INŻYNIERIA OPROGRAMOWANIA

Zwinne metodyki tworzenia oprogramowania. Programowanie ekstremalne

Dilbert by Scott Adams

Wykorzystane materiały:

- materiały szkoleniowe (nie istniejącego już) Zespołu ds. Projektów IT Equilibrium
- http://xprince.net (XPrince)
- http://brasil.cel.agh.edu.pl/~09sbfraczek/programowanie-ekstremalne,1,58.html

Kryzys oprogramowania

- Kryzys oprogramowania (ang. software crisis) — zjawisko polegające na powiększającej się rozbieżności między mocą obliczeniową sprzętu komputerowego a efektywnością produkcji oprogramowania. Zostało zauważone w latach 60-tych XX wieku.
- W odpowiedzi starano się zwiększyć dyscyplinę w wytwarzaniu oprogramowania. W ciągu następnych 20-tu lat zaproponowano wiele standardów (IEEE, ISO, itd.), tradycyjnych modeli oraz metodyk zorientowanych na dyscyplinę oraz pierwsze podejścia zarządzania projektami do wytwarzania oprogramowania (np. PRINCE2).
- Zbyt duża restrykcyjność ogranicza inicjatywę i elastyczność, które są bardzo potrzebne do budowy skomplikowanych systemów ze zmiennymi wymaganiami. Aby temu zaradzić, w latach 90-tych XX wieku powstały tak zwane zwinne metodyki (ang. agile methodologies).

Tradycyjne metodyki programowania

- Przedsięwzięcie informatyczne w centrum uwagi
- Ludzie (programiści) jako odnawialny zasób
- Długoterminowe, ale powolne planowanie
- Mała elastyczność projektu
- Zapis formalnej dokumentacji

Najwięcej o systemie wiedzą jego projektanci,
podczas gdy rola programistów sprowadza się
do rzemieślniczego implementowania
cudzych pomysłów

Konsekwencje i problemy podejścia tradycyjnego

- Zmiana w systemie jest czasochłonna, gdyż należy go ponownie przeprojektować i zaimplementować
- Kontakt z użytkownikiem ograniczony jest do rzadkich spotkań mających na celu zebranie wymagań i przeprowadzenie testów akceptacyjnych
- Potencjał i kreatywność programisty ograniczone są przez projektanta
- W stosunku do programowania, debugowanie i wyszukiwanie błędów zajmuje dużo czasu
- Zamawiane oprogramowanie jest zwykle oddawane po terminie
- Budżet bywa często przekraczany
- Programiści często muszą pracować w dodatkowym czasie
- Jakość stworzonego oprogramowania jest zła
- Funkcjonalność systemu często odbiega od oczekiwań klienta

Manifest zwinnego (*agile*) podejścia do rozwoju oprogramowania

INŻYNIERIA OPROGRAMOWANIA

2001: Manifesto for Agile Software Development – deklaracja wspólnych zasad dla zwinnych metodyk tworzenia oprogramowania, opartych na:

- Ludziach i interakcjach pomiędzy nimi – zamiast procesów i narzędzi
- Działającym oprogramowaniu – zamiast szczegółowej dokumentacji
- Współpracy z klientem – zamiast negocjowania kontraktów
- Reagowaniu na zmiany – zamiast realizowania planu

O ile elementy wymienione po prawej są istotne, te występujące z lewej strony są dużo ważniejsze

Więcej informacji: http://agilemanifesto.org/

Rozwinięcie zasad podejścia zwinnego

- Szybko wytworzone oprogramowanie, skutkujące satysfakcją klienta
- Działające oprogramowanie jest dostarczane okresowo, np. w dwutygodniowych przyrostach (model przyrostowy)
- Podstawową miarą postępu jest działające oprogramowanie (przyrosty)
- Późne zmiany w specyfikacji nie mają destrukcyjnego charakteru na proces wytwarzania oprogramowania, szybka adaptacja
- Bliska, stała (codzienna) współpraca pomiędzy klientem a wytwórcą
- Bezpośredni kontakt jako najlepsza forma komunikacji w zespole i poza nim
- Ciągła uwaga nastawiona na aspekty techniczne oraz dobry projekt
- Prostota projektu i oprogramowania
- Uznanie kodu za dokumentację projektu
- Samozarządzalność zespołów

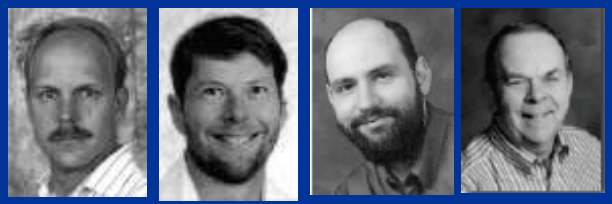
Agile Alliance

INŻYNIERIA OPROGRAMOWANIA

- Organizacja, skupiająca prawie 5500 członków (firmy, instytucje, osoby prywatne; stan na dzień 22/03/2010), założona w 2001 r.
- Promuje zastosowania metodyk zwinnego programowania poprzez wydawanie publikacji, organizowanie szkoleń, realizacją programów proponowanych przez członków

http://www.agilealliance.org/

Kent Beck - twórca metod analizy obiektowej, autor narzędzi xUnit, autor metodyki eXtreme Programming
Alistair Cockburn - autor rodziny zwinnych metodyk Crystal, oraz książek poświęconych inżynierii wymagań
Martin Fowler - twórca pomysłu refaktoryzacji, autor "UML Distilled"
Jim Highsmith - autor metodyki Adaptive Software Development



Główni twórcy manifestu Agile oraz Agile Alliance

Najważniejsze metodyki i praktyki zwinnego programowania

Metodyki:

- Dynamic Systems Development Method (DSDM)
- SCRUM
- eXtreme Programming (XP) – Programowanie ekstremalne
- Feature Driven Development (FDD)
- Agile Unified Process (AUP)

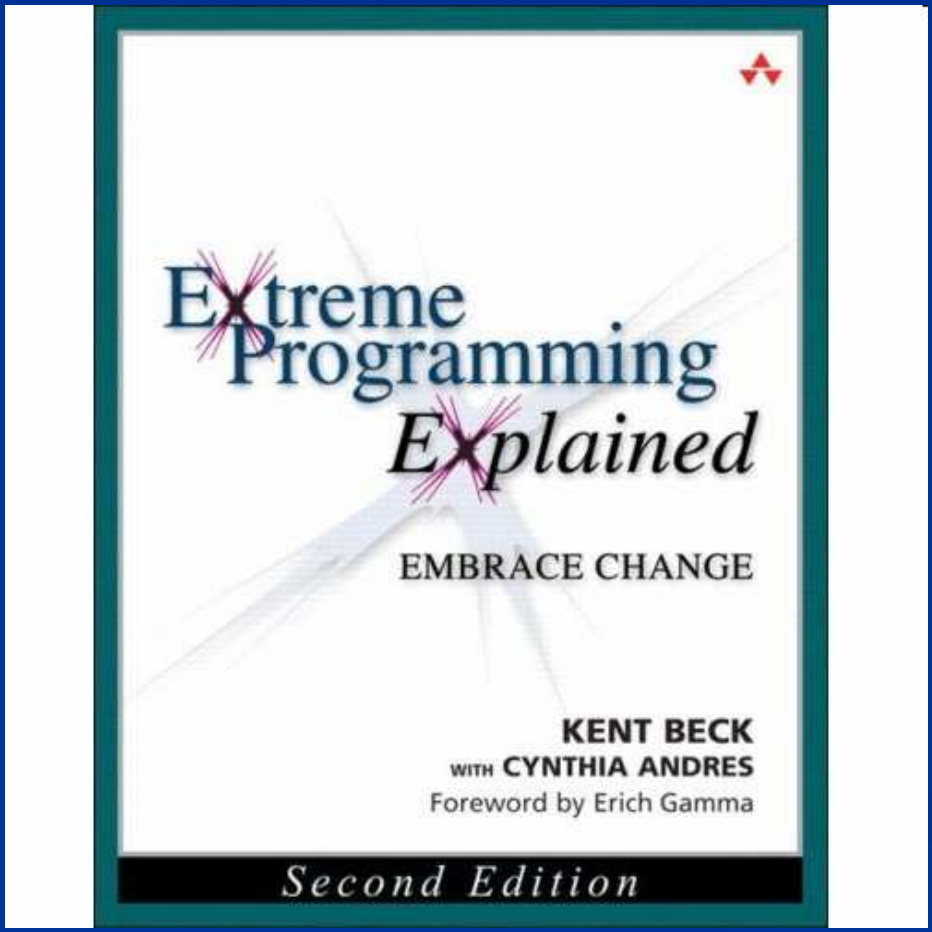
Praktyki:

- Planning poker, Planning game – Gra planistyczna
- Test Driven Development (TDD) – Programowanie sterowane testami
- Behavior Driven Development (BDD) – Programowanie sterowane zachowaniem
- Pair Programming – Programowanie w parach
- Continuous Integration, Refactoring – Ciągła integracja, refaktoring

Dlaczego omówimy eXtreme Programming?

INŻYNIERIA OPROGRAMOWANIA

Tom deMarco: „eXtreme Programming jest dzisiaj najważniejszym ruchem w całej inżynierii oprogramowania”

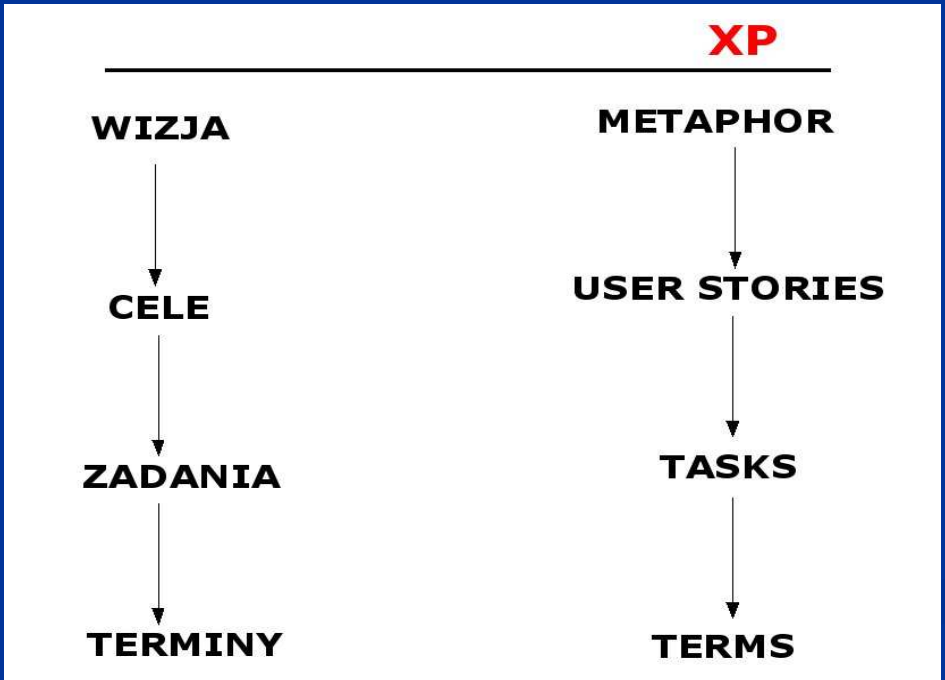


eXtreme Programming zawiera w sobie najważniejsze praktyki podejścia zwinnego

Twórca metodyki: Kent Beck, 1999

eXtreme Programming (XP) – terminologia

Zanim rozpocznie się jakiekolwiek działanie, należy najpierw określić rezultat, jaki spodziewamy się uzyskać oraz określić ogólną koncepcję (wizję), do której dążymy. Aby móc wdrażać ją w życie, określamy w jej obrębie konkretne cele, które z kolei dzielą się na zadania, zaplanowane do wykonania w ściśle określonym miejscu i czasie.



Nazewnictwo w XP: wizji odpowiada metafora systemu (metaphor), celom – opowieści użytkownika (user stories); zadania oraz terminy zachowują swe nazwy

XP – od wizji do produktu

Czteroetapowy model pracy:

- Określ cel
- Wykonaj działanie
- Odbierz informację zwrotną
- Skoryguj działanie tak, by kolejny efekt był bliższy sukcesowi

System powstaje od ogólnej wizji, aż do uzyskania końcowego produktu, dzięki kolejnym przybliżeniom. Każda kolejna wersja zbliża się funkcjonalnością do końcowego produktu, do ideału określonego w metaforze.

XP – główne wartości (1)

- Komunikacja

  Osoby tworzące oprogramowanie muszą identyfikować się z tworzonym projektem. W tym celu niezbędne jest zapewnienie dobrej komunikacji w zespole. Dodatkowo, bardzo ważne są umiejętności interpersonalne w trakcie współpracy z klientem czy podczas programowania w parach.

- Prostota

  Projekt powinien być tak prosty, jak to możliwe. Nie oznacza to stosowania banalnych rozwiązań, lecz stałe utrzymywanie jego przejrzystości i spójności.

- Szacunek

  Szacunek do pracy i czasu innych osób w zespole

XP – główne wartości (2)

- ## Informacja zwrotna

  Informacja zwrotna wraz z odpowiednią reakcją na nią są kluczowymi składowymi każdego przedsięwzięcia. Informację tę programiści uzyskują zarówno na poziomie ogólnym – od klienta, jak i szczegółowym – na podstawie wyników przypadków testowych TDD

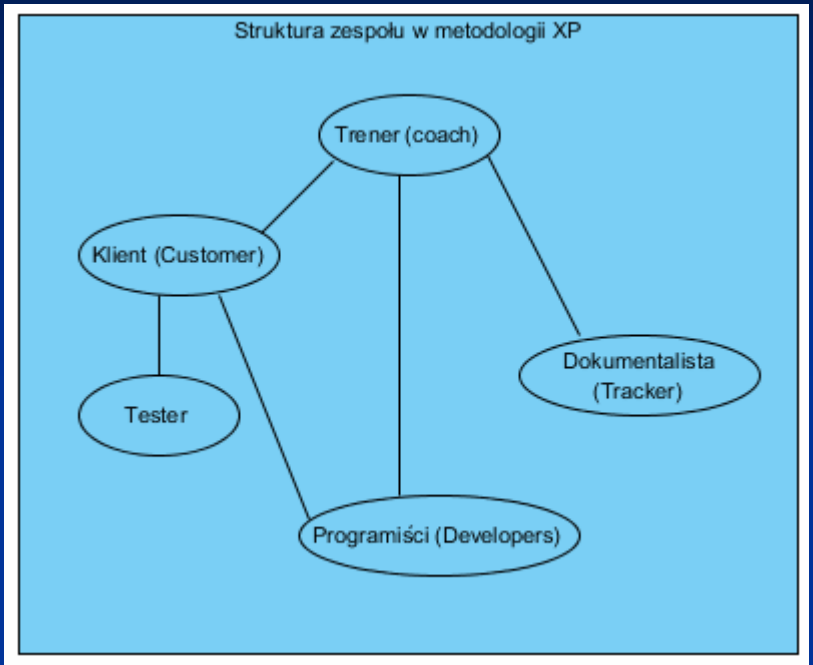
- ## Odwaga

  Programista powinien umieć, zachowując świadomość celu, podejmować ważne decyzje, np. w kwestiach takich, jak refaktoring, kluczowa zmiana architektury systemu, reorganizacja modelu. W związku z tym powinien charakteryzować się odwagą w podejmowaniu i wdrażaniu decyzji, jak również brać za nie pełną odpowiedzialność. Zgodnie z założeniem XP programista zawsze może liczyć na opinię klienta w kwestii istotnych decyzji.

Praktyki / cechy XP

- Określenie metafory tworzonego systemu
- Gra planistyczna
- Częste wydania
- Prosty projekt systemu
- Programowanie sterowane testami (TDD)
- Refaktoring
- Współwłasność kodu
- Ciągła integracja
- Udział klienta w zespole
- Obowiązujący standard kodowania
- Programowanie w parach

Struktura zespołu w XP



Główne role:

- Programiści
- Klient – uważany jest za członka zespołu, więc musi przez cały czas pracować razem z informatykami; czasem nie występuje w tej roli osobiście, lecz ma swojego przedstawiciela.

Role pomocnicze:

- Tester – pisze skrypty testowe na podstawie rozmów z klientem
- Coach – pomaga rozwiązywać napotkane problemy (technologiczne, organizacyjne itp.)
- Tracker – zbiera statystyki dotyczące wykonanych zadań, czasu pracy i tworzy podsumowania postępów projektu

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

A. Stworzenie metafory systemu

Jest to bardzo ogólna specyfikacja wymagań z punktu widzenia użytkownika i tworzona wraz z użytkownikiem.

Metafora projektu składa się zwykle z kilku zdań, które prezentują ogólny szkic tworzonego systemu. Zdania te powinny być zrozumiałe zarówno dla klienta jak i dla zespołu programistów. Dzięki temu wszyscy biorący udział w projekcie będą znali jego główny cel.

Metafora projektu Portalu Rekrutacyjnego:

Portal Rekrutacyjny obsługuje osoby szukające pracy, umożliwiając im wprowadzanie danych ze swojego CV. Ponadto system pozwala pracodawcom wyszukiwać kandydatów odpowiadających ich oczekiwaniom.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

B. Gra planistyczna, której efektem są user stories, czyli konkretne oczekiwania stawiane systemowi, spisywane na pojedynczych kartach, w języku użytkownika.

Etap ten pozwala określić konkretne wymagania klienta co do tworzonego systemu oraz zaplanować działania zespołu tak, aby w terminie osiągnąć zamierzone efekty.

Gra planistyczna pomaga:

- doprecyzować wizję projektu (metaforę)
- zidentyfikować wymagania klienta (user stories)
- określić czas potrzebny na spełnienie konkretnych wymagań
- nadać priorytety poszczególnym wymaganiom
- zaplanować pracę zespołu

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Gra planistyczna „rozgrywa się" pomiędzy klientem a zespołem programistycznym – jest to gra kooperatywna.

Sesja prowadzona jest przy stole, gdzie obok siebie siedzą klienci oraz członkowie zespołu programistycznego. Stół powinien być na tyle duży, aby łatwo można było na nim układać kartki zawierające spisane wymagania.

Wymagania klienta spisuje się na osobnych kartkach i numeruje. Istotne jest, aby user story określało pewien wycinek funkcjonalności systemu, np.:

1. *Osoba szukająca pracy rejestruje się samodzielnie w systemie rekrutacyjnym.*
2. *Osoba szukająca pracy wprowadza dane niezbędne w procesie rekrutacji (dane osobowe, informacje o wykształceniu i doświadczeniu zawodowym).*

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

User story

=

wymaganie użytkownika

Dilbert by Scott Adams

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

User stories należy pisać tak, aby były czytelne i zrozumiałe, unikając szczegółów i technicznego języka, którym na co dzień posługują się programiści.

Zamiast:

3. *Administrator odpowiedzialny za serwer rekrutacyjny generuje konta dla firm, które mają być zarejestrowane w systemie.*

jaśniej będzie:

3. *Administrator systemu tworzy konta dla firm.*

O ile jest to możliwe, należy pisać user stories w taki sposób, aby były niezależne od siebie (np. żeby nie wynikały z siebie, nie pokrywały się). Ułatwi to późniejsze przydzielenie zadań grupom programistycznym.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

User stories powinny być testowalne (weryfikowalne).

Dla przykładu:

4. *Pracodawca może łatwo wyszukiwać pożądane dane w systemie.*

nic nie mówi o tym, kiedy wyszukiwanie będzie łatwe.

Lepiej napisać konkretne wymagania:

4. *Pracodawca może wyszukiwać kandydata wg zadanych kryteriów: wykształcenia, zawodu, miejsca zamieszkania, doświadczenia, umiejętności.*

Czasami zdarza się, że user story jest zbyt duże. Należy je wtedy podzielić na mniejsze.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

Kolejnym etapem jest oszacowanie, ile czasu zajmie implementacja każdej user story. Informacja ta jest niezbędna do zaplanowania zadania dla grup programistów.

Czas szacuje się na podstawie wcześniejszych doświadczeń.

Można np. porównać daną user story z inną, która już kiedyś została zaimplementowana i na tej podstawie określić czas jej implementacji.

Szacować czas należy w możliwie najprostszy sposób.

Szacowanie zawsze będzie przybliżeniem – nie ma sensu stosować wyrafinowanych sposobów szacowania.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Ostatnim etapem jest nadanie priorytetów – należy ustalić, które user stories są najbardziej istotne dla klienta. Dzięki temu można tak zaplanować kolejne kroki, aby najbardziej wartościowe (z punktu widzenia klienta) wymagania były spełnione jako pierwsze.

Priorytety zapisuje się w postaci liczby. Często jest to skala pięciostopniowa, gdzie 1 odpowiada najwyższemu priorytetowi, a 5 najniższemu.

Reguła 80/20

Typowo, 80% wyników pochodzi z 20% wysiłku

Warto więc pokusić się o znalezienie tej części systemu, której wytworzenie (w 20% dostępnego czasu, przy wykorzystaniu 20% budżetu czy zasobów ludzkich itp.) doprowadzi do zbudowania większości (80%) systemu. Nie jest to oczywiście łatwe…

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

Przykładowe user stories dla Portalu Rekrutacyjnego:

1. [1] *Osoba Szukająca Pracy rejestruje się samodzielnie w systemie* [24 godz.]
2. [1] *Osoba Szukająca Pracy wprowadza dane niezbędne w procesie rekrutacji (dane osobowe, dane o wykształceniu i doświadczeniu zawodowym)* [32 godz.]
3. [2] *Administrator systemu tworzy konta dla firm* [16 godz.]
4. [3] *Firmy mają ograniczony czas dostępu do serwisu* [16 godz.]
5. [2] *Pracodawca może wyszukiwać kandydatów wg zadanych kryteriów (wykształcenie, zawód, miejsce zamieszkania, doświadczenie, umiejętności)* [32 godz.]
6. [3] *Wyniki wyszukiwania mogą być wyeksportowane (format: CSV)* [32 godz.]
7. [4] *Rejestracja Osoby Szukającej Pracy musi być zatwierdzona przez Administratora* [8 godz.]
8. [5] *Osoba Szukająca Pracy dostaje informację, że konkretna firma jest zainteresowana jej ofertą* [12 godz.]
9. [4] *Pracodawca po wybraniu najbardziej odpowiadających mu Osób Szukających Pracy i uiszczeniu opłaty, uzyskuje dostęp do ich danych osobowych* [20 godz.]

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

C. Wybór z user stories zestawu do pierwszej iteracji, której efektem będzie pierwsze przybliżenie końcowego produktu.

Zasady:

- Jeśli klient widzi, że coś działa, to znaczy, że działa, a jeśli nie widzi, to znaczy, że nie działa
- Klient nigdy nie będzie w stanie docenić kunsztu twórców programu
- Klient musi „doświadczyć" systemu i tylko wtedy będzie przekonany o celowości kontynuowania przedsięwzięcia

Z tych powodów proces tworzenia projektu dzieli się na części, zwane iteracjami. W czasie danej iteracji powstaje działający system wzbogacony o kolejne cechy. Nie budujemy od razu gotowej aplikacji, lecz dostarczamy ją iteracyjnie tak, aby klient naocznie mógł stwierdzić postępy w pracy (proces przyrostowy). Klient cały czas może obserwować postępy w pracach.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Planując iterację z klientem ustala się, które elementy systemu są dla niego najistotniejsze.

Wybiera się zatem user stories o najwyższym priorytecie, których sumaryczny czas wykonania nie przekroczy długości iteracji (długość iteracji powinna wynosić od 2 do 3 tygodni i być możliwie stała) – możliwie szybko należy dostarczyć system z pewną gotową funkcjonalnością.

Należy też uzgodnić z klientem warunki, które pozwolą stwierdzić, że dana funkcjonalność została zrealizowana. Można to zrobić w sposób formalny – używając języka pseudoalgorytmicznego, notacji matematycznej, tabeli warunków – lub w sposób mniej formalny – używając języka klienta. Skonkretyzowany zbiór warunków akceptacji danego wymagania nazywany jest testem akceptacyjnym.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

W przykładzie, do pierwszej iteracji wybieramy następujące user stories, sugerując się ich priorytetem i czasem wykonania:

1. [1] *Osoba Szukająca Pracy rejestruje się samodzielnie w systemie* [24 godz.]
2. [1] *Osoba Szukająca Pracy wprowadza dane niezbędne w procesie rekrutacji (dane osobowe, dane o wykształceniu i doświadczeniu zawodowym)* [32 godz.]
3. [2] *Administrator systemu tworzy konta dla firm* [16 godz.]
5. [2] *Pracodawca może wyszukiwać kandydatów wg zadanych kryteriów (wykształcenie, zawód, miejsce zamieszkania, doświadczenie, umiejętności)* [32 godz.]

Razem: 104 godziny = 13 dni

Dilbert by Scott Adams, tłum. własne

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

Planuje się tylko następną iteracje. Efektem każdej iteracji powinna być wersja programu spełniającą założenia dla danej iteracji.

Podejście iteracyjne daje potencjalnym użytkownikom czas na zapoznanie się i „oswojenie" z systemem, którego w przyszłości będą używać, a także pozwala na bieżąco uwzględniać ich sugestie oraz potrzeby. Implikuje to dużą elastyczność tworzonego projektu.

Do kolejnych iteracji brane są nie zrealizowane w pełni user stories oraz nowe user stories.

Może zdarzyć się sytuacja, że z niektórych istniejących user stories trzeba zrezygnować (po konsultacjach z klientem).

release early, release often –

wczesne i częste wydania

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

D. Rozdział zadań

Kiedy zakres iteracji został już określony, cały zespół zbiera się w celu wyznaczenia osób odpowiedzialnych za poszczególne prace.

Dla każdej user story wybierana jest para programistów, która będzie je realizować. W przypadku bardziej złożonych systemów wybieranych jest kilka par.

W ramach user story należy wyodrębnić konkretne zadania. O ile user story jest zapisem wymagań z punktu widzenia klienta, o tyle zadanie wyraża konkretne operacje, które wykonują uczestnicy projektu.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Przykład dla user story:

2. [1] *Osoba Szukająca Pracy wprowadza dane niezbędne w procesie rekrutacji (dane osobowe, dane o wykształceniu i doświadczeniu zawodowym)* [32 godz.]

Aby ją zrealizować, można wyróżnić następujące zadania:

- stworzenie modelu danych w postaci diagramu(-ów) UML;
- zaprototypowanie interfejsu użytkownika w postaci ołówkowych szkiców;
- implementacja operacji bazodanowych;
- implementacja logiki aplikacji;
- integracja pozostałych elementów user story.

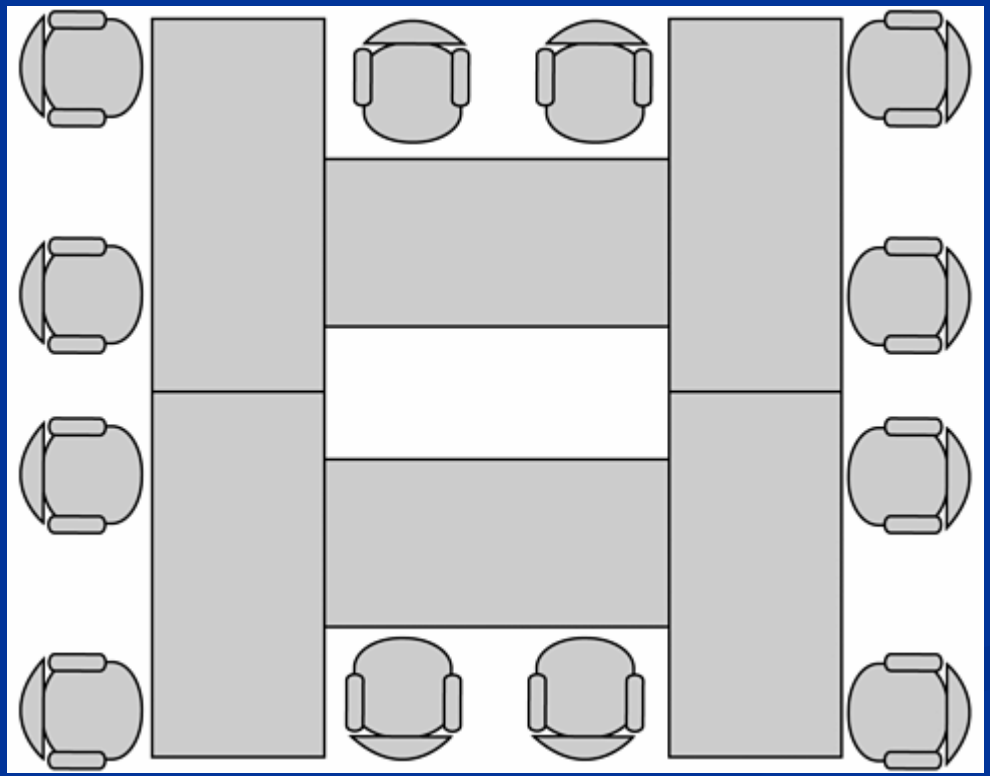
Zadania dobiera się tak, aby były od siebie jak najmniej zależne, a jeśli są zależne, ustala się kolejność, w jakiej muszą zostać zrealizowane. Należy też ustalić terminy zakończenia zadań, w zależności od oszacowanego uprzednio czasu implementacji danego user story.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

E. Implementacja.

Programiści tworzą kod parami. Gdy jeden z nich pisze kod, drugi spełnia rolę kontrolera (zgłasza poprawki, zadaje pytania wyjaśniające). Co kilkadziesiąt minut się zmieniają.



Skład par nie jest ustalany raz na zawsze.

Czasem celowo miesza się dobrego programistę ze słabym, czy też jedna osoba pisze test a druga kod, który ten test powinien spełniać itp..

Na rysunku propozycja układu biurek ;)

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Każdy programista może pracować nad dowolnym fragmentem kodu, nad dowolnym zadaniem – współwłasność kodu.

Współwłasność zapobiega przywiązywaniu się do „swoich" zadań, wiedza o projekcie zostaje rozpropagowana pośród członków zespołu, a sam projekt staje się niejako własnością wszystkich.

Wszyscy, poprzez odwołanie się do metafory, podzielają wspólną wizję projektu, będącą podstawą do tworzenia nazewnictwa i porozumiewania się.

Taki system pracy wymaga stosowania jednolitego standardu kodowania w zespole.

Dodatkowym elementem podtrzymującym integrację w zespole są spotkania (stand up meetings), podczas których zespoły prezentują swoje dokonania oraz najbliższe plany.

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

INŻYNIERIA OPROGRAMOWANIA

F. Programowanie sterowane testami (Test Driven Development, TDD)

Geneza: zanim zapytamy "jak zrobić?", najpierw należy odpowiedzieć na pytanie "co należy zrobić?"

Zanim klasa (lub inna podstawowa jednostka oprogramowania) zostanie zaimplementowana, najpierw należy napisać do niej zestaw przypadków testowych (test cases), które precyzują, jakiego zachowania oczekuje się po danej klasie.

Żaden kod nie może powstać, dopóki nie istnieje test – najpierw pisany jest test, a potem kod, który ten test spełnia

TDD będzie szczegółowo omówione na osobnym wykładzie

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

Zalety TDD:

- wysokie pokrycie kodu przez testy
- mniejszy stopień skomplikowania modułów (pisany jest tylko kod niezbędny do osiągnięcia wymaganej funkcjonalności)
- łatwe odnajdywanie błędów
- pełna automatyzacja testów
- wymuszanie na programistach dobrego projektu oraz odpowiedniej dekompozycji problemu

Etapy (kroki) w TDD:

1. Pisany jest test do nie istniejącego jeszcze kodu
2. Pisany jest najprostszy kod, który sprawi, że test zadziała
3. Refaktoring kodu (zmiana implementacji bez zmiany funkcjonalności) – upewniamy się regularnie, czy test nadal działa

Omówienie praktyk XP na przykładzie Portalu Rekrutacyjnego

G. Refaktoring

Stosuje się go, jeśli klasy nie można w prosty sposób przetestować oraz wówczas, gdy staje się zbyt skomplikowana i nieczytelna. Funkcjonalność się jednak nie zmienia!

H. Dokumentacja

Kod, przypadki testowe, komentarze są częścią dokumentacji projektu. Zwykle nie wymaga się formalnej, skodyfikowanej dokumentacji, a diagramy nie są wytycznymi projektu i pełnią funkcję pomocniczą.

I. Ciągła integracja

Każda zaimplementowana funkcjonalność jest natychmiast integrowana z istniejącym już kodem. W ten przyrostowy sposób kończona jest iteracja i bieżąca wersja przekazywana jest klientowi. Jeśli niektóre z *user stories* nie zostały zaimplementowane, przechodzą do kolejnej lub następnej iteracji.

XP a inne zwinne metodyki

Najczęściej stosowane zwinne metodyki różnią się jedynie szczegółami

Np. stand up meeting (XP) oraz scrum meeting (SCRUM) to różne nazwy codziennych spotkań grupy; user story (XP) to product backlog (SCRUM) itp.

Drobne, przykładowe różnice

- Metafora (XP) – Model (FDD)
- Współwłasność kodu (XP) – Współwłasność klasy (FDD)
- Programowanie w parach (XP) – Inspekcje kodu (FDD)
- Brak formalnego zarządzania (XP) – Oszczędne zarządzanie, raporty itp., Główny Programista (FDD) – „ScrumMaster” (SCRUM)
- Programowanie sterowane testami (XP) – testowanie do uznania Głównego Programisty (FDD)
- Krótkie iteracje (FDD, XP) – nieco dłuższe „sprinty” (SCRUM)

Problemy i kontrowersje

- Brak dokładnej specyfikacji…
- Brak szczegółowego projektu…

  … a w związku z tym trudności ze zrozumieniem projektu, np. przy rotacjach kadrowych
- Krótka perspektywa planowania
- Słaba przewidywalność kosztów wytworzenia systemu
- Konieczna stała dostępność przedstawiciela klienta
- Trudności z wyrażeniem wymagań niefunkcjonalnych
- Nie przynosi korzyści przy średnich lub słabych programistach; dobrzy programiści z kolei poradzą sobie w dowolnych warunkach
- Trudności z ponownym użyciem kodu
- Trudności z konserwacją przez inny zespół
- Współwłasność kodu (XP) – każdy może zmieniać dowolny fragment systemu
- Nie ma dokładnych i ogólnych wyliczeń, jak łączna wydajność pracy zmienia się przy przejściu od tradycyjnego programowania indywidualnego do programowaniu parami (XP)

Dilbert by Scott Adams

XPrince – równowaga między zwinnością a dyscypliną w projektach informatycznych

INŻYNIERIA OPROGRAMOWANIA

XPrince – eXtreme PRogramming IN Controlled Environments (2004 r., Politechnika Poznańska), http://xprince.net/

Jest połączeniem metodyk tradycyjnych (PRINCE2, RUP) z metodyką zwinną (XP)

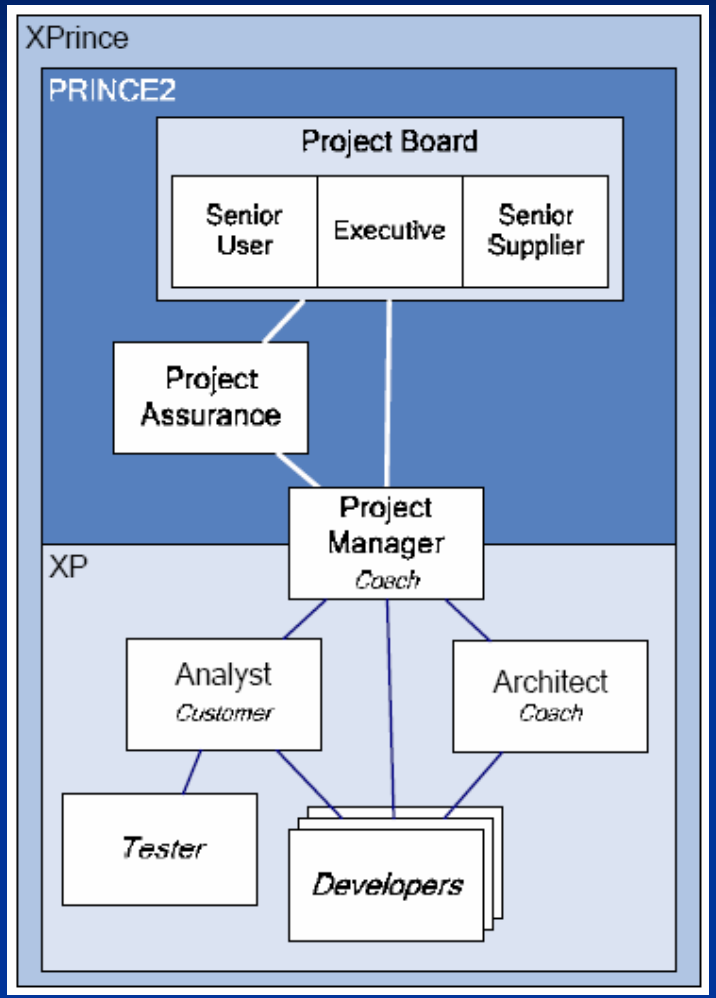
W skrócie:

PRINCE2 (Projects IN Controlled Environments) – restrykcyjne, procesowe podejście do zarządzania projektami; podział na etapy zarządcze i realizacyjne; Kierownik Projektu i Komitet Sterujący; raporty itp.

RUP (Rational Unified Process) – iteracyjny, ale klasyczny proces tworzenia oprogramowania (z pełną dokumentacją, analizą zagrożeń itp.)

XPrince – organizacja zespołu

INŻYNIERIA OPROGRAMOWANIA

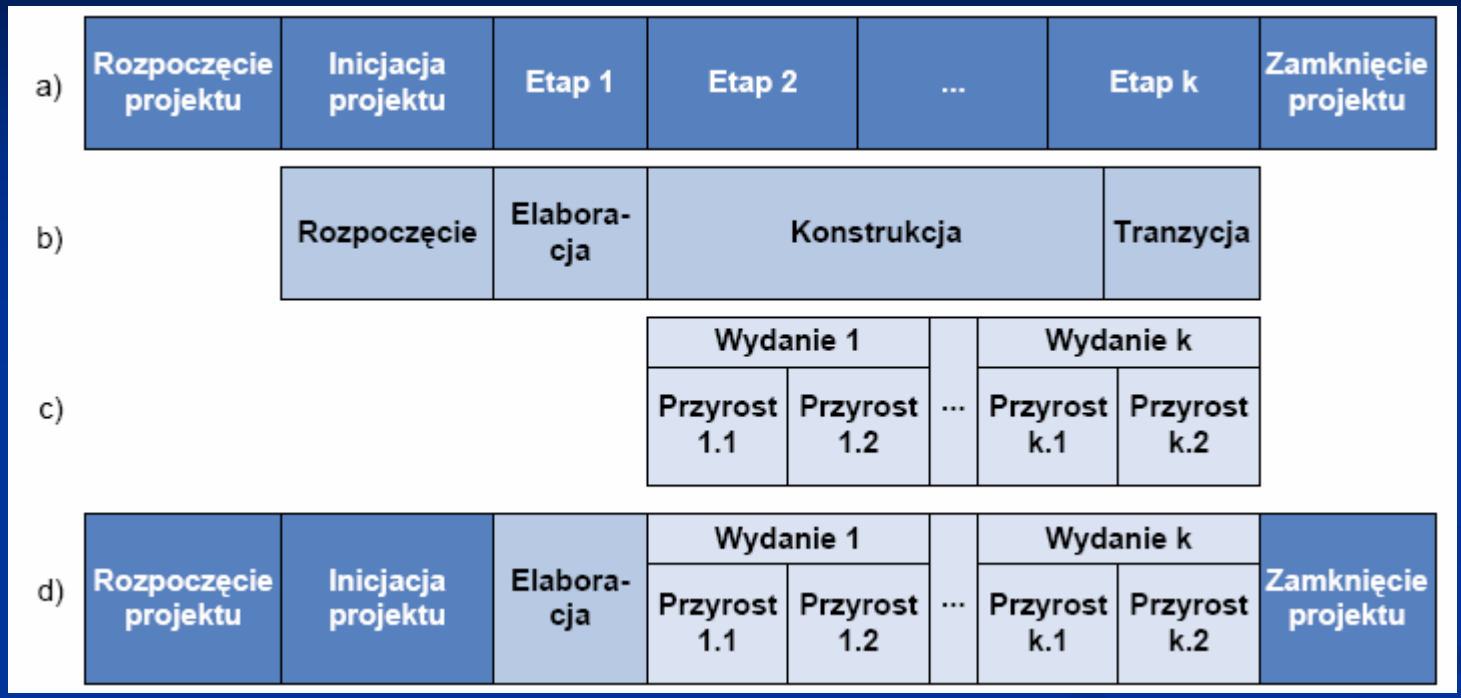


Komitet Sterujący (Project Board):

- Dyrektor (Executive) – przedstawiciel inwestora
- Główny Użytkownik (Senior User) – kieruje użytkownikami końcowymi, skupia się na użyteczności
- Główny Dostawca (Senior Supplier) – reprezentuje wytwórcę

- Menadżer Projektu (Project Manager) – odpowiada za taktyczny poziom zarządzania
- Audytor Projektu (Project Assurance) – kontrola działań Menedżera Projektu
- Analityk (Analyst) – odpowiada za specyfikację
- Architekt (Architect) – odpowiada za techniczne aspekty projektu
- Programiści (Developers)

XPrince – cykl życia systemu



a) PRINCE2
b) RUP
c) XP
d) XPrince

XPrince – opis poszczególnych faz (1)

- Rozpoczęcie projektu
  - Ustanowienie zespołu zarządzania projektem
  - Stworzenie wizji systemu, zawierającej wstępne argumenty biznesowe
- Inicjacja projektu
  - Zrozumienie, co należy zbudować. Budowa modelu biznesowego bazującego na przypadkach użycia, zdefiniowanie problemów, które należy rozwiązać oraz najważniejszej funkcjonalności wymaganej do rozwiązania tych problemów. Opracowanie listy kryteriów jakości i produktów. Ocena ryzyka. (Analityk)
  - Propozycja początkowej architektury (krótki, wysokopoziomowy opis, dostarczający informacji potrzebnej do zaplanowania projektu; lista potrzebnych narzędzi). (Architekt)
  - Planowanie całego projektu i dopracowanie uzasadnienia biznesowego. Plan projektu pokazuje projekt ze strategicznego punktu widzenia. W celu wspierania „zwinności", plan powinien uwzględniać priorytety, precyzować liczbę wydań i przydzielać do nich funkcjonalność (przypadki użycia na wysokim poziomie). Im dłuższy projekt, tym plan projektu powinien być mniej konkretny. Faktyczne planowanie powinno się odbywać na poziomie wydań. (Menadżer Projektu)

XPrince – opis poszczególnych faz (2)

- Inicjacja projektu (c.d.)
  - Ustalenie kanałów komunikacyjnych i środowiska zarządzania projektem. Kanały komunikacyjne obejmują raporty (np. wyniki cotygodniowych testów akceptacyjnych). Środowisko zarządzania projektem może być klasyczne, bazujące na plikach i dokumentach, lub może być wspomagane zaawansowanymi narzędziami. (Menadżer Projektu).

- Elaboracja
  - Faza Elaboracji dotyczy głównie architektury. Architekt powinien zaproponować mechanizmy architektoniczne, rozpoznać ryzyko z tym związane (np. za pomocą eksperymentów) oraz stworzyć szkielet, który będzie wykorzystywany przez Programistów. Analityk i Menadżer Projektu w tej fazie udoskonalają wymagania i plan projektu.

XPrince – opis poszczególnych faz (3)

- Wydanie
  - Na tym etapie proces wytwarzania oprogramowania bardzo przypomina XP.
  - Architekt i Programiści produkują kod i przypadki testowe.
  - Każde wydanie składa się z kilku przyrostów, przyrost jest jedynie wewnętrznym punktem kontrolnym. Każdy przyrost powinien być tak samo długi – to pomaga Programistom czuć rytm iteracji oraz w rezultacie nauczyć się lepiej planować przyrosty.
  - Analityk jest odpowiedzialny za wymagania i testy akceptacyjne, jak również gra rolę klienta będącego na miejscu.
  - Każde Wydanie jest zakończone fazą tranzycji, w której nowa wersja systemu jest wdrażana i przekazywana użytkownikom końcowym.
- Zamknięcie projektu
  - Projekt jest zamykany, identyfikowane są dalsze akcje i następuje ocena projektu.

XPrince – podsumowanie (1)

- Jest zwinna

  XPrince przyjmuje podstawowe założenie metodyki XP – jest nastawiona na jak najszybsze stworzenie działającego produktu, etapy są w niej krótkie, a zarządzanie zmianami praktykowane przez cały czas trwania projektu. Dzięki temu klient otrzymuje szybko kolejne wersje produktu i ma stałą kontrolę jego zakresu.

- Posiada mechanizmy kontroli

  XPrince kontroluje projekt na różnych poziomach. Kontrolowane są zmiany, kontrolowane jest ryzyko, kontrolowana jest jakość produktu a także jakość pracy kierownika projektu. Zarząd przedsięwzięcia jest odseparowany od codziennej pracy w projekcie i kontroluje go na podstawie informacji od kierownika projektu. Rzetelność informacji przekazywanych przed kierownika projektu jest weryfikowana przez kontrolę projektu.

- Zachowuje optymalny poziom dokumentacji technicznej

  XPrince zakłada dokumentowanie wymagań w postaci przypadków użycia systemu oraz innych diagramów UML. Są to metody sprawdzone i popularne, dzięki czemu skraca się czas wdrożenia metodyki.

XPrince – podsumowanie (2)

- Ma prostą i efektywną strukturę organizacyjną

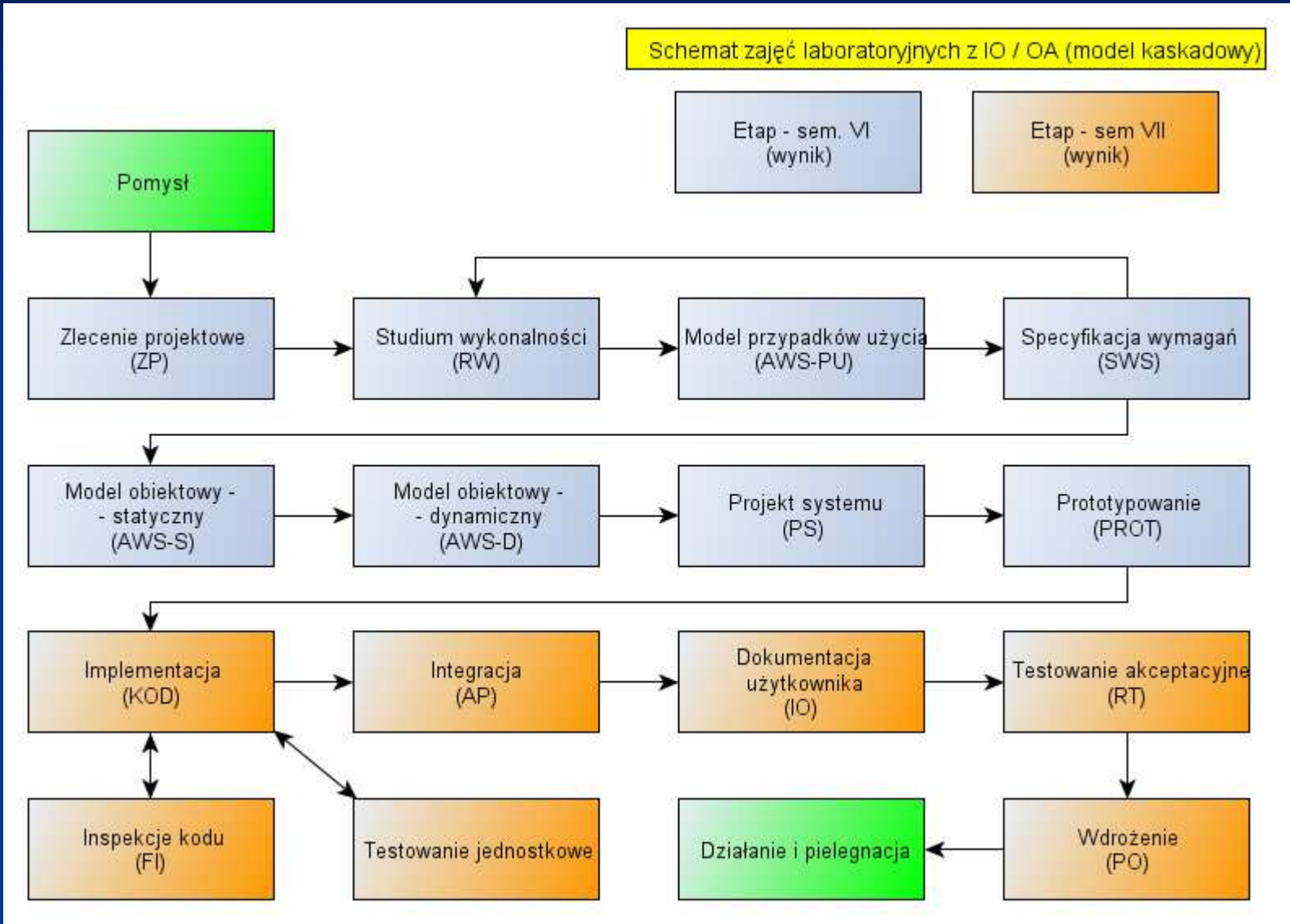
  XPrince wprowadza przejrzysty podział ról w procesie. Hierarchia w jest zminimalizowana, a odpowiedzialność za elementy procesu praktycznie podzielona. Wprowadzone są np. role głównego analityka, odpowiadającego za biznesowe czynniki ryzyka i architekta, odpowiadającego za ryzyko technologiczne (obie role zaczerpnięto z metodyki RUP). Ich zwierzchnikiem jest kierownik projektu, który zarządza ryzykiem całego projektu, kierując się sygnałami od obu specjalistów.

- Wykorzystuje zwinne praktyki programistyczne

  XPrince zaczerpnęło z XP zestaw dobrych praktyk programistycznych. Zarządzanie wersjami, ciągła integracja, testy jednostkowe, testy akceptacyjne, implementacja kierowana testami (ang. TDD – Test Driven Development) to praktyki zapewniające wysoką jakość produktu. Z XP przejęte zostało także opracowanie rozwiązań próbnych (ang. spike solution), które w XPrince stosuje się na początku projektu, w fazie opracowania architektury. Dopuszcza programowanie indywidualne, w parach jak również „side by side”.

Schemat zajęć laboratoryjnych z IO / OA (model kaskadowy)

INŻYNIERIA OPROGRAMOWANIA



Schemat zajęć laboratoryjnych z IO / OA (model przyrostowy)

INŻYNIERIA OPROGRAMOWANIA