołu. Efektem całego procesu jest potencjalny przyrost funkcjonalności [7].



Rys. 0.2. Proces Scrum [10]

Adaptacyjna natura tego procesu pozwala zespołowi na zmianę podejścia sposobu pracy z dnia na dzień. Także wybór wymagań jest kwestią, której podejmuje się zespół rozważając wspólnie możliwe rozwiązania problemu z, których wybierane jest najlepsze podejście. Rozważanie nad metodą rozwiązania problemu przez zespół jest praktyczne z punktu podwyższania ich kwalifikacji, rozwijając umiejętności. W Scrumie wyróżniamy tylko trzy role: zespół, ScrumMaster, właściciel produktu a których opis zostanie podany w następnym podrozdziale [9]. Iteracyjne natura projektu pozwala na rozpoczęcie projektu bez jasno określonej wizji. Z czasem trwania projektu właściciel produktu odpowiadający

przed sponsorami projektu dostarcza wizję produktu którego celem będzie maksymalizacja współczynnika ROI (*Return of Investment* – Zwrot z inwestycji) [7]. Wizja ta do procesu Scrumowego dostarczana jest za pomocą zaległości produktowych które stanowią listę wymagań z wyznaczonymi priorytetami. Lista takich wymagań służy zespołowi do wyboru funkcjonalności którą zobowiązuje się na wykonanie w najbliższym sprincie, który trwa zazwyczaj od 10 do 30 dni aczkolwiek nie ma ograniczeń co do stosowania innej długości jednak ta jest najbardziej zalecana [7].

Każdy początek iteracji rozpoczyna się od spotkania podczas, którego zespół oraz właściciel produktu ustala co zostanie wykonane podczas kolejnego sprintu. Podczas spotkania zespół dokonując wyboru funkcjonalności kieruje się sugestiami właściciela produktu odnośnie tego co należy zrobić w pierwszej kolejności a właścicielowi produktu zespół informuje co z pożądanej funkcjonalności jest wstanie wykonać. Spotkanie takie nazywane jest spotkaniem planującym sprint i nie może trwać dłużej niż osiem godzin a ponadto dzieli się na dwie części z których każda nie może trwać dłużej niż cztery godziny [7]. Pierwsza część składa się z prezentacji przez właściciela produktu zaległości produktowych wraz z priorytetami a zespół w tym czasie jest odpowiedzialny za zdobycie jak największej ilości informacji od Właściciela produktu odnoście prezentowanej wizji. Podczas tego spotkania zespół także wybiera część funkcjonalności, która zostanie zrealizowana w następnym sprincie. Jeżeli przed upływam pierwszych czterech godzin zakończy się spotkanie zespól rozpoczyna drugą część na, której dokładnie omawia i planuje swoja pracę i umieszcza zadania na zaległościach sprintu.

Od początku drugiego spotkania rozpoczyna się czas trwania sprintu o określonej wcześniej długości [7].

Podczas trwania iteracji zespół zbiera się każdego dnia w ustalonym wcześniej miejscu o ustalonej godzinie na krótkie 15 minutowe spotkanie nazywane codzienną inspekcją lub codziennym spotkaniem Scrumowym [9] na, którym spotykają się ScrumMaster oraz zespół. Wszyscy uczestnicy tego spotkania zobowiązani są do stawienia się na spotkanie na czas a każde spóźnienie powinno być karane ogólnie przyjętą i zaakceptowaną przez zespół karą. Przykładem takiej kary może być np. zapłacenie jednego dolara [7] lub co można spotkać w korporacjach chodzenie przez cały dzień z tabliczką z napisem *„I was late”* jak na Rys. 0.3. Samo spotkanie jest organizowane w celu

zsynchronizowania pracy zespołu a sam zespół jest zobowiązany do udzielenia odpowiedzi na trzy podstawowe pytania [7]:

- Co robiłeś/łaś w projekcie od ostatniego spotkania?
- Co zamierzasz zrobić pomiędzy obecnym spotkaniem a następnym?
- Czy są jakieś przeszkody, które przeszkadzają w realizacji założeń tego sprintu?

Rys. 0.3. Spotkanie Scrumowe z osobą ukaraną

Pod koniec sprintu przeprowadzane jest spotkanie nazywane przeglądem sprintu na, którym zespół prezentuje pracę wykonaną w danym sprincie. Spotkanie to mające charakter nieformalny, jest ograniczone do 4 godzin w, podczas którego uczestniczy właściciel produktu oraz opcjonalnie inni udziałowcy. Celem tego spotkania jest prezentacja zrealizowanej funkcjonalności oraz pomoc właścicielowi produktu w określeniu kierunku w, którym zespół powinien rozwijać produkt [7]. Po zakończeniu spotkania przeglądowego ScrumMaster powinien przeprowadzić z zespołem retrospektywne spotkanie, które dotyczy sprintów. Spotkanie to ma na celu zachęcenie zespołu do zmiany parametrów procesu tak by zwiększyć efektywność, jak na przykład czas trwania sprintu, oraz zorganizować pracę tak by stała się ona bardziej wydajna [7].

Wszystkie trzy spotkania planujące sprint, codzienna inspekcja, podsumowanie wraz z retrospekcją odzwierciedlają założenia empirycznej kontroli procesu oraz trzech filarów na, których Scrum się opiera jednocześnie zachowując zasady związane z manifestem Agile. Oprócz trzech spotkań w Scrumie możemy wyróżnić także trzy role oraz trzy artefakty. Opisując metodykę w następnej kolejności skupie się na opisie roli w projekcie a następnie na temat artefaktów Scrumowych.

1.3. Role w projekcie

W Scrumie należy wyróżnić trzy role między które jest podzielona odpowiedzialność w projekcie. Są to [9]:

- Zespół
- ScrumMaster
- Właściciel Produktu

Każda z wyżej wymienionych ról charakteryzuje się odpowiednim zakresem odpowiedzialności. Osoba odpowiedzialna za budowanie fundamentu i obsługę projektu po stronie klienta to właściciel produktu. Jego główną rolą jest komunikowanie się z osobami zainteresowanymi projektem lub produkowanym systemem.

Poprzez komunikację z przyszłymi zainteresowanymi właściciel produktu buduje listę ogólnie sformułowanych wymagań oraz jest odpowiedzialny za budowę celów projektu. Wymagania zbierane przez właściciela produktu są umieszczane w zaległościach produktowych (*Product Backlog*), które powinny zostać sformułowane w ogólny sposób a każde z wymagań powinno mieć przypisany priorytet. Właściciel produktu powinien dbać o to aby na każdy następny sprint uaktualnić listę zaległości produktowych [9].

Zespół w projekcie odpowiada za przekształcenie zaległości sprintu, która została wybrana z zaległości produktowych, w przyrost funkcjonalności. Zespół decyduje o tym w jaki sposób organizować to znaczy, że zespół jest samo organizacyjny i jest odpowiedzialny za budowę przyrostu funkcjonalności. Każdy członek zespołu pracuje nad własną częścią funkcjonalności ale za sukces każdego sprintu i projektu jest odpowiedzialny cały zespół [9].

Osoba która, jest w roli ScrumMastera odpowiedzialna jest za utrzymanie ciągłości procesu scrumowego. ScrumMaster pełni także rolę nauczyciela ponieważ na nim

spoczywa obowiązek nauczenia metodyki Scrum. Oprócz wyżej wymienionych funkcji ScrumMaster odpowiedzialny jest za wdrożenie procesu Scrumowego tak aby był on zgodny z kulturą organizacji a także za usprawnianie samego procesu [9].

Każda osoba zaangażowana w projekt musi mieć przypisaną jedną z wyżej wymienionych ról. Było by idealne gdyby osoby nie będące zaangażowane w projekt nie były by nim zainteresowane. W Scrumie wyróżnia się grupy zaangażowane w projekt, czyli te które mają moc władczą w projekcie oraz osoby które takiej mocy nie posiadają i nie mogą bez wyraźnej potrzeby się wtrącać. Grupy te nazywamy odpowiednio „świniami" (osoby realizujące projekt) oraz „kurczakami" (osoby zainteresowane projektem) [7] a nazwy tych grup wzięły się z pewnej historyjki, która został przedstawiony w formie komiksu na Rys. 0.4.

Rys. 0.4. Komiks opisujący genezę nazewnictwa grup „świnie" i „kurczaki" [11]

Podział na grupy i role w projekcie pozwala w Scrumie na wyznaczenie odpowiedzialności oraz wyznacza obszar decyzyjności poszczególnych ról [7]. Dzięki temu zabiegowi wiemy kto jest odpowiedzialny za przygotowanie zaległości produktowych a kto za wykonanie prac. W następnych podrozdziałach dokładniej opiszę poszczególne role w projekcie oraz ich zakres odpowiedzialności zaczynając od opisu zespołu

1.3.1. Zespół

Zadaniem zespołu w metodyce Scrum jest przekształcanie zadań, które zostały wybrane spośród wymagań zgromadzonych w zaległościach produktowych. Lista takich

wymagań realizowanych podczas konkretnego sprintu nazywa się zaległościami Sprintu (*Sprint Backlogu*). Wymagania umieszczone na tej liście mają taki sam priorytet jak te, które zostały przygotowane przez właściciela produktu. Zespół jest zobowiązany do wykonywania w pierwszej kolejności zadań z wyższym priorytetem co zwiększa wydajność metodyki Scrum z tego względu, że klient w pierwszej kolejności otrzymuje przyrost funkcjonalności, który jest mu najbardziej potrzebny. Scrum zakłada że Zespół jest samoorganizującą się jednostką, która optymalizuje swoją pracę a cały plan realizacji założeń sprintu jest wykonywany przez zespół [7]. Zespół może otrzymywać rady dotyczące samoorganizacji od ScrumMastera oraz „kurczaków" jednak to czy sam zastosuje się do tych rad leży tylko w gestii zespołu.

Czas realizacji wszystkich zadań przez zespół nie więcej niż długość sprintu a jest liczony zaraz po ich wybraniu do realizacji przez zespół zadań [7]. Po starcie odliczania do końca sprintu zespół jest zobowiązany do zrobienia wszystkiego aby spełnić swoje zobowiązania a na końcu prezentuje swoje zobowiązania w postaci przyrostu produktu przed właścicielem produktu oraz udziałowcami [7]. W praktyce dzisiejszych projektów mamy zespoły które, są zarządzanie przez osoby o odpowiednich kompetencjach podczas gdy w Scrumie mamy zespoły samo zarządzające. Jak w takim wypadku wprowadzić zespół, który był zarządzany w samoorganizację? Metodyka zakłada, że rolę wdrożenia do samoorganizowania sobie pracy zespołu powinna przejąć osoba w roli ScrumMastera która uczy zespół zasad empirycznego procesu przy pomocy codziennych spotkań i spotkaniu retrospektywnym (inspekcja i adaptacja), poprzez pomoc w rozwiązywaniu problemów, a także przejrzystością w celu zachowania odpowiedniej jakości [12].

Budowa zespołu odbywa się przed rozpoczęciem pracy nad projektem dlatego też ważne jest aby dobierać już na samym początku kompetentne osoby, które mają stworzyć zespół. Zmiany w zespole podczas trwania projektu mogą powodować niepożądane opóźnienia wynikające z komunikacji w zespole ponieważ każda grupa przechodzi w swoim rozwoju przez kolejne fazy [13]:

- Formowania (*forming*)
- Konfliktów (*stortming*)
- Normalizacji (*norming*)
- Współdziałania (*performing*)

Jakiekolwiek zmiany w zespole i przechodzenie przez zespół od nowa kolejnych faz może spowodować opóźnienia w wykonaniu zaplanowanych prac dlatego też ważne jest zbudowanie zespołu składającego się z odpowiednich osób już na samym początku projektu.

1.3.2. ScrumMaster

W metodyce Scrum role kierownika projektu przejmuje osoba w roli ScrumMastera. Genezę tej nazwy Ken Schwaber opisuje w swojej książce „Sprawne zarządzanie projektami metodyką Scrum" wyjaśniając że nazwa ScrumMaster miała zaznaczać zupełnie inny zasięg roli oraz drastyczne zmiany w prowadzeniu projektu.

Osoba pełniąca tą rolę ma władzę tylko wynikającą z zasad które istnieją w metodyce Scrum i pilnuje aby wszyscy tych zasad też przestrzegali [7]. Jest to osoba która musi mieć umiejętność rozumienia tych zasad i w jasny sposób nauczania innych tak aby wszyscy zrozumieli zasady kierujące Scrumem [9]. ScrumMaster jest też odpowiedzialny za pomoc właścicielowi produktu w określaniu zaległości produktowych oraz priorytetów a zespołowi za pomoc w przemienieniu zaległości produktowych w przyrost funkcjonalności.

1.3.3. Właściciel produktu

Osoba pełniąca rolę właściciela produktu jest odpowiedzialna z zbieranie wymagań dotyczących tworzonego systemu. Praca tej osoby jest związana z komunikacją z wszystkimi osobami zainteresowanymi projektem i spisywaniu wymagań. Następnie lista zebranych wymagań jest analizowana przez właściciela produktu i przedstawiana jako lista zaległości produktowych opisująca wymagania systemu wraz z przypisanymi priorytetami. Obecnie stosowane są wyrafinowane techniki analizy wymagań dzięki, którym możemy tworzyć szczegółowo opisane i przeanalizowane z dużą ilością diagramów dokumenty gdzie często przekraczają ponad 500 stron. Takie dokumenty uniemożliwiają swobodną pracę z wymaganiami z tego względu, że czytanie takiego dokumentu zajmuje dużo czasu a zapamiętanie każdego szczegółu jest prawie niemożliwe. Dzięki roli właściciela produktu Scrum pozwala na zupełne inne podejście, zamiast skupiać się na tworzeniu ogromnej ilości

dokumentacji skupia się na komunikacji ze stronami zainteresowanymi [9]. Kto może być właścicielem produktu? Założenia Scrum, opisane w „*Scrum Giude*", mówią nam że właścicielem produktu musi być koniecznie jedna osoba w związku z czym nie może to być zespół czy specjalnie powołany komitet. Dopuszczenie do roli właściciela produktu grupy lub specjalnego komitetu mogło by spowodować niepotrzebny bałagan oraz spadek odpowiedzialności roli. Dobrze żeby właścicielem produktu była osoba, która ma być menedżerem tworzonego produktu lub tez opiekunem biznesowym informatyzowanej funkcji [9].

W związku z tym, że właściciel produktu jako jedyny posiada moc decyzyjną co do ustalania priorytetów tworzonej funkcjonalności to pozostałe osoby w organizacji muszą szanować jego decyzje. Żadne inne osoby nie mogą nakazywać zespołowi zmiany priorytetów natomiast zespół nie może słuchać osób innych niż właściciel produktu co do priorytetów wykonywanej funkcjonalności [7]. Wszystkie decyzje właściciela produktu widoczne są w każdych zaległościach sprintu (przyznawanie pierwszeństwa zadań), które są wybierane przez zespół co powoduje, że spoczywa na nim duża odpowiedzialność za decyzje dotyczące wyboru funkcjonalności [7].

1.4. Artefakty w Scrumie

Scrum podobnie jak inne metodyki dostarcza nam artefakty dzięki którym jesteśmy w stanie monitorować wykonywaną pracę. W wielu metodykach wykorzystuje się część artefaktów, która pozwala na tworzenie ogromnej ilości dokumentacji. W Scrum wyróżniamy trzy podstawowe dokumenty. Pozwalają one na monitorowanie wielu parametrów podczas trwania projektu a w szczególności postępów prac projektu.
Są to [9]:

- Zaległości produktowe
- Zaległości sprintu
- Wykres wypalenia

Zaległości produktowe są artefaktem pozwalającym na zarządzanie wymaganiami produktu. Są one tworzone przez właściciela produktu i składają i się z listy wymagań z odpowiednio przypisanymi priorytetami. Specyfika Scruma pozwala na szybkie reagowanie na zmiany jakich klient oczekuje i w związku z tym na starcie projektu

wymagania nie muszą być konkretnie sprecyzowane i mogą zostać rozbudowane o kolejne w późniejszej fazie iteracji. Dzięki temu nie musimy poświęcać na odpowiednie zarządzanie zmianą podczas trwania projektu a korzyści jakie płyną dla klienta to otrzymanie produktu, który będzie zgodny z rzeczywistymi wymaganiami [7].

Lista zaległości produktowych może zawierać wiele wymagań różnego rodzaju to znaczy, że w jej spisie mogą wystąpić wymagania funkcjonale, technologiczne, wydajnościowe lub inne. Każdy element umieszczony na liście powinien zawierać także odpowiednie atrybuty takie jak opis, priorytet, koszty a lista powinna byś posortowana według priorytetów. Takie sortowanie pozwala na uświadomienie właścicielowi produktu, które tak naprawdę wymagania są ważniejsze od innych i jest to przedstawione w jasny sposób aczkolwiek nie jest wymagane sortownie według priorytetów jak pokazano to na Rys. 0.5. [7]. Często spotykaną praktyką w opisie wymagań w zaległościach produktowych są tak zwane historie użytkownika (*user stories*), których konstrukcja składa się z opisu danego wymagania poprzez wyznaczenie aktora oraz celu, który chce osiągnąć po to by spełnić pewien zysk. Schemat konstruowania *User Stories* opiera się na szablonie: Jako (kto?, rola), chcę (czego oczekuję?), by (po co?).

Przykładem takiej historyjki może być: Jako zwykły użytkownik serwisu chcę powiększyć zdjęcie, by lepiej mu się przyjrzeć. Zespół analizując taką historię użytkownika powinien mieć wraz z historią dostarczone informacje na temat kryteriów akceptacji zadania. Przy okazji posiadania takich szczegółowych informacji zespół może wykonać zadanie w taki sposób w jaki oczekuje tego klient [14]. Przykładem takich kryteriów dla wyżej wymienionej historii użytkownika mogą być:

- Użytkownik powinien powiększyć i pomniejszyć zdjęcie korzystając z przycisku.
- Użytkownik powinien widzieć aktualną wartość powiększenia obrazu
- Użytkownik powinien mieć możliwość płynnego wyboru powiększenia obrazu za pomocą suwaka
- Powiększenie może być realizowane w zakresie od -100% do 200%

Backlog Description Title Import
Project selection or new
Template backlog for new projects
Create product backlog worksheet with formatting
Create sprint backlog worksheet with formatting
Display tree view of product backlog, releases, sprints
Sprint-1
Create a new window containing product backlog template
Create a new window containing sprint backlog template
Burndown window of product backlog
Burndown window of sprint backlog
Display tree view of product backlog, releases, prints
Display burndown for selected sprint or release
Sprint-2
Automatic recalculating of values and totals
As changes are made to backlog in secondary window, update burndown graph on main page
Hide/automatic redisplay of burndown window
Insert Sprint capability ... adds summing Sprint row
Insert Release capability ... adds summary row for backlog in Sprint
Owner/assigned capability and columns optional
Print burndown graphs
Sprint-3
Duplicate incomplete backlog without affecting totals
Note capability
What-if release capability on burndown graph
Trend capability on burndown server
Publish facility for entire project, publishing it as HTML web pages
Future Sprints
Release-1

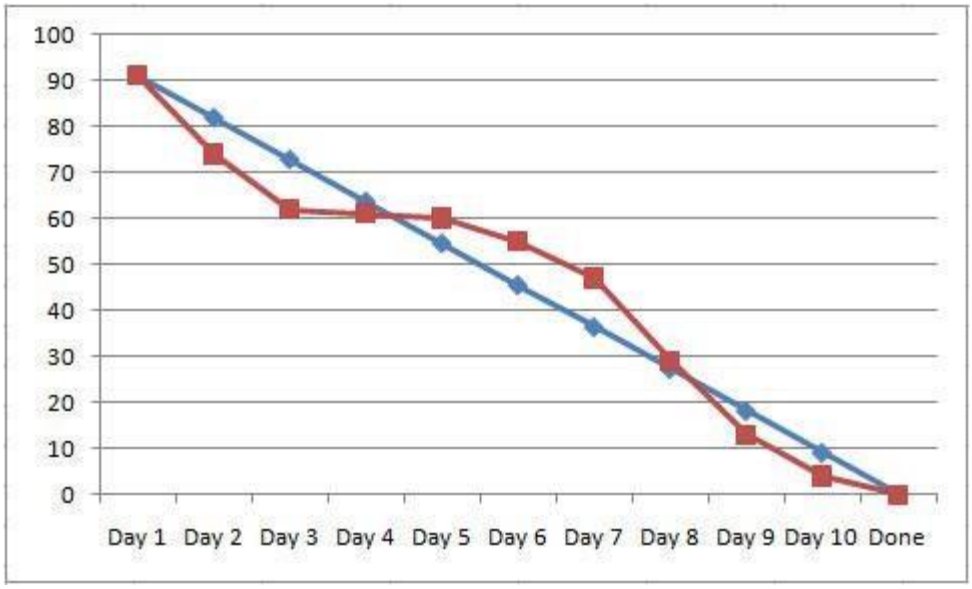
Rys. 0.5. Przykładowy *Product Backlog* [7]

Szacowaniem wielkości wymagań umieszczonych w zaległościach produktowych zajmuje się zespół. Jeżeli wymagania opisane są za pomocą specyficznych dla metodyk zwinnych Historii użytkownika to zespół powinien skorzystać przy estymacji z pomocy *Story Points* które określają wielkość zadania a nie jednostkę czasową [15]. Szacowanie za pomocą tej metodyki opiera się często na relatywności zadań a możemy takie szacowania wykonać wybierając najłatwiejsze zadanie i dając mu liczbę punktów równą 2 a pozostałe szacunki tworzy się w zależności od wielkości wyszacowanego zadania. Najczęściej używana skalą stosowaną w tym celu jest: ½, 1, 2, 3, 5, 8, 13, 20, 40, 100 punktów [15].

Podobną funkcję do zaległości produktowych spełniają zaległości sprintu. Posiadają one podobną konstrukcję z tą różnicą że wybór wymagań zawarty na niej będzie aktualny tylko na czas trwania sprintu i dotyczą prac które ma wykonać zespół aby spełnić założenia sprintu [9]. Większość z zadań znajdujących się na tej liście wybieranych jest podczas

spotkania planistycznego *Sprintu*. Zadania przedstawione na tej liście muszą być zdekomponowane ponieważ ułatwia to komunikację zespołu podczas codziennych spotkań Scrumowych. Lista elementów znajdujących się na liście zaległości produktowych jest modyfikowana przez Zespół podczas trwania sprintu. Modyfikacja listy zaległości sprintu jest konieczna ponieważ w ramach jednego wymagania umieszczonego na liście zaległości produktowych może wyniknąć więcej zadań niż planował zespół [7]. Podczas trwania sprintu ważne jest to że lista realizowanych w danym sprincie wymagań nie może ulec zmianie. Takie ograniczenie wynika z faktu iż istnieją do tego specjalne spotkania planistyczne sprintu na, których zespół ustala z właścicielem produktu zakres prac jakie należy wykonać a samą listę zaległości sprintu może modyfikować tylko zespół [7].

Ostatnim ważnym artefaktem dostarczanym przez Scrum jest wykres wypalenia. Wykres ten pokazuje nam ilość pracy pozostałej do końca sprintu (lub zakończenia produktu) [9]. Wykres ten tworzony jest przez sumę estymacji każdego z zadań w danym sprincie (całości produktu) i uśredniony spadek pracy pozostałej do wykonania dla każdego dnia sprintu (wydania produktu) [7] jak zostało to pokazane na Rys. 0.6. Kiedy określimy sumę estymacji czasów realizacji zadań dla pierwszego dnia sprintu oraz przyjmiemy zero dla ostatniego dnia sprintu otrzymamy linię, która będzie pokazywać średnią ilość pracy, która powinna zostać wykonana do końca sprintu (zaznaczone na Rys. 0.6. na niebiesko) podczas kolejnych dni. Można przyjąć że taka sytuacja jest rozpatrywana jako idealna natomiast rzeczywista pozostała do wykonania praca zaznaczona na wykresie na Rys. 0.6. jest zaznaczona kolorem czerwonym. Jeżeli linia ta znajduje pod linią niebieską to znaczy że prace zespołu idą szybciej niż zostało to wyszacowane w przeciwnym wypadku jeżeli rzeczywista pozostała praca jest ponad pracą średnią to znaczy że zespół nie wyrabia się z realizacją założonych zadań. W obu przypadkach Scrum zakłada odpowiednią reakcję na te przypadki. Jeżeli zespół wykona wszystkie prace przed zakończeniem sprintu można się skontaktować z właścicielem produktu aby dorzucić wymaganie do realizacji lub zakończyć sprint wcześniej. W drugim przypadku kiedy zespół wykonywanej pracy jest więcej niż zespół założył to w tym przypadku także należy się skontaktować z właścicielem produktu i albo zmniejszyć zakres wydania albo przedwcześnie zakończyć sprint i ponownie wybrać zaległości produktowe do danego wydania [7].



Rys. 0.6. Przykładowy Wykres Wypalenia

Wszystkie wyżej wymienione narzędzia pozwalają w Scrumie kontrolować proces tworzenia produktu. Każde z tych narzędzi pozwala na kontrolę kilku krytycznych obszarów projektu takie jak zakres produktu czy zarządzanie zmianą. Artefakty te są oczywiście podstawowymi w tej metodyce jednak nic nie stoi na przeszkodzie aby budować własne oparte na zasadach Scrum. Przykładem może być tutaj historia przytoczona w książce Kena Schwabera na temat firmy *MegaEnergy* gdzie wymagania z zaległości produktowych były raportowane przy pomocy diagramów Gantta [7].

Praktyczne podejście do Scruma

Praktyka pokazuje że ustandaryzowane podejście do różnych problemów nie zawsze może się zakończyć takim samym wynikiem dla podobnych uwarunkowań. Wykonując dwa projekty o podobnym zakresie nie nigdzie zapisane że muszą się oba zakończyć powodzeniem. Taki stan może być spowodowany uwarunkowaniami zewnętrznymi lub też zupełnie innym podejściem do projektu oraz doborem zupełnie innego zespołu. Przypadek w którym do realizacji projektu wybieramy zespół który się składa z dużej ilości ludzi scrum może spowodować trudności w komunikacji a w takich wypadkach należy wykazać się aby zorganizować pracę tak aby było to zgodne z

podejściem metodyki oraz żeby komunikacja została na takim samym poziomie a cele projektu zostały wykonane. Sposobów praktycznego podejścia do Scruma może być wiele w zależności od uwarunkowań a najważniejsze jest to żeby ułatwiały prowadzenie projektów. Najczęściej stosowanymi praktykami w metodyce Scrum są:

- Scrum Scrumów
- Scrum w testowaniu oprogramowania
- Połączenie Scrum'a i Kanbana

Scrum Scrumów polega na odpowiednim zarządzaniu dużym zespołem. Ponieważ założenia Scruma mają na celu między innymi polepszenie komunikacji wewnątrz zespołu (optymalnym zespołem jest zespół 7-9 osób) [7] a w przypadku dużych zespołów (powyżej 10 osób) następują trudności komunikacyjne. W tym przypadku można podzielić zespół na kilka mniejszych zespołów który każdy będzie prowadzony w metodyce Scrum oraz utworzyć zespół który będzie zajmował się synchronizacją prac między zespołami.

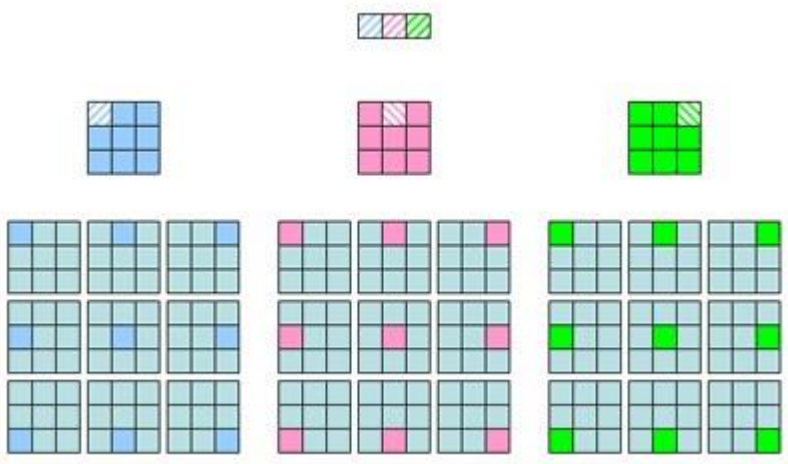
Testowanie oprogramowania jest problematycznym działem tworzenia oprogramowania. W klasycznych kaskadowych metodach tworzenia oprogramowania testowanie jest oddzielną fazą która następuje po fazie implementacji [6]. Scrum zgodnie z I filarem empirycznej kontroli procesów (przejrzystością) wprowadza pojęcie *„Definition of Done".* Pojęcie to ma na celu wyszczególnienie co należy uznać za zrobione a z tym wiąże się testowanie oprogramowania w trakcie wytwarzanie funkcjonalności. W przypadku stosowania historii użytkownika w celu opisu wymagań wymaganie jako zrobione możemy uznać takie które spełnia wszystkie kryteria a przy tym opis wymagań jest na tyle zwięzły że nie trzeba się przebijać przez ogromne dokumenty wymagań [14]. Niestety nie zawsze jesteśmy w stanie wychwycić wszystkie błędy występujące w oprogramowaniu dlatego też można w Scrum stosować praktyki pozwalające na optymalne testy oprogramowania.

Stosowanie metodyki Kanban ułatwia zarządzeni pracą w zespole jednak Scruma zakłada że zespół jest samoorganizującą się jednostką. Osoba pełniąca rolę ScrumMastera może pomóc zespołowi w organizacji jednak nie może narzucać
organizowania się zespołu. Sprawdzonym i często stosowanym przez przedsiębiorstwa sposobem na dobre samo zarządzanie się zespołu jest metodyka Kanban która w prosty sposób pozwala na kontrolę postępu prac [16]. W kolejnych podrozdziałach dokładniej

opiszę każde z praktycznych podejść do Scruma.

1.5. Scrum Scrumów

Scrum sam w sobie jest skuteczny jeżeli w zespole jest niewielka liczba osób. W optymalnym przypadku zespół Scrumowy powinien liczyć 7-9 osób chociażby z powodu codziennych spotkań które nie powinny zajmować dłużej niż 15 minut. W przypadku kiedy mamy zespół liczący 200 osób takie spotkanie zajęłoby znacznie więcej czasu, a nie wykluczone że nawet cały dzień. Spotkania codzienne mają za zadanie synchronizację pracy członków zespołu a spotkanie wykorzystujące tak dużą liczbę osób może spowodować niemałe zamieszanie. Jak w takim przypadku można rozwiązać sprawę swobodnego prowadzenia projektu w metodyce Scrum? Ken Schwaber w swojej książce opisuje metodę w której jeden duży zespół Scrumowy został podzielony na kilka mniejszych zespołów, w których zarządzanie też odbywało się za pomocą Scruma, i jeden dodatkowy proces Scrumowy który miał za zadanie synchronizować pracę zespołów [7]. Jakie korzyści może przynieść taka praktyka? Przede wszystkim dzięki temu Scrum jest prostym do skalowania procesem a synchronizacja prac zespołów może się odbywać na wielu warstwach co znaczy że nie może być wiele hierarchii budowania zespołów Scrumowych synchronizujących pracę z innymi zespołami.
Przypadek taki został przedstawiony na Rys. 0.1.



Rys. 0.1. Schemat zespołów w projekcie w którym się używa się Scruma Scrumów [10]

W takim modelu codzienny Scrum jest nazywany często spotkaniem synchronizującym w którym należy zsynchronizować pracę między wszystkimi zespołami. Na spotkania takie zazwyczaj uczęszcza jedna osoba a najlepiej żeby była to osoba z zespołu ponieważ to on ma synchronizować pracę z innymi zespołami. Rzadko spotykanym rozwiązaniem jest wysyłanie na spotkania synchronizujące właściciela produktu ponieważ właściciel produktu nie jest w stanie określić etapu prac w czasie trwania sprintu ani nie jest w stanie określić czym zespół będzie się zajmował oraz jakie ma problemu [17]. Osoba wybrana przez zespół powinna posiadać odpowiednie kompetencje oraz która zna wszystkie problemy zespołu podczas trwania projektu.

Częstotliwość prowadzenia spotkań synchronizujących jest zależna od zespołów jednak Ken Schwaber sugeruje aby takie spotkania występowały codziennie [7] natomiast Mike Cohn [17] sugeruje aby takie spotkania wykonywać dwa lub trzy razy w tygodniu. Czas spotkania takiego może trwać dłużej niż zwykłego spotkania Scrumowe a według Cohn'a spotkanie synchronizujące powinno trwać od 30 do 60 minut i najczęściej powinno się ono odbywać po spotkaniu codziennym [17]. Wyjątkiem może być kiedy spotkanie codzienne jest zawsze pod koniec dnia pracy. Przebieg spotkania synchronizującego jest podobny do codziennego Scruma należy mieć na uwadze to że nie jest to takie same spotkanie. Na takim spotkaniu każdy oddelegowany członek zespołu musi odpowiedzieć na pytania w trochę zmienionej formie:

- Co twój zespół wykonał od ostatniego spotkania synchronizującego?
- Co twój zespół zamierza zrobić przed następnym spotkaniem?
- Czy są jakieś problemy które spowalniają twój zespół?

Mike Cohn sugeruje dodanie jeszcze jednego pytania mającego na celu usprawnienie procesu synchronizacji [17]:

- Czy masz zamiar umieścić coś dodatkowego w kierunku działań innego zespołu?

Wartość tego ostatniego proponowanego przez Cohna pytania może być bardzo ważna kiedy koordynowana jest praca kilku zespołów. W przypadku kiedy któryś zespół ma problem z wykonywaną przez siebie częścią inne osoby mogą udzielić wskazówek odnośnie zamierzonych działań innego zespołu. Cohn zaleca także aby podczas spotkania nie używać imion poszczególnych osób tylko żeby zwracać się do konkretnego zespołu

[17]. Po odpowiedzi na każde z czterech pytań na spotkaniu synchronizacyjnym jeśli zachodzi potrzeba to jest dyskusja na temat kwestii w zaległościach danego zespołu oraz rozwiązywanie problemów i utrzymywane są zaległości produktowe Scruma Scrumów. W którym znajdują się wszystkie kwestie i problemy. Z każdym nowym poziomem Scrum Scrumów staje się coraz bardziej przejrzysty co jest spowodowanie przejściem na wyższy poziom raportowania podczas spotkania synchronizującego.

Zastosowanie Scruma Scrumów pozwala nam na praktykowanie metodyki podczas pracy z dużymi zespołami w których zastosowanie standardowego Scruma mogło by spowodować ogromny komunikacyjny chaos. Zastosowanie takiej modyfikacji powoduje że jesteśmy w stanie podzielić wielką grupę ludzi na mniejsze zespoły i kontrolować ich pracę bez powodowania dodatkowych problemów komunikacyjnych. Inną dodatkową zaletą podczas prowadzenia dużych projektów jest możliwość skalowania. Jeżeli jest możliwość zrównoleglenia niektórych czynności to wystarczy dodać dodatkowy zespół aby wykonywać dodatkowe zadania przez doda koty zespół
Scrumowy [10]. Dzięki temu możemy korzystać z zalet Scrum i prowadzić duże projekty.

1.6. Testowanie oprogramowania z Agile

W standardowych metodach zarządzania projektami każdy członek zespołu ma swoją obsadzoną rolę dlatego też „tester” kojarzy nam się z osobą której główna aktywność skupia się na testowaniu oprogramowania i na utrzymaniu jakości. Podczas trwania projektu informatycznego najczęściej stosowaną rolą używaną rolą, zupełnie inną od testera, jest „programista” którego główna aktywność skupia się na pisaniu kodu produkcyjnego. Trzeba jednak pamiętać o tym że czynności wykonywane przez odpowiednie role nie ograniczają się wyłącznie do opisu roli. Tak też programiści wykonują więcej niż tylko pisanie kodu na podstawie specyfikacji dlatego też każdą osobę zaangażowaną w dostarczanie oprogramowania możemy nazwać deweloperem. Każdy zespół prowadzony w metodykach agile jest zorientowany na dostarczenie produktu przynoszącego wartość biznesową a testerzy w tej metodyce upewniają się że ich zespół dostarczy oprogramowanie takiej jakości jakiej klient potrzebuje. W dalszej części będę

stosował terminy „tester" i „programista" dla zachowania spójności między standardowymi rolami w projekcie [18].

Zespoły używające Metodyk agile stosują kilka kluczowych praktyk które są powiązane z testowaniem. Tak też programiści agile używają techniki *Test-Driven Development (TDD)* do pisania wysokiej jakości kod. W TDD programista w pierwszej kolejności pisze testy dla niewielkiej części funkcjonalności po czym widzi testy nie przechodzą poprawnie a następnie piszę kod który spełni warunki zawarte w testach. Następnym krokiem jest powtórzenie tego dla następnej małej części funkcjonalności. Do zadań programisty należy także pisanie testów integracji kodu w celu sprawdzenia poprawnej współpracy małych części kodów. Praktyka ta jest stosowana przez dużą liczbę zespołów, także tych które nie są związane z Agile [18].

Rozwój oprogramowania z Agile pokazuje zachęca do zespołowego rozwiązywania problemów. Wszystkie role począwszy od osób biznesowych przez programistów, testerów i na analitykach kończąc są zaangażowane w rozwój oprogramowania i powinny razem ustalać jak najlepiej ulepszyć dostarczany produkt. Z agile testerzy współpracują z grupą ludzi którzy czują odpowiedzialność w dostarczeniu jak najlepszej jakości oraz są zaangażowani w testowanie oprogramowania [19]. Mówiąc o testowaniu w agile najgęściej eksperci mają na myśli mówienie o testach wykonywanych przez osoby biznesowe, pożądanych dodatkach i funkcjonalności. Samo testowanie wykonywane przez testerów zawiera testy które prowadzą analizę krytyczną produktu i skupiają się na odkrywaniu braków, które można poprawić, w gotowym produkcie. Testowanie zawiera wszystko począwszy od testów jednostkowych, testów komponentów aż po testy: funkcjonalne, systemowe, obciążeniowe, wydajnościowe, bezpieczeństwa, użytkowe oraz akceptacyjne a wszystkie te testy mogą być zastosowane do dowolnego projektu.

Testowanie w metodyce agile wymaga zupełnie innego podejścia do testowanie oprogramowania a adaptacja tej metodyki oznacza odpowiednie przeszkolenie zespołów. Testerzy którzy są zupełnie nowi w metodykach agile nie wiedzą czego mogą się na początku spodziewać. Warto w takim przypadku poradzić się innego zespołu agile dzięki któremu możemy się dowiedzieć czegoś na temat testowania w projektach agile. Nowość w Scrumie polega głównie na przedstawianiu wymagań w formie historii użytkownika

gdzie testerzy nie mają do czynienia z dokumentem wymagań stworzonym w zaawansowanej formie a jedynie z prostymi określeniami wymagań wraz z ich kryteriami [12]. W celu porównania metod testowanie przedstawię porównanie metod testowania w tradycyjnych zespołach oraz w zespołach stosujących metodyki agile a także porównanie tradycyjnego testowania do testowania w metodykach Agile [18].

1.6.1. Praca w tradycyjnym zespole

W tradycyjnych metodach testowania testerzy nie uczestniczą ani w bliskiej współpracy z zespołami programistów ani w pracach projektowych na początku projektu. Wynika to z faktu że tradycyjne zespoły stosują tradycyjnych metod prowadzenia projektu i zajmują się tylko niewielkim obszarem testowania [18]. Po ustaleniu wymagań nie mamy wpływu na zgłaszane zmiany wymagań a w takim wypadku jedyne co mamy do powiedzenia to że możemy taką zmianę uwzględnić w następny wydaniu aplikacji [18].

Od liderów zespołów zapewnienia jakości oczekuje się pilnowania jakości produktu. Prawdą jest jednak że Zespoły QA nie są w stanie kontrolować tego w jaki sposób kod został napisany lub czy kiedykolwiek kod został przetestowany a takiej kontroli nie można wykonywać bez odpowiedniej współpracy z zespołem programistów. Praca po fazie developerskiej wymaga zwiększenia jakości produktu a takie podejście stwarza pozorny wpływ na kontrolę jakości. Jedyne co można w tej sytuacji zrobić to wprowadzić produkt na produkcję, opóźnić lub w ogóle zastopować wydanie wersji [18].

Tradycyjne zespoły są skupione na zapewnianiu że wszystkie wymagania są dostarczone w finalnym produkcie. Jeżeli jakaś funkcjonalność z jakiegoś powodu nie jest gotowa do zaplanowanej daty wydania to w tym przypadku takie wydanie jest opóźniane. Zespół rozwojowy nie ma zazwyczaj informacji wejściowych na temat dodatków które maja być w wydaniu a o których klient myśli że będą. Testerzy natomiast studiują obszerne dokumenty wymagań w celu napisania własnych planów testów a dopiero po wykonaniu tego następuje właściwe testowanie [18].

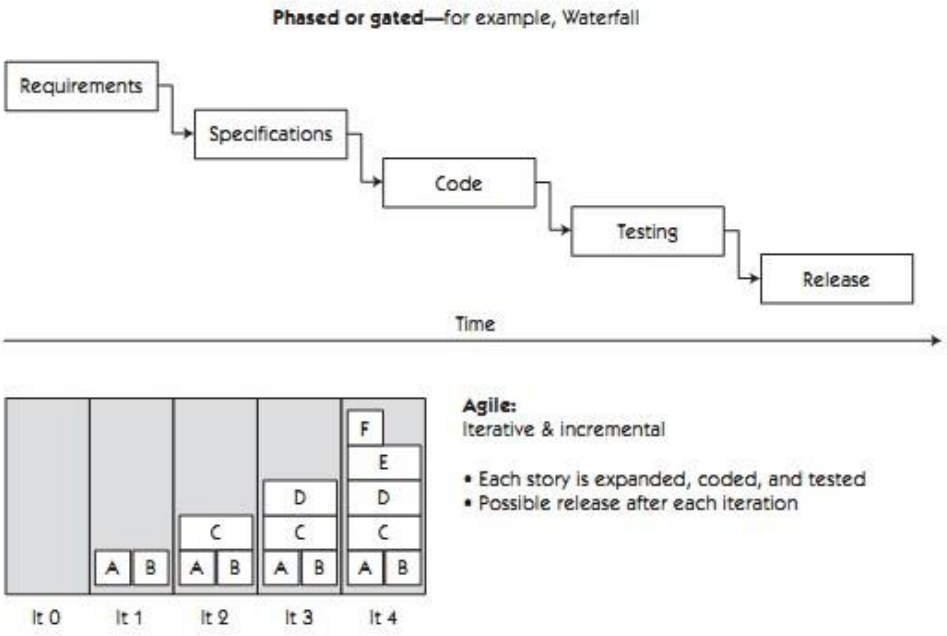
1.6.2. Praca w zespole Scrumowym

Zmiana, w porównaniu do tradycyjnych zespołów, w metodyce Scrum to głownie dostosowanie trybu pracy do krótkich iteracji co może być dla zespołu bardzo szokującym doświadczeniem na samym początku pracy nad projektem [19]. Ważnym pytaniem jest to w jaki sposób podczas tak krótkiego okresu można zdefiniować wymagania i dostarczyć je w stanie produkcyjnym. Jest to trudne zagadnienie w szczególności dla dużych organizacji w których są oddzielne zespoły z różnymi funkcjami lub dla zespołów rozproszonych po kilku geograficznych miejscach. Zespoły często nie mogą odnaleźć swojej roli w metodykach agile a także w jaki sposób realizować zamówienie w produkt gotowy do wydania przy ograniczonym przez iterację czasie [18]. W końcu w jaki sposób można sobie wyobrazić programistę, analityka, testera, project managera i innych specjalistów w zespole agile. Jak napisać kod i przetestować go w tak krótkim czasie?

Otóż zespoły Agile pracują blisko z osobami biznesowymi odpowiedzialnymi za daną funkcjonalność. Dzięki temu zespół może dokładnie zrozumieć wymagania a osobą która najbardziej mu w tym pomaga jest właściciel produktu. Zespół w metodyce Scrum jest odpowiedzialny za ustalenie wymagań najczęściej w formie historii użytkownika a wraz z nimi kryteriów poszczególnych wymagań [14]. Zespoły te są także skoncentrowane na wartości którą mogą dostarczyć a osoba w roli właściciela produktu ustala priorytet ważności poszczególnych wymagań. Testerzy nie muszą się w tym wypadku nudzić i czekać aż kod zostanie napisany. Ich zadaniem jest działanie przez całą iterację a wartością dodaną jaką daje Scrum dla ich pracy jest możliwość współpracy z klientem i dowiedzenie się szczegółów na temat danej funkcjonalności i jej sposobu działania, co skutkuje dostarczeniem oprogramowania o lepszej jakości [18].

1.6.3. Porównanie tradycyjnego testowania z testowaniem w Agile

Na samym początku porównania warto spojrzeć na Rys. 0.2. Na którym przedstawiony jest sposób testowania w tradycyjnych metodach zarządzania projektami oraz w metodykach opartych na Agile.



Rys. 0.2. Porównanie testowania agile i testowania w tradycyjnym zarzadzniu projektami [18]

Na diagramie opartym na fazach (Phases or gated) widzimy że testowanie znajduje się zaraz za kodowaniem (Code) ale przed wydaniem (Release). Oczywiście jest to podejście idealistyczne ponieważ nigdy nie mamy tyle samo czasu przeznaczone na każdą z poszczególnych faz a ponadto kodowanie zazwyczaj zajmuje więcej czasu niż się spodziewano. Faza testowania jest zazwyczaj skurczona ponieważ kodowanie zajmuje dłużej niż zostało to zaplanowane docelowo [18].

Scrum jest przeciwieństwem do tradycyjnych metod zarządzania, jest przede wszystkim metodyką iteracyjną to znaczy że testerzy testują każdy przyrost kodu jak tylko zostanie uważany za skończony. Sama iteracja może mieć różny czas trwania, może być krótka i trwać jeden tydzień ale może być też długa i trwać jeden miesiąc. Zespół buduje i testuje małą część kodu, upewnia się że pracuje poprawnie i przechodzi do następnego kawałka który potrzeba zbudować i przetestować. Wielkość projektu nie ma znaczenia na testowanie w agile. Pojedynczy zespół może prowadzić cały projekt lub też być częścią jednego większego projektu. W agile ważne jest aby reagować na zmiany co jest związane z adaptacyjną naturą metodyki ponadto należy pamiętać że nie ma nigdy dwóch takich

samych problemów, ponieważ dwóch różnych klientów pod jednym hasłem może mieć zupełnie co innego na myśli, i do każdego należy podejść w zupełnie inny sposób a takiego podejścia wymaga od nas agile [18].

W agile częściej niż pisanie testów na podstawie dokumentu wymagań tworzonego przez analityka biznesowego jest tworzenie testów przed rozpoczęciem pisania kodu. Wymaga to wspólnego wysiłku pomiędzy osobą biznesową a testerem, analitykiem lub innym członkiem zespołu rozwijającego funkcjonalność [18]. Szczegółowe przypadki testowe, najlepiej oparte na przykładach dostarczonych przez osobę biznesową, wspierają opisane wymagania. Na podstawie ich testerzy będą manualnie prowadzić testy eksploracyjne aby znaleźć najważniejsze błędy a zautomatyzowane testy funkcjonalne są dodawane do zestawu testów regresji. Gdy podczas testów demonstracyjnych minimalna wymagania dotyczące danej funkcjonalność są spełnione, zespół uważa historię użytkownika jako zrealizowaną [18].

Stosując Agile główną różnicą jest stosowanie krótkich ram czasowych, wszystkie aktywności przebiegają w sposób współbieżny a wszyscy uczestnicy, testy i narzędzia muszą zostać zaadaptowane w szybki sposób pod dany projekt. Oprócz tego najbardziej krytyczną różnicą dla developerów w projektach agile jest szybka reakcja z testów. To powoduje że projekt idzie naprzód utrzymując wysoką jakość czyli odbywa się to w zupełnie inny sposób niż przypadku metod tradycyjnych gdzie po fazie developerskiej następują testy a następnie relacja z testów jest z powrotem przekazywana do deweloperów jeżeli system nie spełnia wymagań [18]. Początkowe zastosowanie agile w projektach może sprawiać wrażenie chaosu, braku dyscypliny a braku dokumentacji brak formalnego spisania wymagań klienta ale z czasem gdy zespół się dostosuje do metodyki praca zaczyna się robić przyjemna komunikacja z klientem lepsza a jakość produktu wyższa [7].

1.7. Scrum i Kanban

Scrum jako metodyka zarządzania projektami nie nakłada na zespół sprawdzonych technik zarządzania ale pozwala na samoorganizację zespołu [9].
Początkowo taka sytuacja może być dla członków zespołu nowa bo w końcu jak to jest że nikt nimi nie kontroluje [7]? W jaki sposób mamy teraz zarządzać nasza pracą?

Owszem jest osoba ScrumMastera która pilnuje aby projekt był prowadzony zgodnie z zasadami zdefiniowanymi w Scrum jednak nie może ona decydować na temat działań zespołu. Ma ona za to możliwość pomóc odnaleźć się zespołowi w nowej sytuacji [12]. Często stosowaną praktyką przez zespoły Scrumowe jest zaczerpnięcie sposobu sterowania produkcją z metodyki Kanban w celu samo organizacji pracy zespołu [16]. Zastosowanie takie pozwala zespołowi w prosty i przyjemny sposób na realizację zadań oraz wcielenie się w nową samoorganizującą się jednostkę. Jak wygląda sam Kanban? W jaki sposób połóczyć Scrum i Kanban razem? Jakie są główne różnice między czystą metodyką Kanban a metodyką Scrum? Na te pytania postaram się odpowiedzieć w następnych podrozdziałach. Na początku zacznę od opisu metodyki.

1.7.1. Czym jest Kanban

Metodyka Kanban została opracowana w Japonii w latach 50 XX wieku metoda zarządzania produkcją. Samo słowo Kanban pochodzi z języka japońskiego i oznacza kartkę papieru a w wolnym tłumaczeniu znaczy "widoczny opis" [20].

Docelowo metodyka ta w prosty sposób pozwala na wizualizację przepływu materiałów w przedsiębiorstwie. Kanban ma za zadanie sterowanie zapasami i pozwala na prawie całkowitą eliminację magazynów. Magazynowanie przedprodukcyjne, poprodukcyjne jak i międzyoperacyjne jest znikome, gdyż wszelkie materiały od dostawców są dostarczane „dokładnie na czas” i to samo dzieje się, jeśli chodzi o wysyłkę wyrobu gotowego [20].

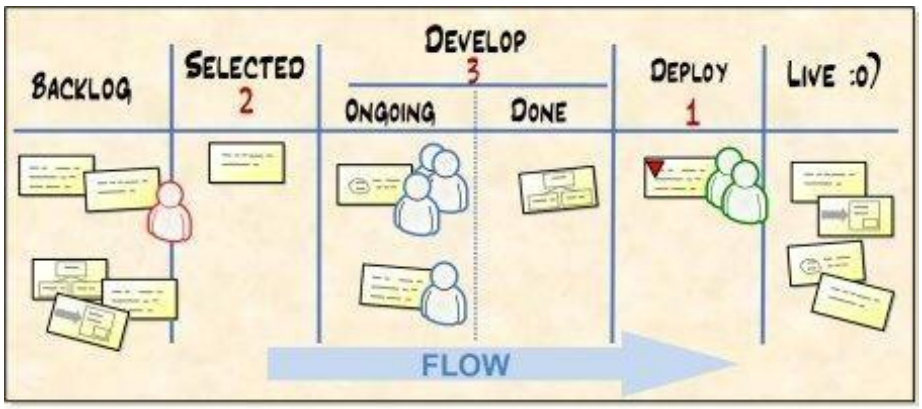
Cele systemu Kanban można przedstawić za pomocą hasła "7 x żadnych" [20]:

- żadnych braków,
- żadnych opóźnień,
- żadnych zapasów,
- żadnych kolejek - gdziekolwiek i po cokolwiek,
- żadnych bezczynności,
- żadnych zbędnych operacji technologicznych i kontrolnych, żadnych przemieszczeń.

System Kanban jest samoregulującym się narzędziem operacyjnego sterowania produkcją. System ten jest sterowany zdarzeniami występującymi bezpośrednio na produkcji (a nie o plan). Zorientowany jest na zapewnienie [20]:

- krótkiego czasu przetwarzania,
- niskich zapasów przy jednoczesnej terminowości realizacji
- wielkość produkcji dopasowana do liczby zamówień, oraz kontrola jakości na wszystkich etapach procesu.

Główną zaletą metodyki Kanban jest jednak wizualizacja pracy która pokazuje status wykonania poszczególnych zadań. Wykonuje się to stosując tablicę kankanową taką jak przedstawiona na Rys. 0.3. na której widzimy przebieg prac podzielonych na poszczególne fazy. Do wykonania całego produktu jesteśmy w stanie podzielić prace na małe części i wykonywać je po małym kawałku.



Rys. 0.3. Tablica Kanban [16]

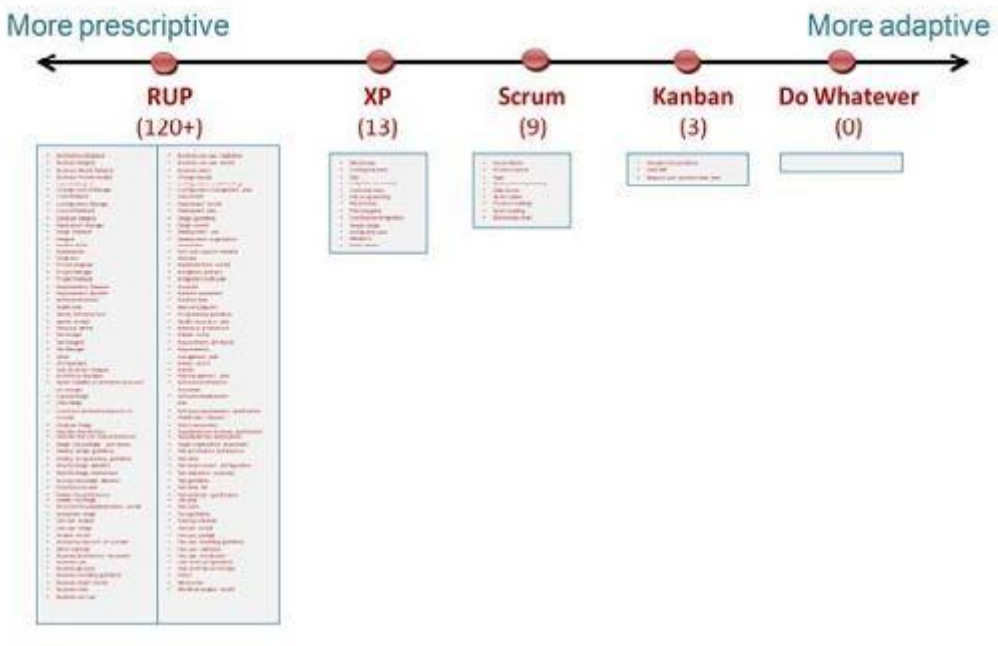
W metodyce Kanban dzięki zastosowaniu limitu prac wykonywanych w danej fazie produkcji możemy w łatwy sposób kontrolować zastoje w produkcji angażując inne osoby w miejsce w którym się pojawia zastój. Kanban podobnie jak Scrum jest narzędziem procesowym które pozwala na bardziej wydajną pracę jednak mogą być zastosowane obydwie razem. W metodykach agile wykorzystywana jest wizualizacja przepływu pracy co pomaga zespołom w samoorganizacji zadań. Zanim przejdę do opisu połączenia Kanbana z metodyką Scrum przedstawie różnice pokazujące adaptacyjną naturę Kanban w porównaniu do metodyki Scrum [16].

1.7.2. Różnice między Kanban a Scrum

Scrum zarówno jak i Kanban należą do adaptacyjnych narzędzi procesowych jednak Scrum narzuca na użytkownika ograniczenia ze względu na konieczność zastosowania się do wielu zasad wynikających z tej metodyki i podstaw metodyk agile. Żadna z tych metodyk nie jest ani kompletna ani doskonała. Nie mówią nam wszystkiego co musisz zrobić w danej sytuacji, po prostu niektóre dostarczają ograniczenia i wytyczne w jaki sposób prowadzić projektem [16]. Na przykład ograniczenia w Scrum to czasowe iteracje i wielofunkcyjne zespoły, a ograniczenia w Kanban to korzystanie z tablicy kanbanowej i limit rozmiaru kolejki poszczególnej fazy produkcji.

Możemy porównywać narzędzia pod względem zasad, które dostarczają. Patrząc pod tym aspektem możemy wyróżnić dwie kategorie metodyk, „nakazowe" co oznacza "więcej zasad do przestrzegania" i „adaptacyjne" co oznacza "mniej zasad do przestrzegania". Metodyka w pełni nakazowa oznacza, że nie trzeba zastanawiać się co w danej sytuacji na leży poczynić ponieważ wszystko jest opisane regułą. Zupełnym przeciwieństwem są metodyki nakazowe gdzie można pokierować się zasadą „zrób wszystko aby dojść do celu". W rzeczywistości nie jesteśmy w stanie kierować w odpowiedni sposób ani jedną skrajnością ani drugą [16].

Metodyki Agile, nazywane także metodami lekkimi, są metodami adaptacyjnymi ponieważ są one mniej nakazowe niż metody tradycyjne to znaczy że posiadają mniej zasad. Tak naprawdę pierwszy dogmat manifestu agile („Ludzi i ich wzajemne interakcje (współdziałanie) ponad procedury i narzędzia.") wskazuje już na taką klasyfikację metodyki. Obie metodyki, Scrum jak i Kanban, są wysoce elastyczne, ale relatywnie Scrum jest surowszy niż Kanban ponieważ Scrum narzuca więcej zasad, a przez to jest mniej otwarty. Przykładem takim może być to że Scrum narzuca używanie iteracji o stałym czasie natomiast Kanban nie. Na Rys. 0.4. przedstawiona została oś z wyżej opisanymi kategoriami klasyfikacji metodyk oraz klasyfikację tychże metodyk na „nakazowe" (*prescriptive*) oraz „adaptacyjne" (*adaptive*) [16]. Rysunek ten nie zawiera dokładnych opisów zasad ma na celu tylko prezentację porównania skali zasad dla metodyk bardziej nakazowych i bardziej adaptacyjnych.



Rys. 0.4. Oś przedstawiająca metodyki od bardziej nakazowych (od lewej) do bardziej adaptacyjnych (do prawej) [16].

Z Rys. 3.4. wynika że metodyka RUP jest najbardziej nakazowa z metodyk sklasyfikowanych - ma ponad 30 ról, ponad 20 aktywności i ponad 70 artefaktów. Nie sposób wszystkiego wykorzystać dlatego też należy korzystać tylko z tych które będą odpowiednie dla danego projektu co może się w praktyce okazać trudne. Taki stan może spowodować dla którego wdrożenie RUP może być dużo cięższe od wdrożenia metodyk agile takich właśnie jak Scrum czy XP które jest bardziej nakazowe od Scruma ponieważ zawiera dodatkowe praktyki które w metodyce Scrum nie są stosowanie [16]. Jedną z głównych różnic między Scrum i RUP jest to, że w RUP jest zbyt dużo zasad a te niepotrzebne można bez problemu usunąć. Natomiast Scrum dostarcza tylko kilka zasad i można bez problemu dodawać swoje [16].

Scrum jest mniej nakazowy niż XP, ponieważ nie przewiduje żadnych szczególnych dodatkowych praktyki. Natomiast w porównaniu do Kanban, Scrum jest surowszy ponieważ nakazuje takie rzeczy jak iteracje czy wyznaczenie ról [16]. Kanban pozostaje otwarty ponieważ jego jedyne ograniczenia to wizualizacja przepływu pracy i ograniczenie pracy wykonywanej w danej fazie. Najbardziej adaptacyjne jest niestosowanie żadnej

metodyk która z góry narzuca zasady. Główne różnice między Scrumem a Kanbanem przedstawię w dalszej części. W pierwszej kolejności zostaną przedstawione podobieństwa wynikające z zastosowania ich przy projektach IT [16]:

- Obie są lekkie
- Obie są adaptacyjne.
- Obie skupiają się na dostarczaniu działającego oprogramowania wcześniej i często.
- Obie bazują na samo zarzadzających się zespołach.
- Obbie dzielą pracę na części.

Wymienione wyżej podobieństwa są przedstawienie na wysokim stopniu abstrakcji. Gdyby spojrzeć na podobieństwa metodyk na niższym stopniu abstrakcji to wynika że obie metodyki realizują te same czynności w zupełnie inny sposób. Dlatego też obie metodyki są zupełnie różneco zostało przedstawione na Tabela nr. 1:

Scrum	Kanban
Iteracje czasowe są narzucone.	Iteracje czasowe są opcjonalne. Może posiadać oddzielne kadencje dla planowania, wydania i ulepszenia procesu. Iteracja może być sterowana zdarzenami zamiast iteracji czasowej.
Zespół zobowiązuje się do wykonania pewnej części pracy na koniec każdej iteracji.	Zobowiązania zespołu są opcjonalne
Zespoły wielofunkcyjne są z góry narzucone/	Zespoły wielofunkcyjne opcjonalne. Dozwolone są zespoły specjalistów.
Wymagania muszą być podzielone taka aby mogły zostać wykonane podczas jednego sprintu.	Brak konkretnego opisu wielkości zadania.
Wykres wypalenia jako podstawa monitoringu pozostałej do wykoania pracy.	Brak narzuconego diagram monitorowanej pracy
Ilość ograniczonej pracy ograniczona w sposób pośredni (ilość pracy do wykonania na sprint).	Ilość pracy ograniczona bezpośrednio przez poszczególne fazy produkcji.
Estymacja zadań narzucona.	Estymacja zadań opcjonalna
Nie można dodać wymagania do wykonania podczas trwania sprintu	Można dodawać nowe element podczas trwania iteracji.

Narzuca 3 role.	Nie narzuca żadnych ról
Nie narzuca żadnego monitoring wykonywania zadania.	Monitoring zadania za pomocą tablicy kankanowej.
Narzucone prioretyzowanie zadań.	Priorytety zadań mogą być opcjonalnie.

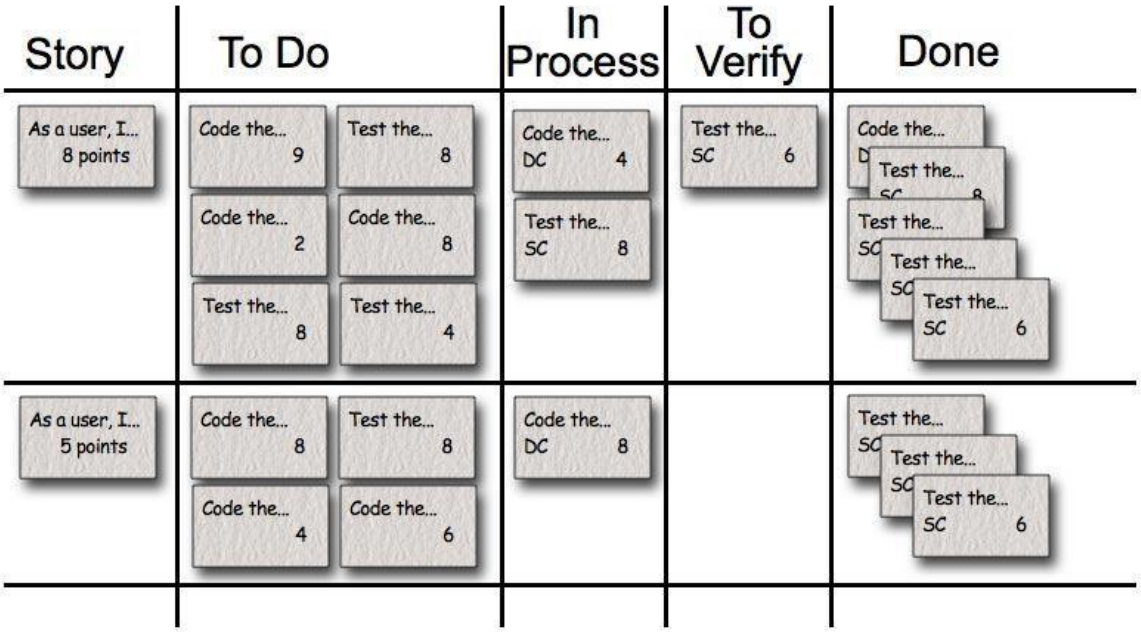
Tabela nr. 1 Porównanie zasad narzuconych w metodyce Scrum i w metodyce Kanban [16]

Z różnic przedstawionych powyżej można wywnioskować że Scrum jest bardziej nakazową metodyką niż Kanban. Scrum narzuca więcej zasad do zastosowania w projekcie natomiast Kanban wymaga dostosowania według upodobań i potrzebnych narzędzi. Najważniejsze podobieństwa łączące Kanban i Scrum to ich lekkość i łatwość w zastosowaniu. Scrum ponadto narzuca zasadę samoorganizacji zespołu. Najłatwiejszym przypadkiem do zastosowania się do tej zasady używając scrum jest monitoring wykonania zadań zaczerpnięty z metodyki Kanban czyli prosta tablica z realizacją przepływu zadania.

1.7.3. Sposób połączenia Scrum i Kanban

Założeniem Scrum jest samoorganizujący się zespół. Członkowie zespołu nie muszą posiadać odpowiednich kompetencji do organizowania pracy całego zespołu. Osoba w roli ScrumMastera może w takim wypadku doradzić zespołowi zastosowanie zaczerpniętej z metodyki Kanban tablicy przepływu pracy. Zepół może dostosować taką tablicę do aktualnych potrzeb czyli do aktualnego podziału zadań na fazy produkcji.

Przykładem takiej tablicy może być tablica skonstruowana na potrzeby projektu jak przedstawiona na Rys. 0.5.



Rys. 0.5. Przykładowa tablica Scrumowa generowana przez oprogramowanie do wspomagania pracy w metodyce Scrum.

Na niej możemy zauważyć podział historii użytkownika na odpowiednio podzielone zadania a także podział pracy. Zespół może sam ustalić kroki (fazy produkcji) jakie należy wykonać na w celu zakończenia zadania. Przykładem takich faz produkcji wymagania może być podział na zadania jakie należy wykonać do jego zakończenia, fazę projektową, fazę produkcyjną, fazę weryfikacyjną, wdrożeniową a po przejściu wszystkich faz zadanie może zostać oznaczone jako zakończone. Taka wizualizacja pracy wraz z wykresem wypalenia pomaga na określenie członkom zespołu tego w jakim stanie są prace które zobowiązali się wykonać [16]. W przypadku gdy okaże się że na przykład zadanie wykonywane przez jednego z członków zespołu nie przeszło poprawnie weryfikacji może ono zostać cofnięte z powrotem do fazy produkcyjnej a w przypadku gdy podczas testów okaże się że coś działa zupełnie inaczej niż było to wymyślone zadanie może zostać cofnięte nawet do fazy projektowej (o ile taka faza została założona przez zespół).

Jeżeli zespół znajduje się w jednej lokalizacji geograficznej to w takim wypadku najlepiej jest zrealizować taką tablicę korzystając z białej tablicy na której rysujemy odpowiednią siatkę pokazującą przepływ zadania a zadani można wypisać na samoprzylepnych kolorowych karteczkach a każdy kolor powinien być przyporządkowany

danemu członkowi zespołu [16]. Korzystając z takiego podziału kolorystycznego zespół może w łatwy sposób rozróżnić kto w danym momencie zajmuje się zadaniem. Można też skorzystać z jednokolorowych karteczek a każde zadanie wybrane przez członka zespołu do realizacji powinno być oznaczone inicjałem. Przykład takiej tablicy możemy zobaczyć na Rys. 0.6.

Rys. 0.6. Przykład tablicy scrumowej dla zespołu znajdującego się w jednej lokalizacji geograficznej. Tablica została wykonana przy wykorzystaniu zwykłej tablicy do rysowania markerem

W przypadku kiedy zespół nie znajduje się w jednej lokalizacji można skorzystać z oprogramowania wspomagającego zarządzanie w Scrumie. Zazwyczaj takie oprogramowania udostępniają opcję korzystania z tablicy Scrumowej czyli dostosowanej na potrzeby Scruma tablicy Kanbanowej.

Podsumowując połączenie Scrum i Kanban polega na odpowiednim przedstawieniu przepływu pracy za pomocą graficznej tablicy. Połączenie to pomaga zespołowi w samoorganizacji a prostota tego rozwiązania powoduje że każdy szybko zrozumie na czym polega taka metoda.

6. Bibliografia

[1] THE STANDISH GROUP REPORT, "CHAOS," 1995.

[2] R. Wysocki, R. McGary, "Efektywne zarządzanie projektami", Helion, Gliwice, 2005.

[3] VitalSmarts, "Silence Fails," 2005.

[4] G. Asproni, "Wstęp do Scrum," *Software Development Journal*, pp. 64-69, Czerwiec 2006.

[5] agilemanifesto.org. Witryna Agile Manifesto. [Online]. http://agilemanifesto.org/

[6] W. Dąbrowski, K. Subieta, "Podstawy inżynierii oprogramowania", Wydawnictwo PJWSTK, Warszawa, 2005.

[7] K. Schwaber, "Sprawne zarządzanie projektami metodą Scrum", Microsoft Press, Warszawa, 2005.

[8] H. Takeuchi, I. Nonaka, "The new new product development game," *Harvard Business Review*, pp. 137-146, January-February 1986.

[9] K. Schwaber, J. Sutherland, "Scrum Guide", Scrum.org, 2010.

[10] Mountain Goat Software. Mountain Goat Software. [Online]. http://www.mountaingoatsoftware.com

[11] M. Vizdos. Implementing Scrum. [Online]. http://www.implementingscrum.com/

[12] M. Cohn, "Succeeding with Agile", Addison-Wesley, Boston, 2010.

[13] R. Szczepanik, "Budowanie zespołu. Organizacja szkoleń outdoor i wypraw incentive. Poradnik dla menedżera personalnego.", OnePress, Gliwice, 2005.

[14] M. Cohn, "User Stories Applied", Addison-Wesley, Boston, 2004.

[15] M. Cohn, "Agile Estimating and Planning", Prentice Hall, Upper Saddle River, 2006.

[16] H. Kniberg, M. Skarin, "Kanban and Scrum - making the most of both", C4Media Inc., 2010.

[17] M. Cohn. ScrumAlliance. [Online]. http://www.scrumalliance.org

[18] L. Crispin, J. Gregory, "Agile testing. A practical guide for testers and agile teams.", Addison-Wesley, Boston, 2009.

[19] R. A. Collaris, E. Dekker. (2010, January) www.agilerecord.com. [Online]. www.agilerecord.com

[20] J. Gross, K. Mcinnis, "Kanban Made Simple", AMACOM, New York, 2003.

[21] W. Stott, J. Newkirk, "Visual Studio Team System Better Software Development for Agile Teams", Addison-Weslay, Boston, 2007.