Podstawy Agile i Scrum

Eliza Kasperuk

Wstęp

Podstawy Agile i Scrum

Zaliczenie pierwszego semestru - test

Zaliczenie drugiego semestru - pomysł na własny projekt

Kanały komunikacji: e-mail, **Slack**, MSTeams, FB.
Plany zajęć będą umieszczanie na serwerze z planami.
Więcej informacji w sekretariacie lub na #general na Slacku.

Zadawanie pytań na bieżąco

Zajęcia 1szy semestr

Zjazd 1szy - 14-15.10.2022

Zjazd 2gi - 25-26.11.2022

Zjazd 9ty - 10-11.02.2023

Oceny - semestr I

PROCENTY	PUNKTY	OCENA
0 – 30%	0 - 40 pkt	niedostateczny - 2,0
31 – 45%	41 - 60	dostateczny - 3,0
46 – 60%	61 - 80	dostateczny plus - 3,5
61 – 75%	81 - 100	dobry - 4,0
76 – 90%	101 - 120	dobry plus - 4,5
91 – 100%	121 - 134	bardzo dobry - 5,0

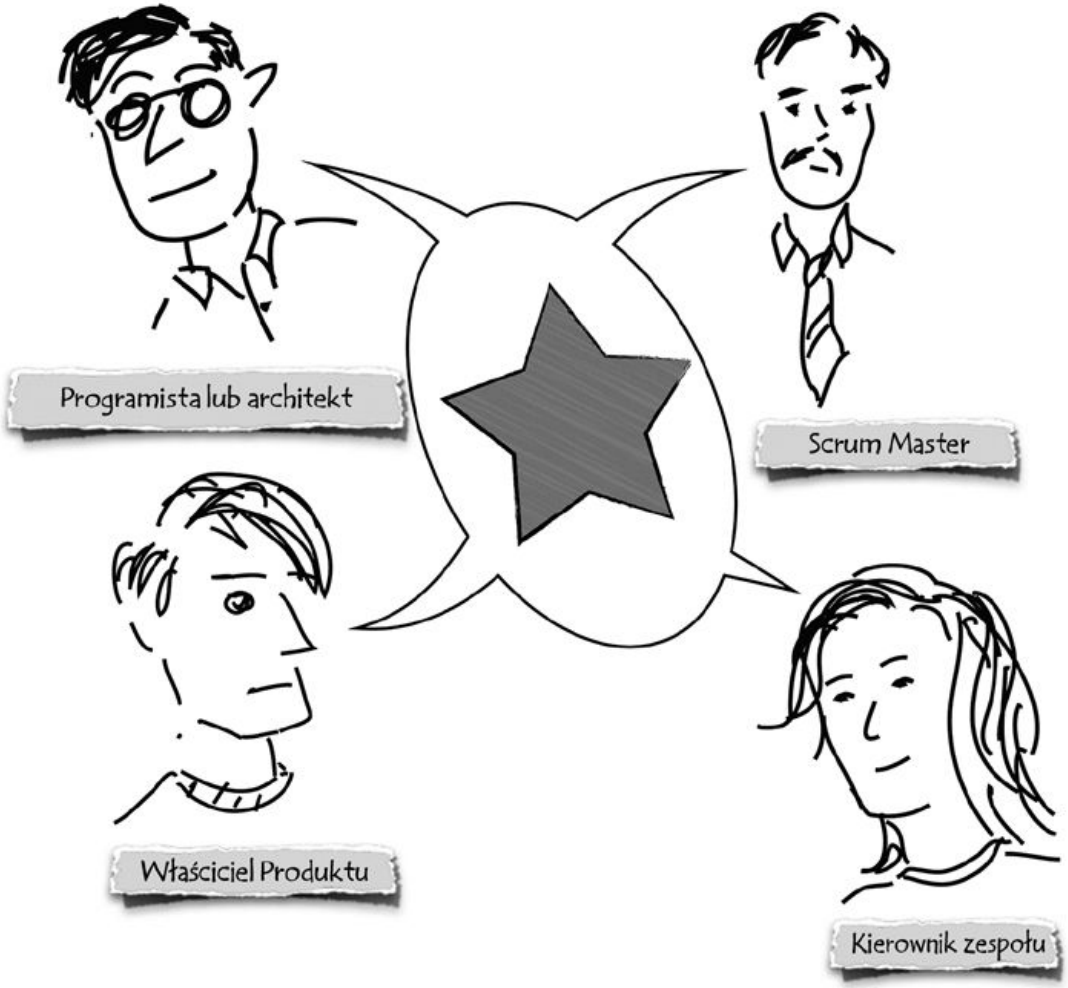
“Najważniejszym, możliwym do wypracowania nastawieniem jest potrzeba ciągłego uczenia się”

- John Dewey, *Experience and Education*

Cele

- Zrozumienie **idei** leżących u podstaw efektywnego prowadzenia zespołów **zwinnych**, a także wartości i zasady, które je do siebie zbliżają.
- Poznanie **zasady** najpopularniejszych szkół myślenia zwinnego — **Scrum, XP, Lean i Kanban** — oraz zrozumienie że mimo sporych różnic wszystkie te podejścia zaliczają się do metodyk zwinnych.
- Nauka pewnych **praktyk zwinnych**, z których można zacząć korzystać w swoich projektach już dziś — ważne jest też przekazanie wartości i zasad, które pomogą efektywnie zaimplementować te praktyki.
- Lepsze zrozumienie swojego zespołu i firmy, tak by każdy z Was mógł/ła **wybrać najlepsze podejście zwinne, które pasuje do Waszego sposobu myślenia** (lub jest możliwie zbliżone), a także byście razem ze swoimi zespołami weszli na tory nowego sposobu myślenia, które sprawi, że staniecie się sprawniejszym zespołem zwinnym.


WPROWADŹMY CODZIENNE SPOTKANIA, NA KTÓRYCH BĘDĘ MÓGŁ UZYSKAĆ OD WAS INFORMACJE O CODZIENNYCH POSTĘPACH. TO WSPANIAŁA METODA, KTÓRĄ MOŻEMY WSZYSCY...
MAMY JUŻ ZBYT WIELE SPOTKAŃ! JEŚLI MI NIE UFASZ, ZNAJDŹ KOGOŚ INNEGO NA MOJE MIEJSCE
Kierownik projektu
Programista

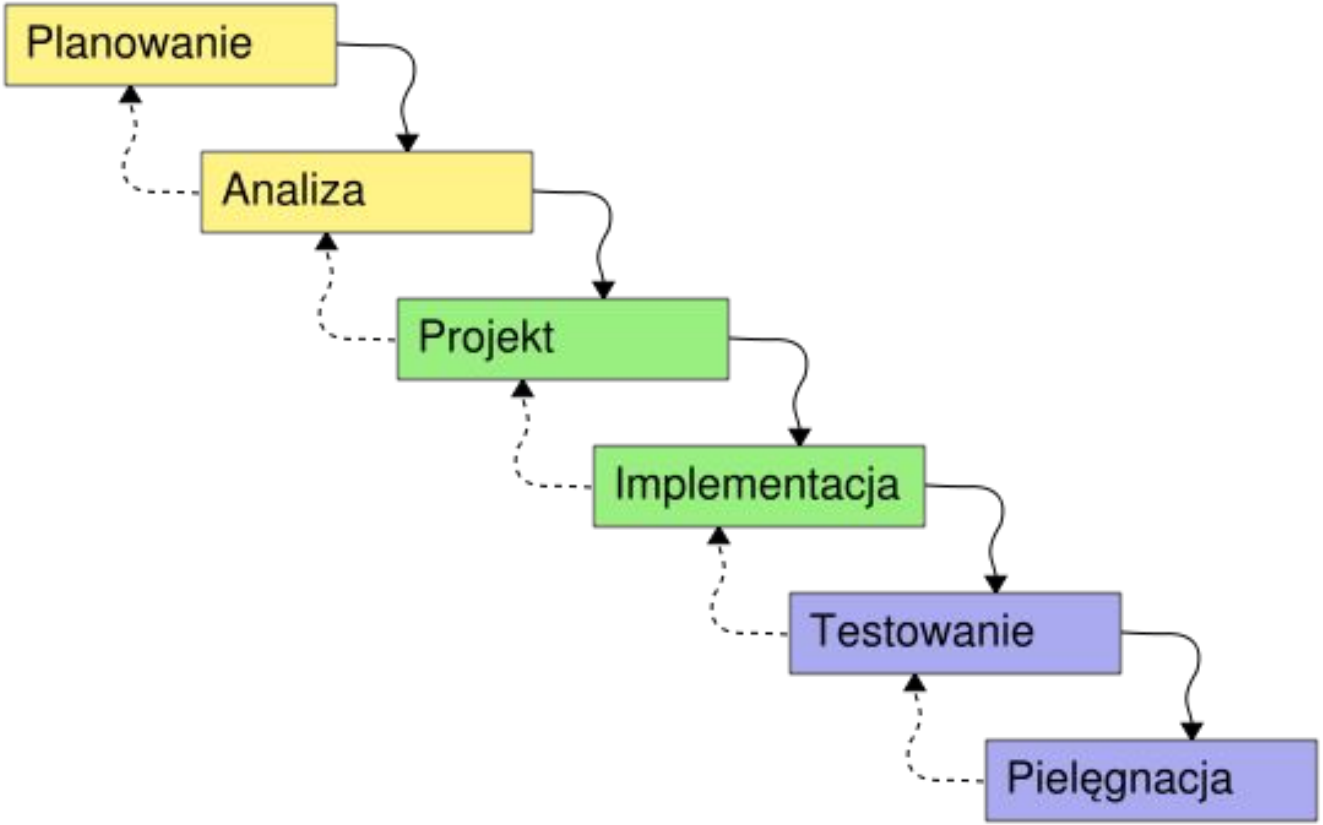

Programista lub architekt
Scrum Master
Właściciel Produktu
Kierownik zespołu

Waterfall

Proces wytwarzania oprogramowania. Polega on na wykonywaniu podstawowych czynności jako odrębnych faz projektowych, w porządku jeden po drugim.

- Planowanie systemu (w tym specyfikacja wymagań).
- Analiza systemu (w tym analiza wymagań i studium wykonalności).
- Projekt systemu (poszczególnych struktur itp.).
- Implementacja (wytworzenie kodu).
- Testowanie (poszczególnych elementów systemu oraz elementów połączonych w całość).
- Wdrożenie i pielęgnacja powstałego systemu.

Waterfall


Planowanie
Analiza
Projekt
Implementacja
Testowanie
Pielęgnacja

Kluczowe Wady Waterfall

- Nieelastyczny podział na kolejne rozłączne iteracyjne fazy, w której tworzone są często niepotrzebne dokumenty;
- Przejście do następnej fazy możliwe po zakończeniu poprzedniej;
- Wysoki koszt iteracji przez powtarzanie wielu czynności;
- Planowanie ‘z góry’ - trudno dostosować się do zmienności warunków;
- Bardzo opóźniona informacja zwrotna - testowanie na koniec;

Manifest programowania zwinnego - Agile

Deklaracja wspólnych zasad dla zwinnych metod tworzenia oprogramowania. Została opracowana na spotkaniu 17 developerów, które miało miejsce w dniach 11-13 lutego 2001 roku w ośrodku wypoczynkowym Snowbird w USA (stan Utah). Uczestniczyli w nim reprezentanci nowych metod tworzenia oprogramowania, będących alternatywą dla tradycyjnego podejścia opartego na modelu kaskadowym. Do wspomnianych metod należą: rapid application development (RAD - 1970 i 1980), **Scrum** (1995), Dynamic Systems Development Method (1994), Adaptive Software Development, Crystal Clear (1996), programowanie ekstremalne (1996), Feature Driven Development (1997), Pragmatic Programming. Od nazwy manifestu metody te zaczęto określać mianem metod zwinnych (ang. *agile*).

Manifest programowania zwinnego - Agile

Odkrywamy nowe metody programowania dzięki praktyce w programowaniu i wspieraniu w nim innych. W wyniku naszej pracy, zaczęliśmy bardziej cenić:

Ludzi i interakcje od procesów i narzędzi

Działające oprogramowanie od szczegółowej dokumentacji

Współpracę z klientem od negocjacji umów

Reagowanie na zmiany od realizacji założonego planu.

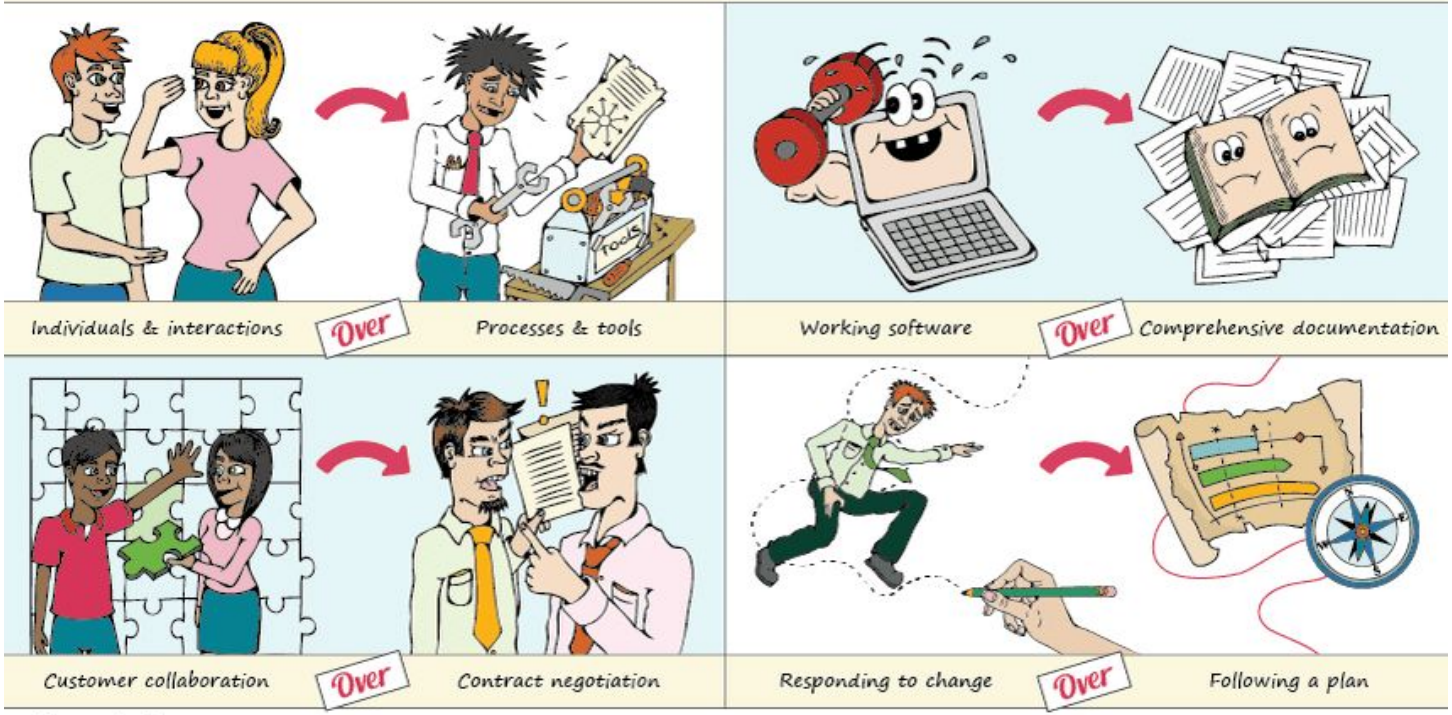
Oznacza to, że elementy wypisane po prawej są wartościowe, ale większą wartość mają dla nas te, które wypisano po lewej.

Źródło: http://agilemanifesto.org/

Manifesto for Agile Software Development*

"We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:



That is, while there is value in the items on the right, we value the items on the left more."

Knowledge TRAIN®

12 założeń Manifestu Agile

1) Najwyższy priorytet ma dla nas zadowolenie klienta dzięki wczesnemu i ciągłemu wdrażaniu wartościowego oprogramowania.

2) Bądźcie gotowi na zmiany wymagań
nawet na późnym etapie jego rozwoju.
Procesy zwinne wykorzystują zmiany
dla zapewnienia klientowi konkurencyjności.

12 założeń Manifestu Agile

3) Dostarczajcie funkcjonujące oprogramowanie często, w kilkutygodniowych lub kilkumiesięcznych odstępach. Im częściej, tym lepiej.

4) Zespoły biznesowe i deweloperskie muszą ściśle ze sobą współpracować w codziennej pracy przez cały czas trwania projektu.

12 założeń Manifestu Agile

5) Twórzcie projekty wokół zmotywowanych ludzi. Zapewnijcie im potrzebne środowisko oraz wsparcie i zaufajcie, że wykonają powierzone zadanie.

6) Najbardziej efektywnym i wydajnym sposobem przekazywania informacji zespołowi deweloperskiemu

i wewnątrz niego jest rozmowa twarzą w twarz.

12 założeń Manifestu Agile

7) Działające oprogramowanie jest podstawową miarą postępu.

8) Procesy zwinne umożliwiają zrównoważony rozwój. Sponsorzy, deweloperzy oraz użytkownicy powinni być w stanie utrzymywać równe tempo pracy.

12 założeń Manifestu Agile

9) Ciągłe skupienie na technicznej doskonałości i dobrym projektowaniu zwiększa zwinność.

10) **Prostota** – sztuka minimalizowania ilości koniecznej pracy – jest kluczowa.

12 założeń Manifestu Agile

11) Najlepsze rozwiązania architektoniczne, wymagania i projekty pochodzą od samoorganizujących się zespołów.

12) W regularnych odstępach czasu zespół analizuje możliwości poprawy swojej wydajności, a następnie dostraja i dostosowuje swoje działania do wyciągniętych wniosków.

https://agilemanifesto.org/iso/pl/principles.html

Agile

Programowanie zwinne (ang. agile software development) – grupa metod wytwarzania oprogramowania opartego na programowaniu **iteracyjno-przyrostowym**, powstałe jako alternatywa do tradycyjnych metod typu **waterfall**. Najważniejszym założeniem metodyk zwinnych jest obserwacja, że **wymagania odbiorcy (klienta) często ewoluują** podczas trwania projektu. Oprogramowanie wytwarzane jest przy współpracy **samozarządzalnych** zespołów, których celem jest przeprowadzanie procesów wytwarzania oprogramowania.
Pojęcie zwinnego programowania zostało zaproponowane w 2001 w Agile Manifesto.

Agile

- Podejście do zarządzania i pracy nad projektami
- Proste podejście do zarządzania złożonością
- Przykładowe projekty
  - Projekty programistyczne
  - Zarządzanie usługami
  - HR i procesy zatrudnienia
  - Zadania na zorganizowanie pikniku
  - Napisanie książki
  - Zbudowanie rakiety

Etapy realizacji zadań w Agile

- Plan (Planowanie)
- Design (Projektowanie)
- Develop (Programowanie)
- Test (Testowanie)
- Release (Implementacja)
- Feedback (Informacja zwrotna)

Etapy realizacji zadań w Agile

- **Planowanie** - zbieranie dokładnych wymagań od klienta dotyczących danego zadania, ich analiza oraz zaplanowanie kroków koniecznych do realizacji danego zadania w oparciu o pozyskane informacje.

- **Projektowanie** - projektowanie wykonania danego elementu będącego celem zadania na bazie informacji zebranych na etapie Planowania.

- **Programowanie** - właściwy etap prac

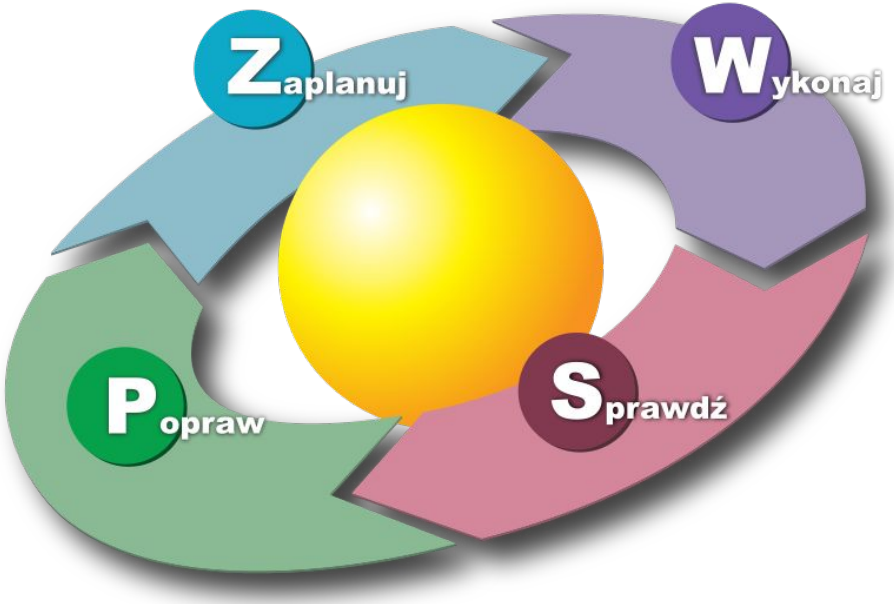
Etapy realizacji zadań w Agile

- **Testowanie** - testowanie danego elementu będącego podmiotem zadania od strony technicznej przez osobę lub zespół wykonujący dane zadanie oraz od strony klienta (User Acceptance Test) czy dany element jest tym czego klient oczekiwał

- **Implementacja** - po pozytywnych wynikach testów zarówno technicznych jak i klienckich (akceptacji przez klienta) przekazanie danego elementu projektu na "produkcję" do finalnego użytkowania przez klienta

Etapy realizacji zadań w Agile

- **Informacja zwrotna** - przekazanie informacji zwrotnej przez klienta do osoby lub zespołu realizującego dane zadanie w projekcie odnośnie ewentualnych mniejszych błędów, których nie wykryto podczas testów, zgłaszanie potencjalnych usprawnień do realizacji w kolejnych cyklach lub zgłoszenie zmiany wymagań klienta.



Cykl Deminga
(PDCA = Plan-Do-Check-Act)

Kaizen - ciągłe doskonalenie

Cykl Deminga

Podstawowa zasada ciągłego ulepszania (ciągłego doskonalenia, kaizen), stworzona przez Williama Edwardsa Deminga, amerykańskiego specjalistę statystyka pracującego w Japonii.

1. ZAPLANUJ (ang. Plan): Zaplanuj lepszy sposób działania, lepszą metodę.
2. WYKONAJ, ZRÓB (ang. Do): Zrealizuj plan na próbę.
3. SPRAWDŹ (ang. Check): Zbadaj, czy rzeczywiście nowy sposób działania przynosi lepsze rezultaty.
4. POPRAW (ang. Act): Jeśli nowy sposób działania przynosi lepsze rezultaty, uznaj go za normę (obowiązującą procedurę), zestandaryzuj i monitoruj jego stosowanie.

Metodyki programowania zwinnego

- Extreme Programming
- Lean
- Kanban
- Scrum
- Dynamic Systems Development Method…

Extreme Programming (XP)

Metodyka programowania mające na celu wydajne tworzenie małych i średnich "projektów wysokiego ryzyka", czyli takich, w których nie wiadomo do końca, co się tak naprawdę robi i jak to prawidłowo zrobić.

(ang. *eXtreme Programming*, ***XP***)

Techniki Extreme Programming (XP)

- **Iteracyjność** - Program tworzy się w iteracjach (krótkie, przyrostowe kroki programistyczne) - i co ważniejsze - planuje tylko następną iterację.
- **Nie projektować z góry** - Nie można z góry przewidzieć, jaka architektura będzie najlepsza dla danego problemu. Dlatego należy ją tworzyć w miarę rozszerzania programu.
- **Testy Jednostkowe** - Pisze się zanim w ogóle zacznie się pisać kod - najlepiej na początku iteracji. Potem pisze się kod, który potrafi je wszystkie przejść - TDD.

Techniki Extreme Programming (XP)

- **Ciągłe modyfikowanie architektury** - Architektura nie jest czymś, czego nie wolno zmieniać.
- **Programowanie parami** - Programiści piszą w parach: jedna osoba pracuje przy klawiaturze i jest głównym koderem, druga obserwuje pierwszą, zgłasza poprawki, zadaje pytania wyjaśniające. Programiści programujący w parze zamieniają się rolami co kilkadziesiąt minut.
- **Stały kontakt z klientem** - Specyfikacje są prawie zawsze wieloznaczne, dziurawe i sprzeczne ze sobą. Tak więc należy mieć stały kontakt z tym, dla kogo to oprogramowanie jest tworzone

Techniki Extreme Programming (XP)

- **Dziesięciominutowy proces budowania -** Zespół tworzy automatyczny proces budowania całego kodu bazowego, wykonywany w mniej niż dziesięć minut. Ten proces obejmuje automatyczne uruchamianie wszystkich testów jednostkowych i generowanie raportu informującego, które testy zakończyły się powodzeniem, a które porażką.

Techniki Extreme Programming (XP)

- **Ciągła integracja** - Jest ona oparta na narzędziach pozwalających wielu osobom jednocześnie pracować nad jednym zbiorem plików z kodem źródłowym. Jeśli wszyscy muszą pracować nad tymi samymi plikami, członkowie zespołu nie mogą jednocześnie modyfikować tych samych fizycznych kopii plików, bo prowadzi to do nieustannego zastępowania zmian wprowadzonych przez inne osoby. Dlatego zespół musi stosować system kontroli wersji z centralnym (lub zdecentralizowanym) nadrzędnym zbiorem plików. Poszczególni programiści pobierają (ang. check-out) te pliki, czyli tworzą prywatne kopie całego kodu bazowego, używanego tylko przez daną osobę w jej izolowanym środowisku (ang. sandbox). Programista pracuje nad swoją kopią kodu w izolowanym środowisku i okresowo przesyła ją z powrotem do repozytorium.

Techniki Extreme Programming (XP)

- **Tygodniowy cykl** - polega na zastosowaniu jednotygodniowych iteracji. Jest ona ściśle powiązana z techniką historii (są one identyczne jak opisane wcześniej historie użytkowników). Każdy cykl rozpoczyna się od spotkania poświęconego planowaniu, na którym członkowie zespołu omawiają poczynione w projekcie postępy, wspólnie z klientami wybierają historie do danej iteracji, a następnie rozbijają historie na zadania. Potem szacowany jest czas potrzebny na wykonanie zadań i są one przydzielane programistom.
- **bufor (ang. slack) -** Technika ta polega na tym, że zespół dodaje do każdego tygodniowego cyklu niewielkie historie o niskim priorytecie. W trakcie spotkania poświęconego planowaniu są one rozbijane na zadania, pozostawiane jednak na koniec iteracji. Dzięki temu jeśli zespół natrafi na nieoczekiwane trudności, może łatwo pominąć buforowe historie i mimo to dostarczyć na koniec cyklu kompletne, działające oprogramowanie, które jest w pełni gotowe (działa, przechodzi testy i może zostać zaprezentowane użytkownikom).

Techniki Extreme Programming (XP)

- **Cykl kwartalny** - Raz na kwartał zespół spotyka się, aby omówić projekt na ogólnym poziomie. Zespoły XP mówią o motywach (ang. theme), czyli o ogólnych zagadnieniach z rzeczywistego świata, które pozwalają powiązać historie z projektu ze sobą. Omawianie motywów pomaga zespołowi ustalić, jakie historie należy dodać do projektu. Zapewnia też kontakt z problemami biznesowymi, które oprogramowanie ma rozwiązywać. Zespół może wtedy nabrać perspektywy i porozmawiać o wewnętrznych oraz zewnętrznych problemach, z którymi się styka, uciążliwych błędach, a także niewprowadzonych jeszcze poprawkach. Można wówczas poświęcić czas na analizę dotychczasowych postępów, realizowania potrzeb użytkowników i ogólnego przebiegu projektu. Niektóre zespoły stosujące XP posługują się retrospekcją używaną w zespołach scrumowych.

Lean software development

Metoda została zaadoptowana przez Mary Poppendieck i Toma Poppendiecka. W szczupłym wytwarzaniu oprogramowania wyróżnia się 7 zasad wspomaganych przez 22 narzędzia.

Lean

1) Eliminacja strat - Marnotrawstwo to wszystko, co nie dodaje wartości do produktu. To klient definiuje wartość.

2) Tworzenie jakości i spójności - Jakość i spójność produktu finalnego polega na uzyskaniu równowagi pomiędzy funkcjami aplikacji, jej niezawodnością i wartością ekonomiczną wytworzoną dla klienta firmy.

3) Wzmocnienie pozyskania wiedzy - Tworzenie oprogramowania jest jednocześnie procesem uczenia się, poznajemy nową organizację, nowe zasady rządzące danym biznesem dla którego tworzymy aplikację. Dlatego też bardzo istotnym jest uzyskanie jak najlepszego sprzężenia zwrotnego.

Lean

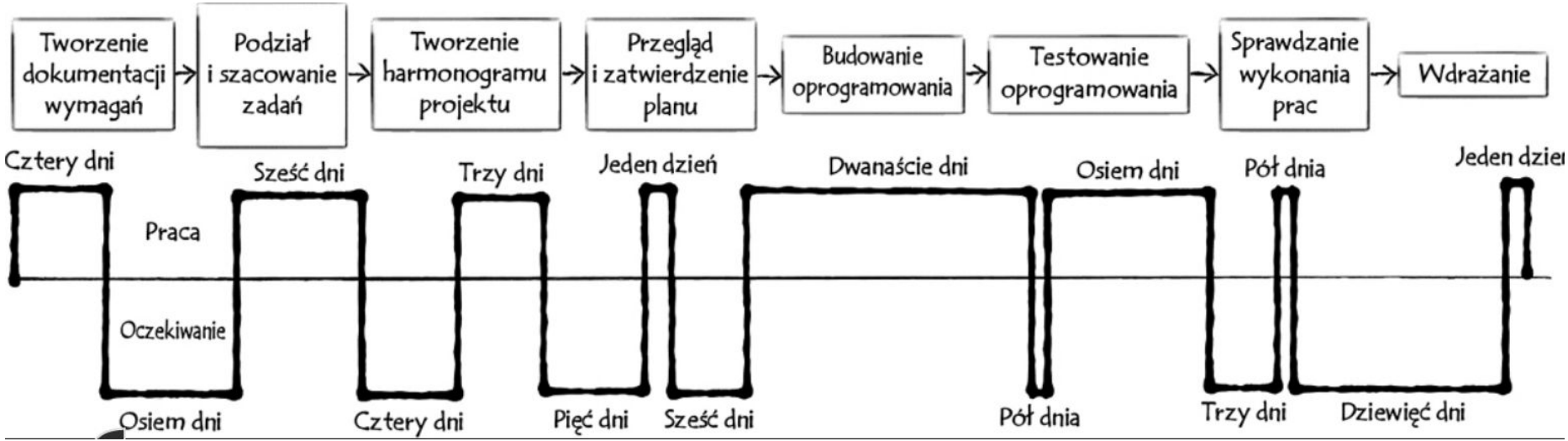
4) Podejmowanie decyzji najpóźniej jak to możliwe - Podczas tworzenia oprogramowania zespół musi podjąć szereg decyzji, choćby takich jaką technologię zastosować, z którą bazą danych związać produkt itp.

5) Wdrażanie najwcześniej jak to możliwe - Zalecane jest dostarczenie produktu szybko i w małych porcjach, implementując je w poszczególnych iteracjach.

6) Respektowanie zespołu - Idealnym zespołem jest zespół samoorganizujący się

7) Spojrzenie na całość - Należy widzieć produkt jako całość

Lean - mapa strumienia wartości



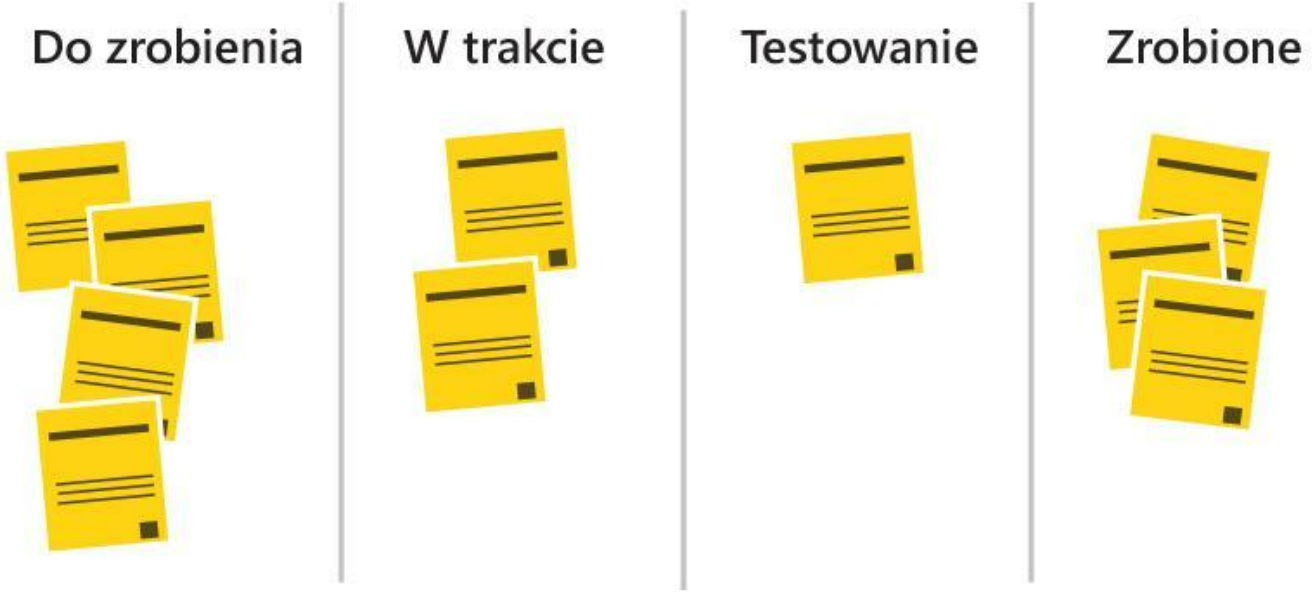
Kanban

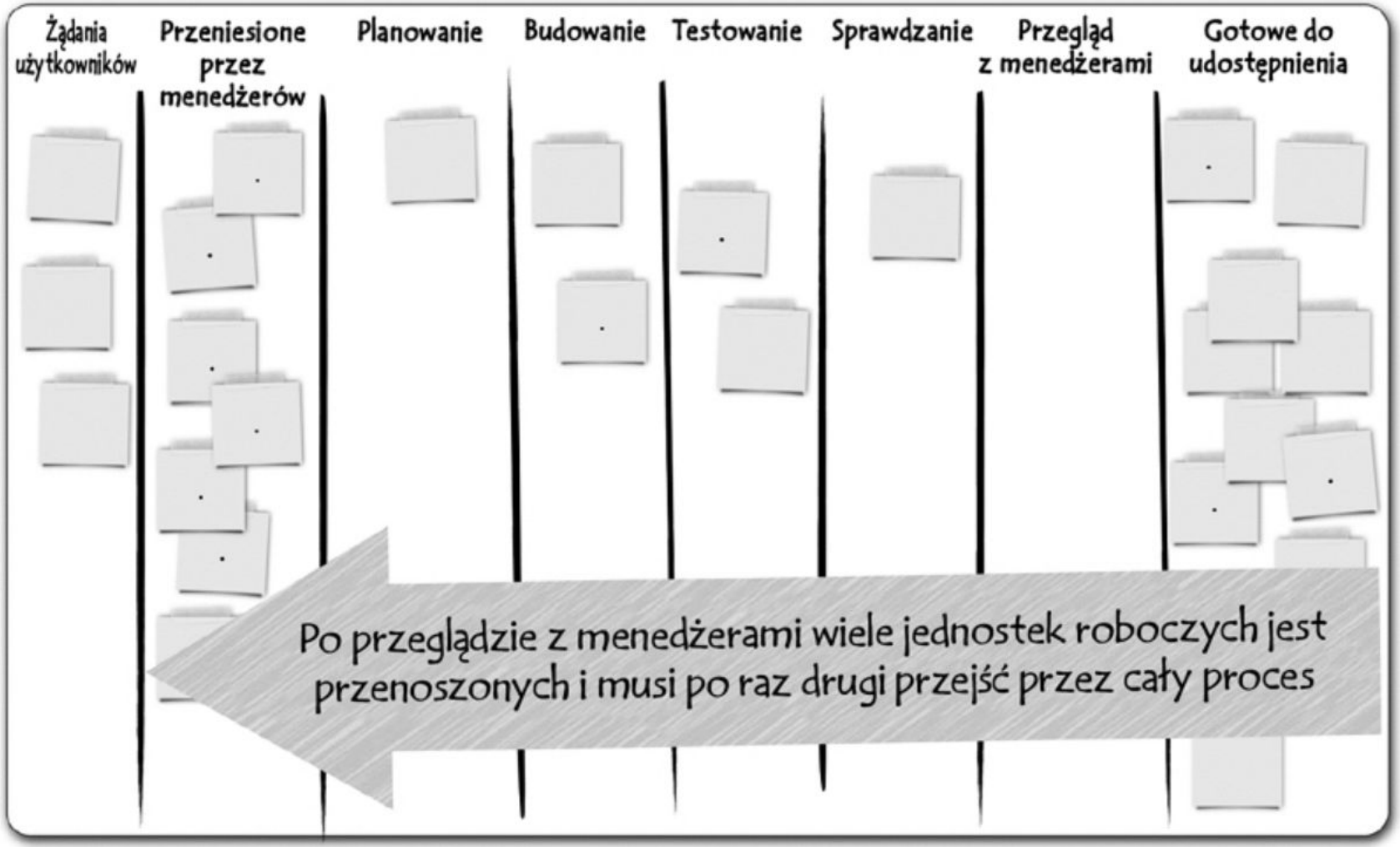
Głównym celem zastosowania koncepcji Kanban w inżynierii oprogramowania jest terminowe dostarczenie klientom oprogramowania o wysokiej jakości. Kanban umożliwia sprawowanie pełnej kontroli nad procesem tworzenia oprogramowania, pozwala na wyeliminowanie przyczyn nieefektywności i zwiększenie produktywności.

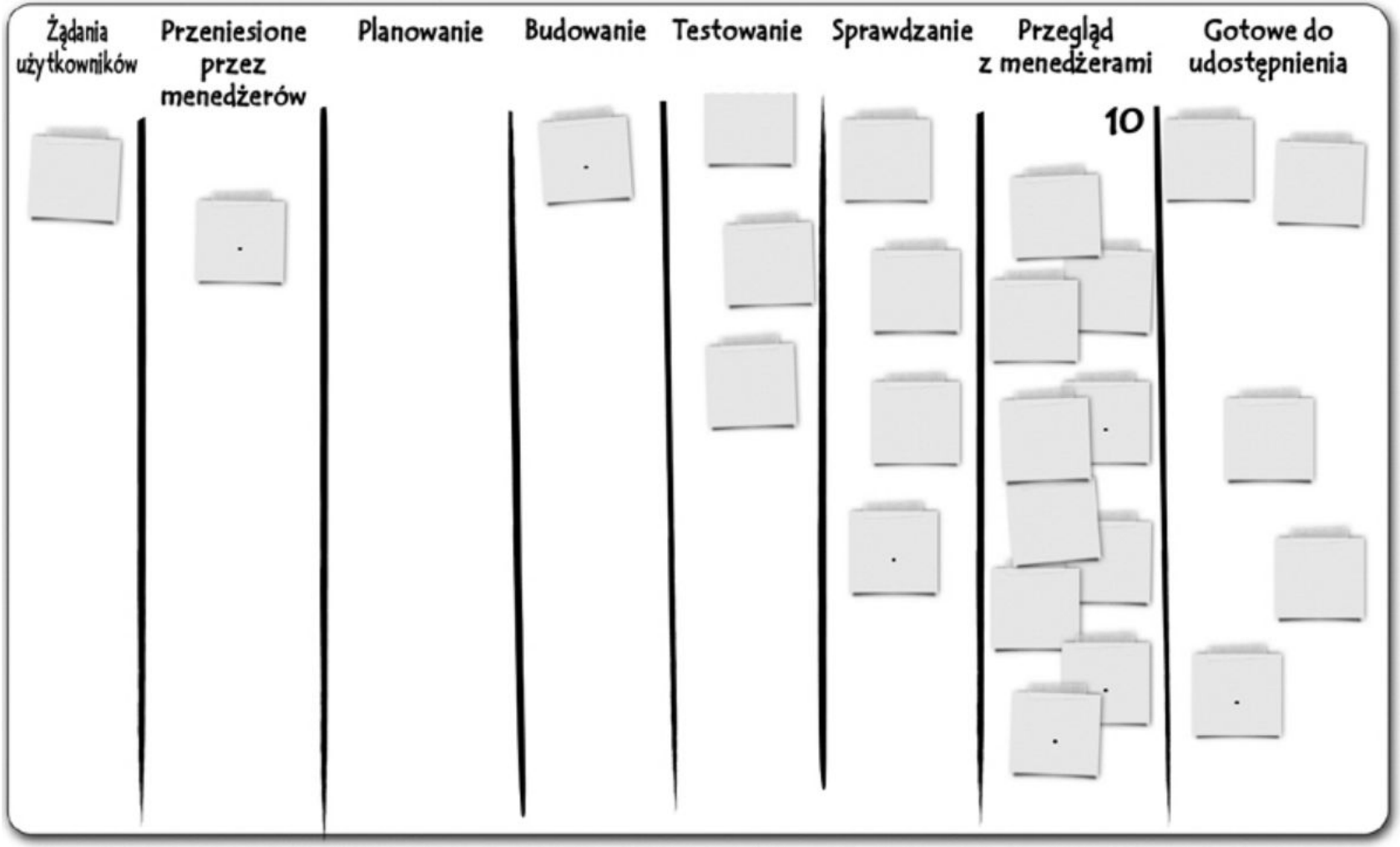
Kanban

- **Wizualizacja** – przedstawienie kolejnych etapów procesów na tablicy (ściennej lub elektronicznej) np. analiza, wytwarzanie, testowanie, wdrażanie, zadania skończone. Następnie zapisanie zadań na kartkach i umieszczenie w odpowiednich kolumnach
- **Ograniczenie pracy w toku** – ustalenie maksymalnej dopuszczalnej liczby zadań, które mogą znajdować się w danej kolumnie. W tym celu wykorzystywane jest m.in. prawo Little'a.
- **Zarządzanie strumieniem** – systematyczny pomiar takich wartości, jak czas i płynność wykonywania zadań, w celu optymalizacji procesów

Tablica Kanban




Żądania użytkowników
Przeniesione przez menedżerów
Planowanie
Budowanie
Testowanie
Sprawdzanie
Przegląd z menedżerami
Gotowe do udostępnienia
Po przeglądzie z menedżerami wiele jednostek roboczych jest przenoszonych i musi po raz drugi przejść przez cały proces


Żądania użytkowników
Przeniesione przez menedżerów
Planowanie
Budowanie
Testowanie
Sprawdzanie
Przegląd z menedżerami
10
Gotowe do udostępnienia

Agile vs Waterfall - zalety

Agile

- Pozwala na systematyczne wprowadzanie zmian przez klienta
- Dzięki ciągłym modyfikacjom pozwala na ulepszenie programu i nadążanie za zmianami w branży
- Pod koniec sprintu planowane są priorytety projektu dzięki czemu na bieżąco możny uzyskiwać pożądany efekt
- Testowanie na każdym etapie pozwala szybko eliminować błędy
- Produkt może zostać uruchomiony na koniec dowolnego cyklu

Waterfall

- Podkreśla istotność dokumentacji
- Klient wie czego i kiedy się spodziewać zna harmonogram prac
- W przypadku rotacji pracowników silna dokumentacja powoduje mniejszy wpływ na projekt

Agile vs Waterfall - wady

Agile

- Ciężej jest mierzyć postęp prac ponieważ występuje on pomiędzy kilkoma cyklami
- Wymagane są większe umiejętności od Developerów i Klienta w zakresie definiowania i realizowania zadań
- Projekt łatwo może wyjść poza plan jeżeli klient nie wie czego oczekuje

Waterfall

- Po ukończeniu kroku programiści nie mogą do niego wrócić i zrealizować zmian
- Wszystko zależy od początkowych wymagań
- W przypadku znalezienia błędu koncepcyjnego projekt musi zostać zrealizowany od początku
- Cały projekt jest testowany na końcu.
- Plan nie uwzględnia ewoluujących potrzeb klienta

Agile vs Waterfall - kiedy używać

Agile

- Kiedy szybka produkcja jest ważniejsza od jakości
- Kiedy klienci będą mogli zmieniać zakres projektu
- Kiedy nie wiadomo jak ma wyglądać projekt końcowy
- Kiedy zespół developerski jest doświadczony i potrafi pracować niezależnie
- Kiedy produkt jest kierowany do często zmieniającego się środowiska

Waterfall

- Kiedy wiemy dokładnie jak ma wyglądać projekt
- Kiedy klienci nie będą mieli możliwości zmiany zakresu projektu po jego rozpoczęciu
- Kiedy zakres a nie czas jest głównym celem projektu

Agile

- **Przyrostowy (Incremental)** - użyteczna funkcjonalność;
- **Iteracyjny** - ciągłe udoskonalanie produktu i procesu jego budowania;
- **Skupienie na wartości** - Zespół pracuje nad użytecznymi elementami produktu, wsparte informacją zwrotną od użytkownika;
- **Upełnomocniony Zespół** - Zespół planuje i decyduje o pracach, nie jest sterowany, każdy członek zespołu jest decyzyjny;

Agile się jest

Przydatne linki i książki

- Manifest Agile http://agilemanifesto.org/
  http://agilemanifesto.org/iso/pl/manifesto.html
  http://agilemanifesto.org/iso/pl/principles.html

- Andrew Stellman, Jenifer Green, Agile. Przewodnik po zwinnych metodykach programowania, Helion 2015

Dziękuję!

eliza.kasperuk.jsd@podyplomowe.pb.edu.pl