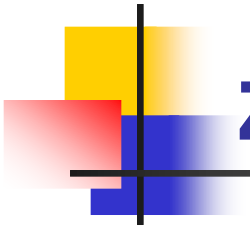
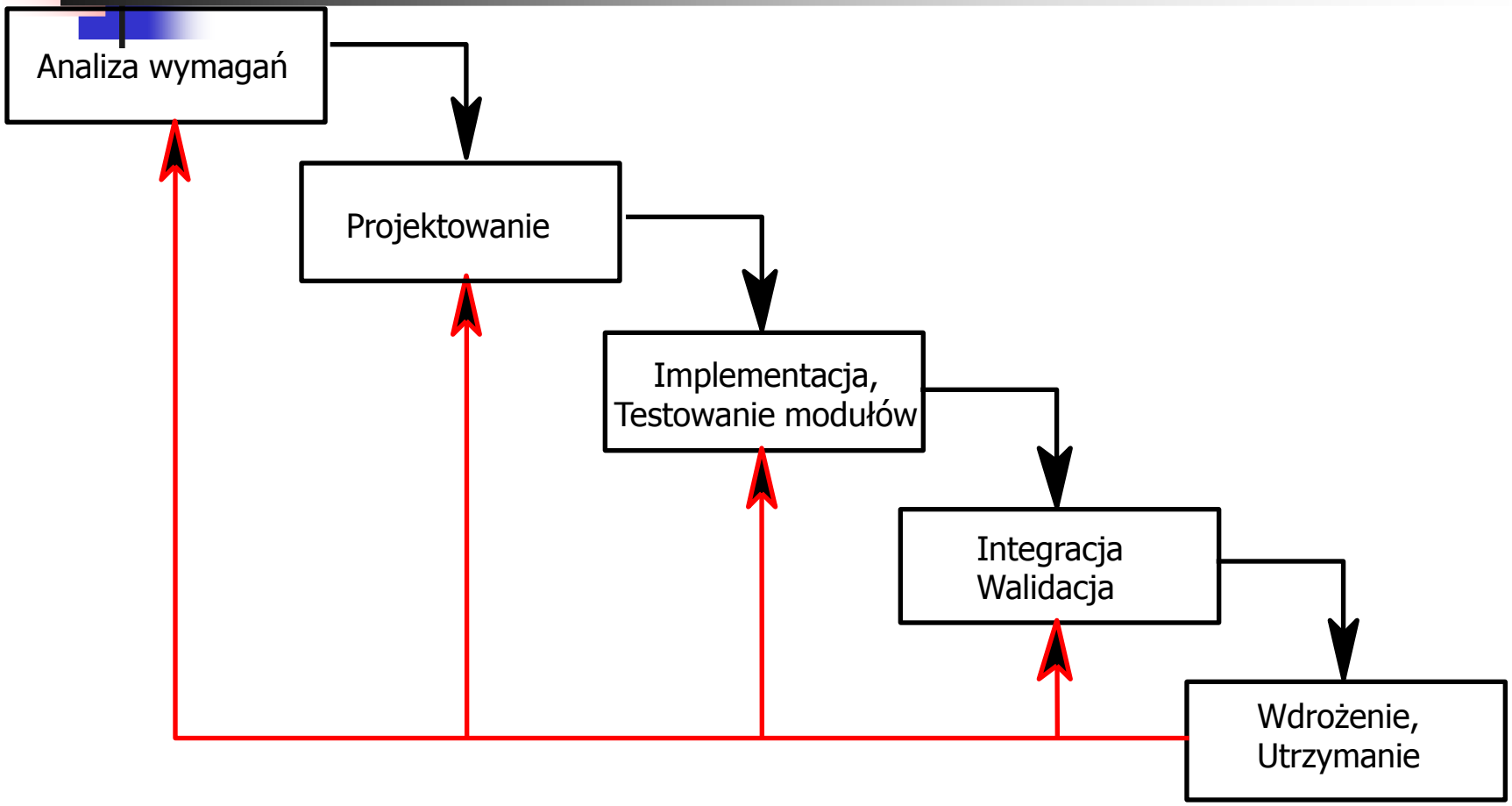


Zagadnienia

- Co to jest eXtreme Programming (XP)
- Czym charakteryzują się tzw. lekkie metodyki zarządzania procesem produkcji oprogramowania
- Reguły i praktyki XP
- Dlaczego i kiedy można a w jakich przypadkach nie powinno się stosować XP

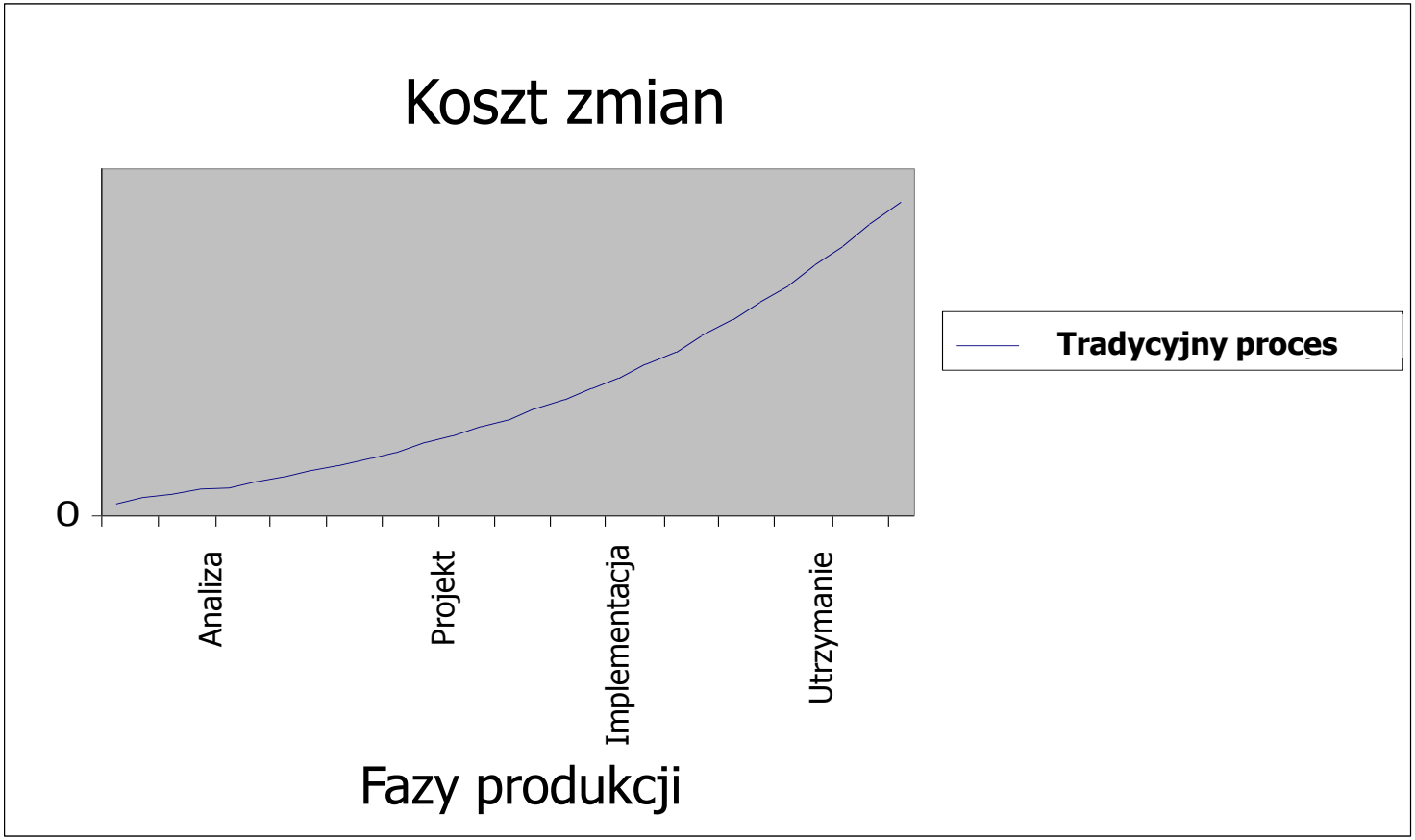
Tradycyjny proces produkcji oprogramowania



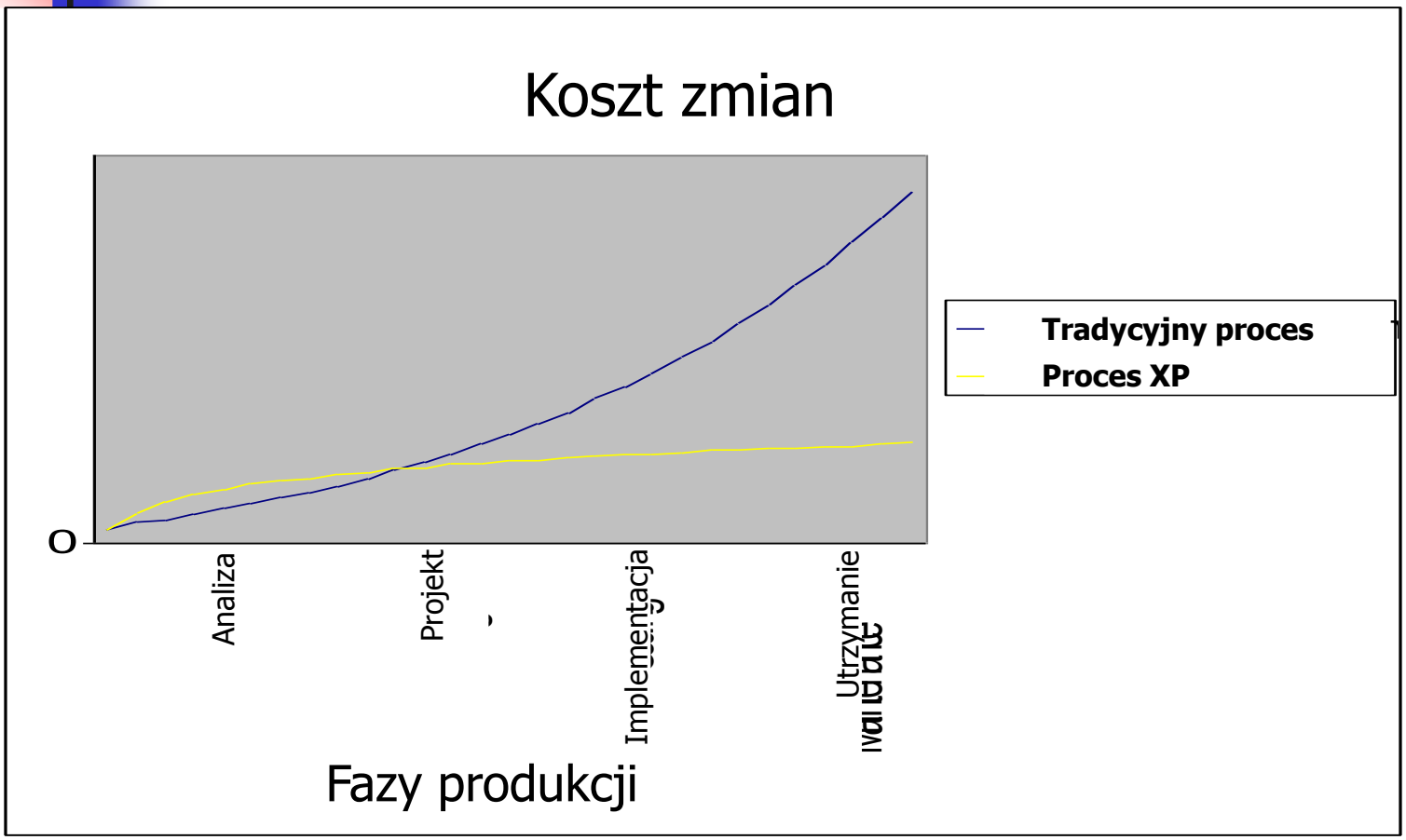
Tradycyjny proces produkcji oprogramowania

- Składa się z następujących po sobie faz, przy czym rozpoczęcie kolejnej fazy następuje dopiero po całkowitym zakończeniu poprzedniej fazy
- Weryfikacja i walidacja oprogramowania realizowana jest w końcowych fazach procesu
- Duży koszt i długi czas potrzebny na stworzenie „kompletnej" specyfikacji wymagań która nie podlega zmianom w trakcie produkcji oprogramowania
- Klienci i użytkownicy systemu biorą udział tylko w początkowych i końcowych fazach produkcji oprogramowania
- Bardzo duży nacisk kładziony na produkcję, w wielu przypadkach nieprzydatnej dokumentacji poszczególnych faz produkcji (raporty, diagramy, formularze itp.)
- Brak zarządzania ryzykiem i zmianami wymagań

Koszt zmian w tradycyjnych procesach produkcji



Koszt zmian w procesie XP



Czym jest eXtreme Programming (1/2)

- Relatywnie nowa tzw. lekka metodyka (ang. agile) opisująca proces tworzenia oprogramowania
- Powstała w wyniku obserwacji tradycyjnych procesów poprzez eliminację elementów spowalniających tworzenie oprogramowania i położenie szczególnego nacisku na elementy zwiększające wydajność, efektywność i jakość oprogramowania
- *„skuteczny, bezpieczny, elastyczny, naukowy i przyjemny sposób programowania”* (Kent Beck)

Czym jest eXtreme Programming (2/2)

- Proces zgodny w wymaganiami XP:
  1. Opiera się na ścisłej komunikacji i dobrych „przyzwyczajeniach” programistów i grup projektowych
  2. Ukierunkowany na ludzi
  3. Potrafi zarządzać zmianami wymagań
  4. Składa się z krótkich iteracji
  5. Umożliwia natychmiastowe przekazywanie informacji zwrotnych
  6. Potrafi zarządzać ryzykiem
  7. Wspiera koncepcje obiektowe
  8. ....

Czym XP rożni się od innych metodyk?

- Wczesne, konkretne i ciągłe informacje zwrotne w krótkich cyklach programowania
- Planowanie przyrostowe
- Zdolność do elastycznego projektowania i implementacji funkcjonalności w zależności o dynamicznie zmieniających się wymagań
- Opieranie się na automatycznych testach napisanych wspólnie z Klientem
- Opieranie się na komunikacji słownej i kodzie źródłowym oprogramowania w celu przekazania intencji programistów i Klienta
- Zapewnienie bliskiej współpracy programistów
- Promowanie prostoty rozwiązań i iteracyjnego procesu poprawy zaimplementowanych rozwiązań

Cztery wartości XP

- **Komunikacja**
  - Ścisła, bezpośrednia współpraca pomiędzy programistami i Klientem
- **Prostota**
  - Poszukiwanie najprostszego rozwiązania spełniającego wyznaczone założenia
- **Informacja zwrotna**
  - Szybkie iteracje i bezpośredni kontakt umożliwiają uzyskiwanie niemalże natychmiastowych informacji na temat stanu projektu i aktualnie istniejących zagrożeń
- **Odwaga**
  - Szybkie podejmowanie decyzji w oparciu o zebrane informacje, agresywne i zachęcające podejście do zmian mogących wpłynąć na jakość, termin i koszt oprogramowania wspierane przez pozostałe trzy wartości XP

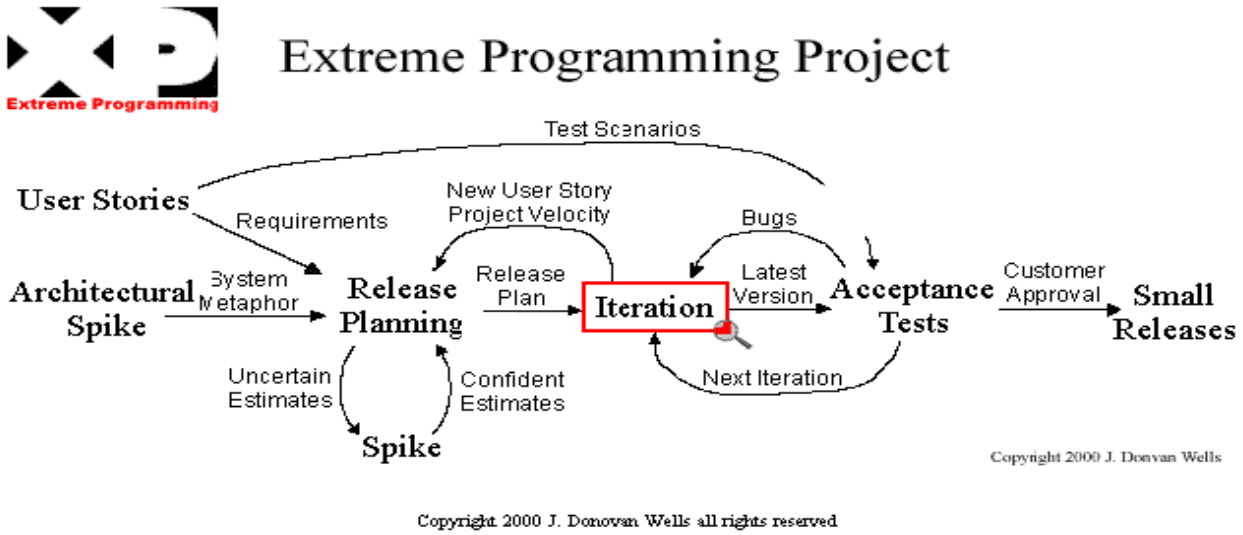
Prawa i obowiązki programisty płynące z czterech wartości XP

- Prawo do informacji co należy zrobić, dzięki prostej, czytelnej specyfikacji wymagań z jasno określonymi priorytetami
- Obowiązek określenia ile realizacja danego zadania zajmie czasu i prawo do zmiany estymacji
- Prawo akceptacji odpowiedzialności za zadanie w miejsce obowiązku akceptacji przydzielonego odgórnie zadania i dotyczącej go estymacji
- Obowiązek tworzenia wysokiej jakości oprogramowania spełniającego wyznaczone wymagania (ale nic wykraczającego poza wymagania!)
- Prawo do spokojnej, **dającej poczucie dobrej zabawy** i spełnienia pracy

Prawa i obowiązki Klienta płynące z czterech wartości XP

- Obowiązek stworzenia wspólnie z programistami jasnej czytelnej specyfikacji wymagań
- Prawo do informacji jaki jest koszt, czas implementacji poszczególnych funkcjonalności i jaki nakład pracy potrzebny jest do osiągnięcia poszczególnych celów
- Prawo do obserwacji postępu prac poprzez ewaluację dostarczanych w kolejnych iteracjach **działających** części systemu
- Prawo do potwierdzenia jakości systemu poprzez wykonanie serii testów
- Prawo do zmiany zdania, podmiany wymagań i zmiany priorytetów
- Prawo do informacji o zmianach planu, ewentualnych opóźnieniach, zmianach estymacji tak aby możliwa była zmiana zakresu w celu uzyskania interesującej funkcjonalności w założonym czasie i budżecie

Model procesu produkcji oprogramowania według XP

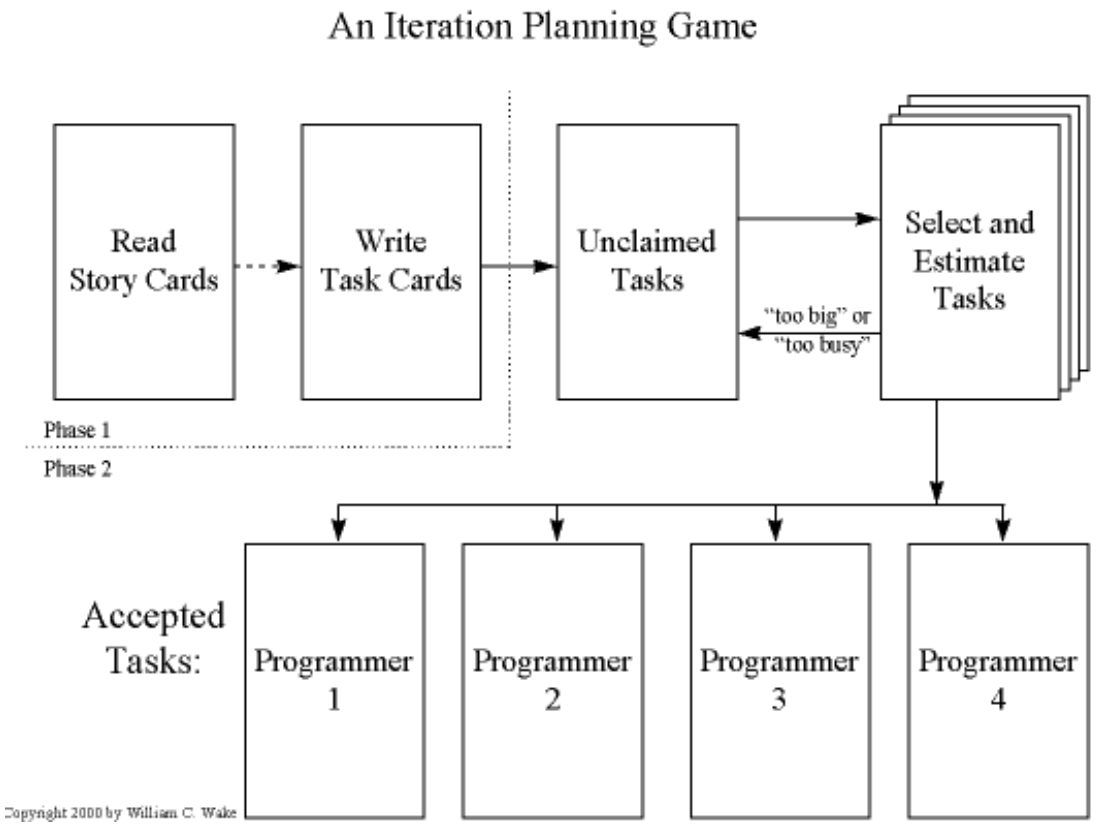


http://www.extremeprogramming.org/map/project.html

Praktyki XP

- Praktyki wspierające wartości XP podzielone zostały na następujące obszary:
  1. Planowanie (planning)
  2. Projektowanie (designing)
  3. Kodowanie (coding)
  4. Testowanie (testing)
- W odróżnieniu od innych metodyk w XP nie ma wyraźnego podziału na fazy, w których realizowane są zadania z poszczególnych obszarów tzn. **planujemy** przez **cały czas**, **projektujemy** przez **cały czas**, **implementujemy** przez **cały czas** i **testujemy** przez **cały czas**

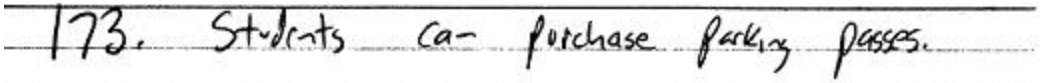
Praktyki XP związane z planowaniem



Praktyki XP związane z planowaniem

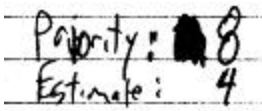
- Spisane scenariusze (user stories)
  1. Podobne do usecasów ale krótsze i pisane językiem zrozumiałym dla Klienta, stanowią formalny dokument wymagań
  2. Stosowane w celu określenia wstępnych estymacji
  3. Stanowią podstawę dla planowania i podziału zadań
  4. Stanowią podstawę dla testów akceptacyjnych

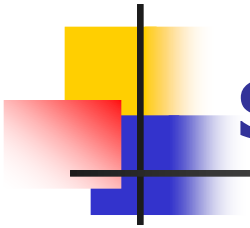
Scenariusze



Jako pewna rola (role) chcę zrealizować pewne zadanie (something) żeby osiągnąć cel (benefit).

"As a Student I want to purchase a parking pass so that I can drive to school".





Scenariusze

- Studentci mogą kupować bilety miesięczne online.
- Bilety miesięczne mogą być opłacane za pomocą karty kredytowej
- Bilety miesięczne mogą być opłacane z wykorzystaniem PayPal ™.
- Profesor może wpisywać oceny do systemu SUSZI online.

Praktyki XP związane z planowaniem

- Plan dostaw tworzy harmonogram projektu
  1. Cały zespół bierze czynny udział w estymacjach zadań (scenariusze dłuższe niż 3 tygodnie powinny zostać podzielone na mniejsze)
  2. Plan iteracji powstaje po wykonaniu estymacji
  3. Podejmowane są decyzje, które scenariusze zostaną zaimplementowane
  4. Poszczególne iteracje nie są planowane w szczegółach do momentu ich rozpoczęcia
  5. Planowanie opiera się na 4 zmiennych: zasobach, czasie, jakości i zakresie
     - W trakcie planowanie głównie zmianom podlega zakres pozostałe zmienne określone są przeważnie przez Klienta

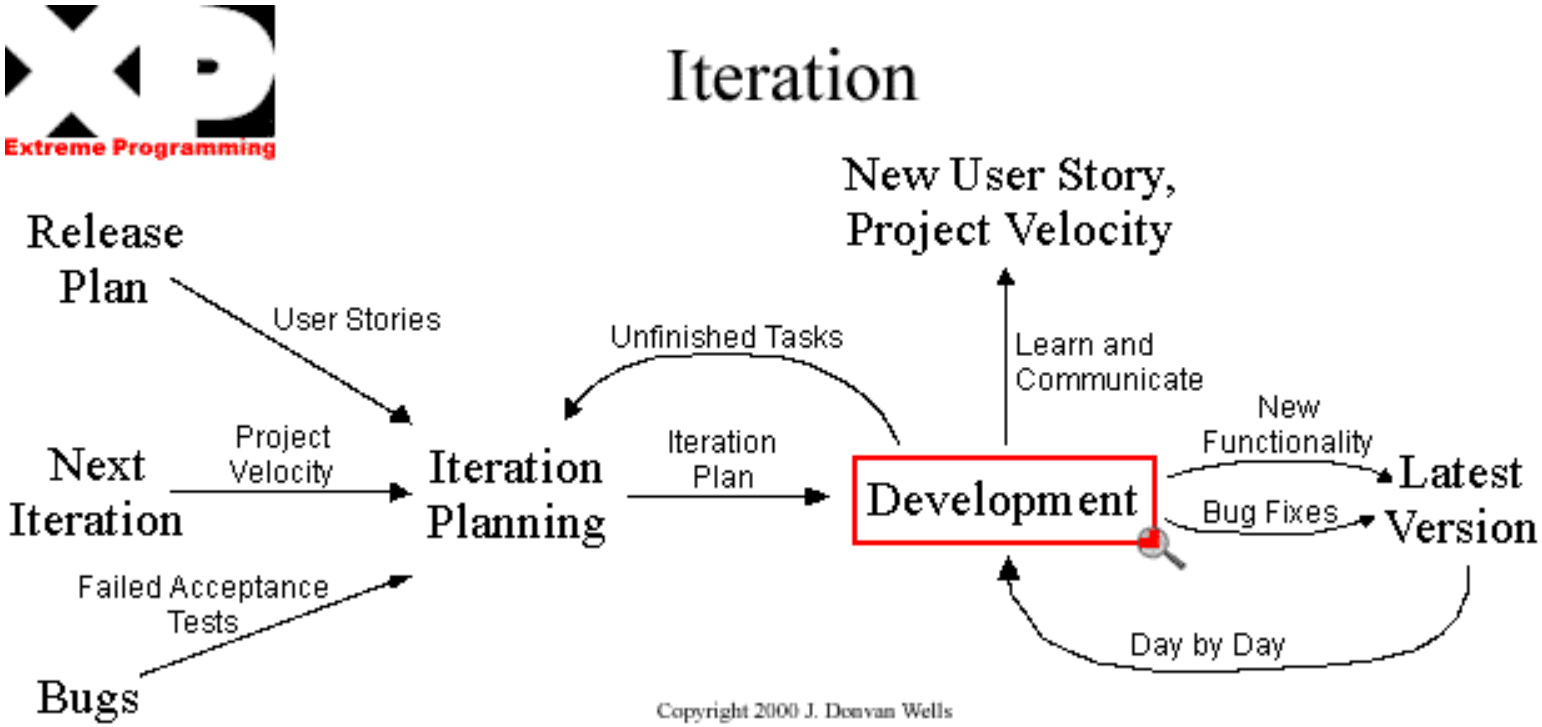
Praktyki XP związane z planowaniem

- Szybkie iteracje
  1. Każda iteracja rozpoczyna się spotkaniem, na którym uzgadniany jest szczegółowy plan iteracji
  2. Każda iteracja trwa od 1-4 tygodni
  3. Każda iteracja składa się z planowania, projektowania, kodowania i testów akceptacyjnych dla scenariuszy implementowanych w danej iteracji
  4. Każdy scenariusz dzielony jest na zadania dla poszczególnych programistów i grup
  5. Zadania polegają na implementacji funkcjonalności nowego scenariusza lub naprawy błędów zgłoszonych w testach poprzedniej iteracji

Praktyki XP związane z planowaniem

- Planowanie iteracje
  1. Estymacja wyznaczonych zadań
  2. Usunięcie zadań jeżeli termin ukończenia iteracji może być zagrożony
  3. Przypisanie priorytetów do zadań
     1. Klient ze względu na ważność zadań
     2. Programiści ze względu na złożoność implementacji
  4. Harmonogram realizacji zadań (spotkania kontrolujące stan realizacji zadań tzw. „stand-up meeting” każdego dnia przed rozpoczęciem pracy)
  5. Przydzielanie ludzi do różnych zadań i do pracy w różnych grupach

Planowanie XP



Praktyki XP związane z projektowaniem

- Stosowanie rozwiązania najprostszego z możliwych

  „Always do the simplest thing that could possibly work”

  - Nigdy nie projektuj funkcjonalności, która może nigdy nie zostać wykorzystana
  - Jeżeli projektowane rozwiązanie okaże się zbyt uproszczone może zostać zmienione w trakcie najbliższego refactoringu

Praktyki XP związane z projektowaniem

- Minimalizacja ryzyka poprzez budowę prototypów (spike solutions)
  1. Prototypowanie dla zagadnień stanowiących potencjalne ryzyko wystąpienia problemów z implementacją
  2. Prototyp powinien tylko obejmować zagadnienia związane z ryzykiem i nic poza tym
  3. Prototypowanie powinno się zakończyć w momencie uzyskania odpowiedzi, że dany problem można rozwiązać
  4. Nigdy **nie wykorzystywać** kodu prototypu w produkcji! (*„Build one to throw away”*)

Praktyki XP związane z projektowaniem

- Refaktoring
  1. Upraszczanie kodu
  2. Zmiana istniejącej implementacji w celu umożliwienia rozbudowy o nową funkcjonalność
  3. Dostosowanie kodu do przyjętych standardów
  4. Sesje refaktoringowe powinny być przeprowadzane regularnie w trakcie trwania projektu
  5. Korzystanie z narzędzi wspomagających wprowadzanie jednoczesnych zmian w całym kodzie (np. zmian sygnatur metod w deklaracji klasy i klasach wykorzystujących te metody)
  6. Nigdy nie należy refaktorować kodu, dla którego nie istnieją testy

Praktyki XP związane z kodowaniem

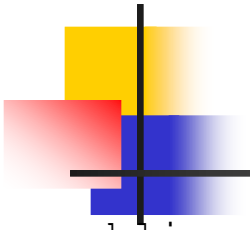
- Wykorzystanie standardów kodowania
  1. Każdy projekt XP powinien korzystać z istniejących lub uzgodnić własne standardy implementacji (wpływa to na jakość i czas oraz wspiera pozostałe praktyki XP)
  2. Wykorzystanie narzędzi, wzorców do generacji jak największej ilości sprawdzonego kodu

Praktyki XP związane z kodowaniem

- Klient dostępny przez cały projekt, w idealnym przypadku pracuje w tym samym pomieszczeniu co zespół projektowy (ang. on-site customer) i jest w stanie odpowiedzieć na dowolne pytanie związane z wymaganiami i założeniami dotyczącymi budowanego systemu
- Programowanie w parach (ang. pair programming)

Praktyki XP związane z kodowaniem i testowaniem

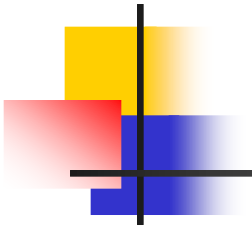
- Planowanie i implementacja tzw. unit testów (tzn. testów na poziomie klas/pakietów/modułów) **przed** implementacją funkcjonalności
- Przystępując do kodowania kolejnej funkcjonalności (user story) najpierw kodowany jest **test**, później powstaje **najprostsza** z możliwych implementacja a następnie rozważana jest potrzeba ewentualnego **refaktoringu**
- W przypadku implementacji związanej z poprawą błędu najpierw kodowany jest **test** pozwalający powtórzyć błąd, później powstaje **najprostsza** z możliwych implementacja poprawki, która przechodzi **wszystkie testy** a następnie rozważana jest potrzeba ewentualnego **refaktoringu**
- Automatyzacja testów, uruchamianie testów po każdej integracji kodu (testy uruchamiane kilkukrotnie w ciągu dnia)
- Istnienie testów ułatwia zarządzanie i wprowadzanie zmian



```
public class Person {
   String name;
   int favorite = -1;
   public Person (String name, int favorite) {
           this.name = name;
           this.favorite = favorite;
   }
}
```

Zadanie:

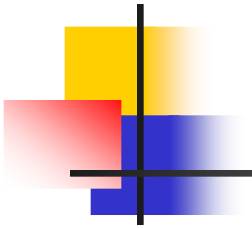
- Zaimplementować serilizację obiektu do postaci XML:
  <name>the-name</name>
  (jeżeli favorite mniejsze 0), lub
  <name favorite="nn">the-name</name>
  jeżeli favorite równe 0 lub większe od 0.



- Budujemy test:

```
Person p = new Person("Pop", -1);
assertEquals("<name>Pop</name>", p.asXml());
```

- W trakcie kompilacji wystąpi błąd składni ponieważ metoda `asXml()` nie istnieje.
- Implementujemy zaślepkę:
  `public String asXml() {return null;}`
- Kompilujemy i uruchamiamy; Test oczywiście nie przechodzi !
- Implementujemy metodę w najprostszej możliwe postaci:
  `public String asXml() {return "<name>" + name + "</name>";}`
- Kompilujemy i uruchamiamy, test przechodzi ! – pełen cykl XP.



- Możemy dodać nowy test, i przetestować inną sytuację:

```
Person p2 = new Person("Mom", 1);
assertEquals("<name favorite=\"1\">Mom</name>", p2.asXml());
```

- Test nie przechodzi bo metoda `asXml()` nie ma pełnej funkcjonalności.
- Implementujemy dodatkowy kod i ponownie kompilujemy.

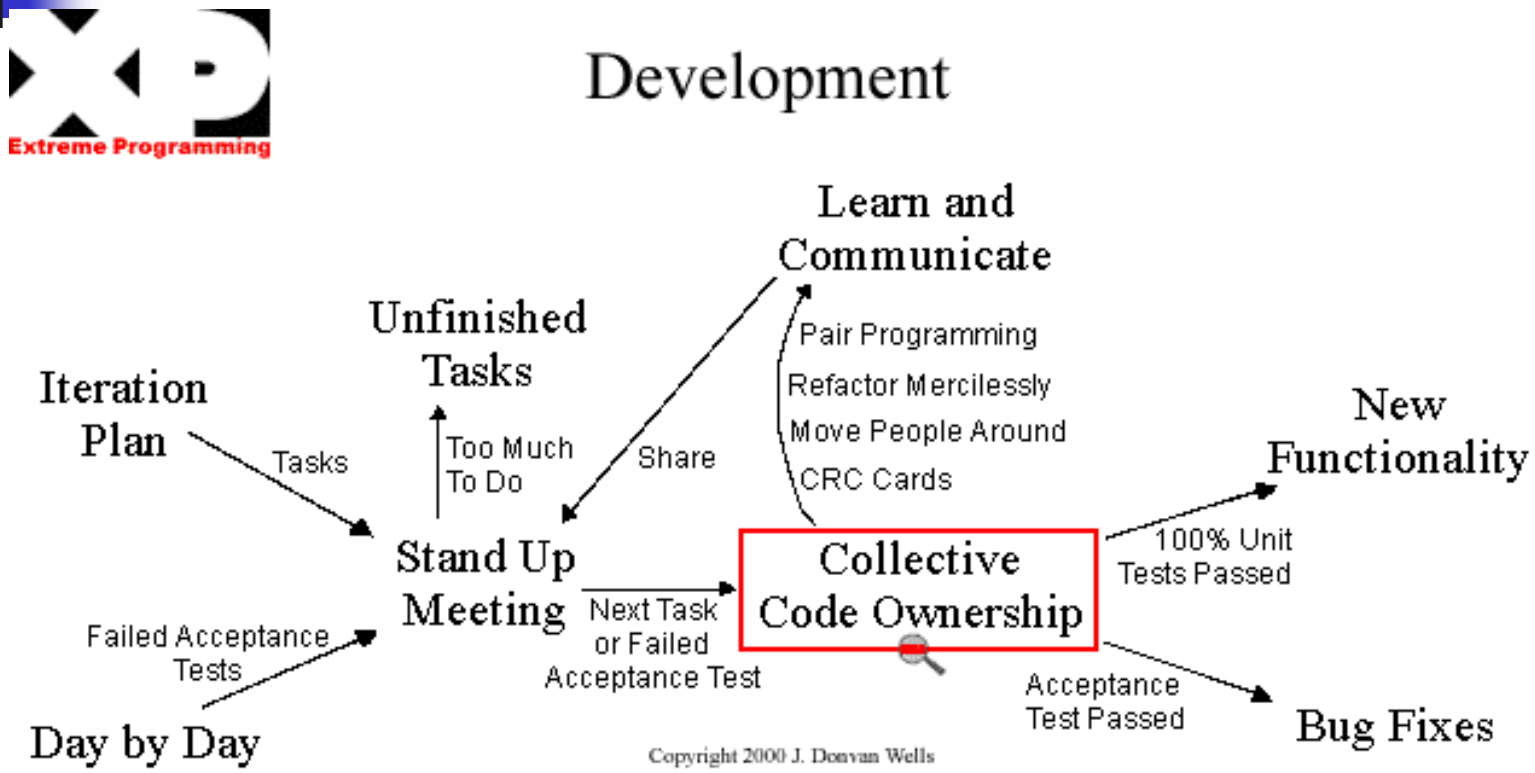
Praktyki XP związane z kodowaniem i testowaniem

- Integracja poszczególnych modułów tworzących system jak najczęściej i jak najwcześniej jest to możliwe
- Po integracji należy sprawdzić czy nowa wersja systemu przechodzi wszystkie zdefiniowane do tej pory testy
- Kod, pliki źródłowe, dokumenty stanowi wspólną własność zespołu, każdy członek może (i powinien) modyfikować kod napisany przez innych członków (propagacja wiedzy na temat systemu w grupie projektowej). Lepsze wykorzystanie zasobów i obniżenie ryzyka związanego z odejściem kluczowych dla projektu pracowników

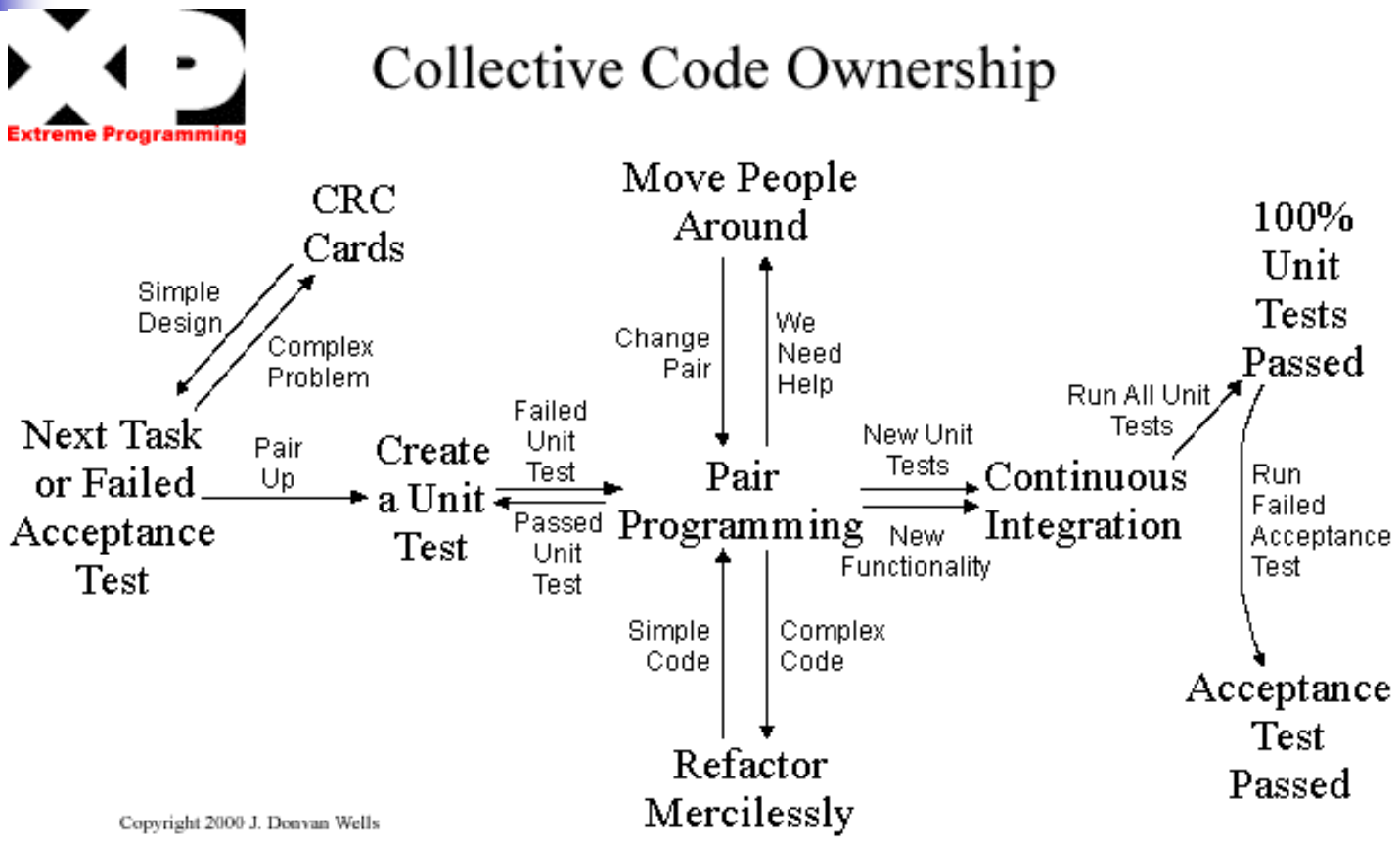
Praktyki XP związane z kodowaniem i testowaniem

- Optymalizacja dokonywana jest w końcowej fazie implementacji
  - *„Make it work, make it right, make it fast”*
- Testy akceptacyjne tworzone są przed implementacją i wykonywane często w trakcie kolejnych iteracji. Kod, który przeszedł testy akceptacyjne jest natychmiast dostępny dla Klienta (w postaci kolejnych wersji systemu) wraz rezultatami testów
- Klient odpowiedzialny jest za analizę wyników testów i zgłoszenie ewentualnych uwag, które mogą wpłynąć na priorytety zadań w kolejnych iteracjach

Implementacja zadań (1/2)



Implementacja zadań (2/2)



Kiedy nie należy stosować XP

- Jeżeli wykorzystywany proces umożliwia produkcję oprogramowania w sposób satysfakcjonujący Klientów i członków grupy projektowej
- W bardzo dużych zespołach projektowych (kiedy uznajemy zespół za duży ??)
- W przypadku gdy wymagania nie ulegają zmianom (przypadek bardzo, bardzo rzadki !)
- Zarządzanie częstymi zmianami nie jest możliwe a ich koszty są bardzo wysokie (np. ze względu na wykorzystane technologie)
- Możliwość uzyskania szybko informacji zwrotnej na temat dostarczanych w ramach iteracji produktów jest bardzo ograniczona
- Zespół projektowy i/lub Klient pracuje w różnych, odległych lokalizacjach

Kiedy należy stosować XP

- W każdym innym przypadku niż przedstawione na poprzedniej slajdzie a w szczególności jeżeli:
  1. Wymagania dotyczące systemu nie są ustabilizowane i mogą podlegać częstym zmianom (nawet po dostarczeniu i wdrożeniu systemu)
  2. Istnieje potrzeba zarządzania ryzykiem
  3. Istnieje grupa projektowa składa się z deweloperów o wysokich umiejętnościach
  4. Istnieje możliwość budowy automatycznych testów
  5. Wymagana jest większa efektywność pracy zespołu projektowego
  6. Działający system ma większe znaczenie niż jego dokumentacja ☺