Opracowanie do kolokwium
Programowanie ekstremalne 2019

Spis treści

1 Źródła

Opracowanie na podstawie:

- Wykłady z UPEL
- Pytania z kolokwium 2018 (wiki.stosowana)
- http://www.extremeprogramming.org/

2 Spis wykładów chronologicznie

1. Metody Zwinne - XPM - Plik pdf: Wykład cz.1
2. XPM testy jednostkowe - Plik pdf: Wykład - Testowanie w XPM
3. Metodyki Zwinne - XPM - Plik pdf: Wykład cz.2
4. XPM Projektowanie zwinne - Plik pdf: Wykład - Projektowanie w XPM
5. Wzorce projektowe dla Agile - Plik pdf: Wykład - Wzorce projektowe w XPM

Było coś jeszcze o refactoringu, ale nie ma nawet tego na UPEL więc pewnie nie łapie się w zakres...

3 Wykład cz.1

3.1. Slajd 1

Czym jest XP ?

XP należy interpretować jako metodykę zwinną planowania i realizowania projektów informatycznych.

Czym są User Stories ?

Opisują wymagania systemu(dla bieżącej iteracji), są podstawą do planowania iteracji i budowania testów jednostek.

Czym jest Architectural Spike ?

Ogólne określenie rozwiązań technologicznych (środowisko, narzędzia) użytych przy wytwarzaniu systemu. Pozwala na wstępną redukcję ryzyka, wynikiem Architectural Spike jest tzw. Spike Solution.

Czym jest System Metaphor ?

Wstępna koncepcja architektury systemu powstała po analizie Architectual Spike. System Metaphor służy klientowi aby będąc osobą nietechniczną mógł zrozumieć działanie systemu oraz programistom aby mogli lepiej zrozumieć działanie systemu.

Kolokwium 2018

Co to jest i do czego służy System Metaphor ?

3.2. Slajd 2

Czym jest Release Planning ? (Release Plan)

Plan, którego założeniem jest wypuszczenie kolejnej wersji produktu po przejściu określonej liczby iteracji.

Po co stosujemy testy w XP ?

- Stosowane do walidacji jednostek (testy jednostkowe)
- Stosowane do zaliczania iteracji zgodnie z Release Plan (testy akceptacyjne)

Do czego służą testy akceptacyjne ?

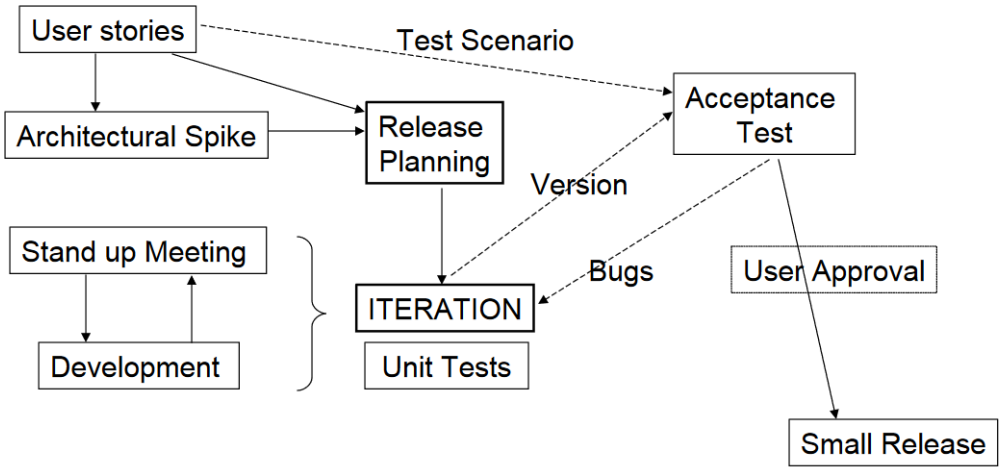
Ich zaliczenie umożliwia prezentację wersji (release) systemu klientowi. Release nie jest obowiązkowy - można przejść do kolejnej iteracji z kolejną user story bez zrobienia releasu, a z zakończonym pozytywnie testem akceptacyjnym.

Czym jest small release ?

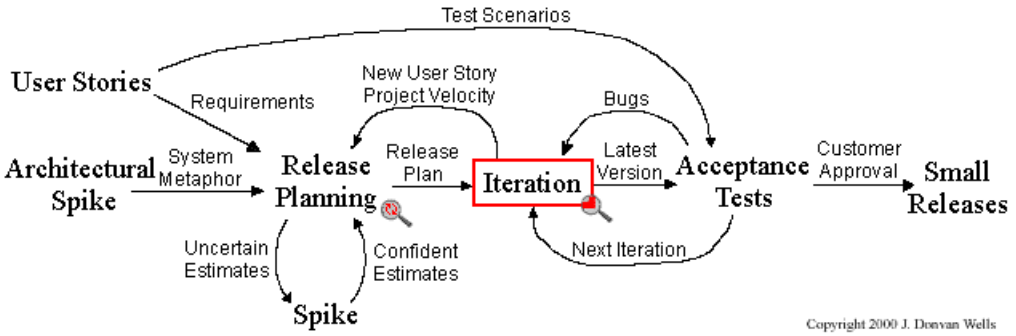
Jeśli wersja została zaakceptowana przez klienta, to może zostać wypuszczona, co nazywane jest small release.

3.3. Slajdy 3,4

Taka ogólna mapka XP...


User stories
Test Scenario
Acceptance Test
Architectural Spike
Release Planning
Version
Stand up Meeting
Bugs
User Approval
ITERATION
Development
Unit Tests
Small Release

Ale ta jest chyba lepsza...



Czym się kończy iteracja, a co ją poprzedza ?

- Iteracja kończy się osiągnięciem działającej wersji systemu (po zakończeniu kodowania), złączeniu i przetestowaniu nowo powstałej wersji systemu.
- Przed rozpoczęciem iteracji analizowany jest tzw. architectural spike (według wykładu)

3.4. Slajdy 4-7

Wymień 4 postulaty XP

- Planowanie
- Projektowanie
- Implementacja
- Testowanie

Opisz postulat XP: Planowanie

Prowadzone jest iteracyjnie, ma na celu wyodrębnić zadania do realizacji w danej iteracji. Zadanie muszą być uproszczone, czyli na tyle małe aby zmieściły się w iteracji. Oszacowania czasowe zawsze ujmuje się z dołu, czyli zakładając iż realizacja zadania potrwa na pewno nie mniej niż to zostało określone w planie.

Opisz postulat XP: Projektowanie

- Projekt funkcjonalności przyszłego systemu, jeśli w ogóle istnieje, to zakłada maksymalną prostotę.
- W czasie realizacji poszczególnych iteracji nie dodaje się żadnej zbędnej funkcjonalności.

- Dodatkową funkcjonalność dodawać będzie można w przyszłości – dopiero wtedy, gdy zajdzie taka potrzeba.
- Także kod posiadający funkcjonalność nadmiarową na skutek wycofania się klienta z wcześniej stawianych wymagań winien być z niej oczyszczony.

Opisz postulat XP: Implementacja

Metodyka XP ingeruje silnie w kwestie tworzenia kodu implementowanego systemu. Definiuje szereg wytycznych, postulujących między innymi:

- Utrzymywanie ściśle określonej konwencji kodowania,
- Ciągłe definiowanie testów tworzonego kodu (jeszcze przed jego utworzeniem),
- Często prowadzoną integrację kodu,
- Programowanie w parach,
- Współdzielenie kodu przez wszystkich członków zespołu

Opisz postulat XP: Testowanie

- System testowania tworzonego kodu jest sztandarowym atrybutem programowania ekstremalnego, odróżniającym tą metodykę od innych jest nacisk postawiony na testowanie.
- Testy są definiowane nawet zanim powstanie kod, swoją treścią przybliżając zakres przyszłej funkcjonalności systemu.
- Kod jest umieszczany w ogólnodostępnym dla zespołu repozytorium dopiero po przejściu testów
- Te same testy są prowadzone po zintegrowaniu kodu, jego rozszerzeniu funkcjonalnym czy refaktoryzacji
- Testy są również środkiem zapobiegającym nawrotom już wykrytych i usuniętych błędów w oprogramowaniu, po usunięciu błędu tworzony jest test.

3.5. Slajdy 9-16

Co robi programista w XP ?

Podobnie jak w innych metodykach zajmuje się wytwarzaniem kodu realizującego konkretną funkcjonalność.
Inaczej niż w innych metodykach, jest on odpowiedzialny za całość wytwarzanego kodu więc jego zakres kompetencji jest większy. Programista odpowiada dodatkowo za pisanie testów jednostkowych, wydzielanie zadań z historyjek oraz ich estymację, refaktoryzacje kodu.

Kolokwium 2018

Jak różni się rola Programista w XP od metodyk klasycznych ?

Co robi tester w XP ?

Tester nie jest jedynie przeciwwagą dla programisty, pełni szerszą rolę. Dostarcza informację o wynikach prowadzonych periodycznie testów. Tester współpracuje z programistami w wytwarzaniu oprogramowania oraz z klientem przy tworzeniu testów, co pozwala uzyskać informację uzupełniającą nt. wymagań.

Co robi klient w XP ?

Dostarcza podstawowej informacji o wymaganiach. Współpracuje z testerem w dziedzinie tworzenia testów (lub nawet tworzy je sam). Posiada najbardziej autorytatywną wiedzę nt. zasad funkcjonowania systemu. Dokonuje oględzin gotowych wersji produktu.

Na jakie role można podzielić klienta ? Opisz każdą z ról.

- Storytellers (Goal Donor) - mają wiedzę o działaniu systemu, definiują wymagania
- Acceptors - użytkownicy systemu, przeprowadzają testy akceptacyjne
- Gold Owners - podstawowe źródło finansowania projektu, inwestor
- Planners - pracują nad zdefiniowaniem i zarządzaniem wdrożeniem systemu
- Big Boss - dba o całościowy sukces projektu, monitoruje stan projektu

Kolokwium 2018

Kto to jest Gold Donor ?

Co oznacza klient TBD (to be determined) ?

Projekty tworzone w sytuacji gdy klient nie wie jeszcze, że potrzebuje danego systemu, rolę klienta musi pełnić członek zespołu.

Kolokwium 2018

Uzasadnij czy metodyka XP może być stosowana w projektach dla klientów TBD.

Co robi konsultant w XP ?

Konsultant to osoba mająca wiedzę dziedzinową w danej branży, konsultant nie śledzi całego procesu wytwarzania oprogramowania, ale zespół zwraca się do niego z konkretnym problemem.

Co robi zarządca w XP ?

Rolę zarządcy pełni wyłoniony jeden z członków zespołu. Zarządca archiwizuje przebieg prac (zwłaszcza powodzenie testów) oraz analizuje obłożenie prac w poszczególnych iteracjach oszacowując możliwość ukończenia projektu na czas.

Kolokwium 2018

Jakie funkcje XPM są przypisane do roli Zarządca ?

Co robi trener w XP ?

Trener to doświadczona osoba, która stymuluje zespół do podejmowania zadań idących we właściwym kierunku poprzez przedstawianie przypadków użycia, które wymagają specjalnej uwagi, a nie są oczywiste. Trener nie narzuca jednak konkretnych rozwiązać.

3.6. Slajd 17

Z czego składa się planning game, z jakich dwóch etapów ? W którym z tych etapów uczestniczy klient ? Jak często odbywa się planning game ? Co realizowane jest na każdym z tych etapów ?

Występuje przeważnie raz na iteracje (np. raz na tydzień). Składa się z dwóch etapów:

- Release Planning: Z udziałem klienta. Zakłada określenie wymagań stawianych najbliższej wersji oraz terminu jej dostarczenia
- Planning: Bez udziału klienta. Tu są planowane zadania i aktywności dla członków zespołu.

3.7. Slajd 18

Z jakich etapów składa się release planning ? Opisz co wykonywane jest w każdej z tych faz.

Trzy etapy:

- Exploration Phase: Klient przedstawia listę wysoko wartościowych cech produktu (z jego punktu widzenia). Cechy te zapisywane są na karty „user story”:
  Write a Story -> Estimate a Story -> Split a Story
- Commitment Phase: uzgodnienie zakresu prac dla iteracji oraz terminu zakończenia prac:
  Sort by Value->Sort by Risk->Set Velocity->Choose Scope
- Steering Phase: Ewentualne korekty planu: dodawanie, modyfikacja lub usuwanie funkcjonalności.

Kolokwium 2018

Na jakiej podstawie wybierane są historyjki do realizacji w pierwszej kolejności ?

3.8. Slajd 19

Czym jest user story ?

- Stanowi jednostkową funkcjonalność w projekcie XP
- Jest przypomnieniem (udokumentowaniem) konwersacji z klientem

Jakie są cechy dobrego user story ?

- zrozumiała (zarówno dla klienta jak i dewelopera) – najlepiej zapisana w języku naturalnym
- testowalna
- wartościowa dla klienta
- wystarczająco mała, żeby można zaimplementować ich kilka w ciągu jednej iteracji
- (możliwie) niezależna od innych

3.9. Slajd 20

Przykłady user stories...

3.10. Slajd 21

Do czego służy technika CRC Cards (Class, Responsibilities, Collaboration) ? Jak jest realizowana ?

Karty te opisują części systemu (np. dany moduł, interfejs, integracje modułów), ich celem jest uporządkowanie wszystkich modułów. Karta zawiera nazwę modułu (class), jego funkcje (responsibilities) oraz nazwy modułów z którymi współpracuje (Collaboration).
Gdy dane karty opisują przedmioty uzależnione – kładziemy je pod sobą w kolumnie (na wierzchu jest pierwsza rzecz do realizacji). Gdy przedmiot rozważań jest niezależny – kartę kładziemy w kolejnej kolumnie.

Kolokwium 2018

Co oznacza akronim CRC Card?

3.11. Slajd 22

Z jakich etapów składa się Iteration Planning ? Co się dzieje w każdym z nich ?

Trzy etapy:

- Exploration Phase: Wymagania są rozbijane na zadania (tasks). Tworzone są karty z zadaniami.
- Commitment Phase: Przypisywanie zadań (tasks) do programistów i estymacja (z ich udziałem) czasu potrzebnego na realizację
- Steering Phase: Zadania są realizowane i efekty ich realizacji są integrowane w ramach realizacji pierwotnej historii użytkownika (user story).

Kolokwium 2018

W której fazie (podaj jej nazwę) “Iteration Planning” odbywa się estymowanie złożoności danego zadania (tasku)?

3.12. Slajd 23

Jakie są czynności wykonywane w Steering Phase aby wykonać dane zadanie ?

1. Weź taska
2. Znajdź partnera
3. Zaprojektuj taska
4. Napisz test jednostkowy
5. Napisz kod
6. Odpal kod
7. Zrefaktoruj kod
8. (Sprawdź testy akceptacyjne)

3.13. Slajd 24

Dlaczego pierwsza iteracja ma szczególne znaczenie w XP ?

- Definiuje rdzeń systemu na bazie którego będą budowane następne iteracje
- Ma duże znaczenie we współpracy z klientem ponieważ tworzy pierwsze wrażenie

Kolokwium 2018

Dlaczego pierwsza iteracja jest tak ważna ?

Co powinna dostarczać pierwsza iteracja ?

- Powinna dostarczyć kilku najbardziej wartościowych historii użytkownika dotyczących podstawowych funkcjonalności
- Dostarczane historie użytkownika nie powinny zależeć od innych historii nie uwzględnianych w tej iteracji
- Pierwsza iteracja (z klientem) nie może być iteracją z zerową funkcjonalnością !

3.14. Slajd 25

Do czego służy zerowa iteracja w XP ?

- Przygotowanie środowiska pracy (instalacja i konfiguracja środowiska programistycznego, przygotowanie środowiska do testów, system kontroli wersji itp.)
- Przygotowanie infrastruktury deweloperskiej
- Zerowa iteracja odbywa się bez klienta, a pierwsza już z klientem

3.15. Slajd 26

Jakie czynności odbywają się periodycznie w ramach planowania iteracji ?

- Projektowanie wymagań powstałych po uwzględnieniu User Stories
- Estymowanie pracochłonności realizacji założeń projektowych opisanych w User Stories
- Ciągłe uzupełnianie puli rozwiązań technicznych (Spike solutions, np. biblioteki, wzorce projektowe)

3.16. Slajd 27

Czym są Story-Points ? Jak są wyrażane ?

Story Points używane są do wyceny (estymacji) każdego elementu historii użytkownika. Są to bezjednostkowe punkty przypisywane do każdego zadania. Konieczne jest zdefiniowanie skali Story-Points; kolejne liczby, litery, ciąg Fibonacciego itp.

Kolokwium 2018

Co to są Story Points?

Kolokwium 2018

Jakiej miary (pośredniej lub bezpośredniej) używamy do szacowania czasochłonności historyjek ?

Trochę nie wiem o co chodzi w tym pytaniu. Z jednej strony User Story są szacowane przez Story Points, skale sobie dobieramy sami. User Story są rozbijane na taski i one znów mają przypisaną estymatę w roboczodniach, czyli idealnych dniach pracy programisty bez zakłóceń... (?)
Oprócz tego na extremeprogramming.org jest napisane:

> The essence of the release planning meeting is for the development team to estimate each user story in terms of ideal programming weeks. An ideal week is how long you imagine it would take to implement that story if you had absolutely nothing else to do. No dependencies, no extra work, but do include tests. The customer then decides what story is the most important or has the highest priority to be completed.

Może więc chodzić o to, że Story Points jako miara pośrednia, a roboczo-tygodnie jako miara bezpośrednia (?).

3.17. Slajd 28

Opisz technikę Planning Poker

Służy do estymowania User Stories.

1. Klient odpowiada na pytania zespołu
2. Każdy członek zespołu określa estymatę (typuje kartę bez jej ujawniania)
3. Głos otrzymuje autor estymaty najwyższej i najniższej
4. Dyskusja jest limitowana czasowo stoperem, po dyskusji następuje kolejna iteracja estymowania aż do uzyskania zgodności

Kolokwium 2018

Opisz przebieg estymowania z użyciem techniki “Planning Pokera”

3.18. Slajd 29

Co oznacza ’?’ na karcie do planning pokera ? A co oznacza kawa ?

- ’?’ oznacza brak zdania, często sugerujący też, że historia jest zbyt skomplikowana do estymowania

- Kubek z kawą oznacza prośbę o przerwę

Kolokwium 2018

Co oznacza znak '?' na kart do Planning Poker ?

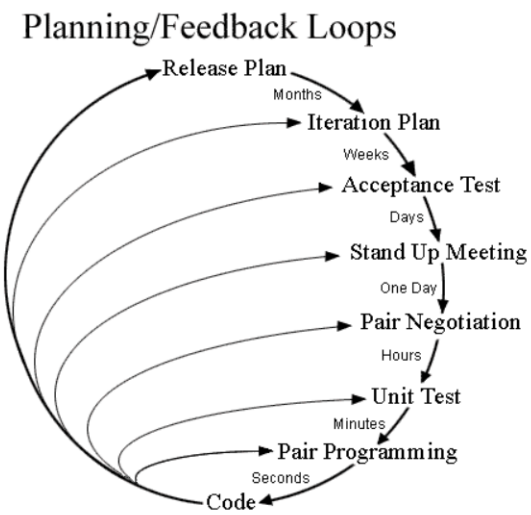
3.19. Slajd 30

Wymień zalety planning pokera

- Lepsza komunikacja w zespole
- Lepsza ocena user story przez zaangażowanie wielu osób
- Możliwość poruszania wątków pokrewnych względem danego user story
- Zespołowa akceptacja planu do wykonania

3.20. Slajd 31

Pętla zwrotna XP...



Kolokwium 2018

Ile standardowo powinna trwać iteracja w XP ?

(Od jednego do kilku tygodni)

3.21. Slajdy 32-37

Wymień 10 praktyk stosowanych w XP

- Whole team (jedność zespołu)
- Kondycja zespołu
- Informative workplace (Informatywne miejsce pracy)
- Customer visible functionality (Wartość odczuwalna dla klienta)
- Planning Game (Iteracyjne planowanie)
- Pair programming (Programowanie w parach)
- Spike (Szybkie prototypowanie)
- Collective code ownership
- Single code base (Pojedyczna linia oprogramowania)
- Wysoka jakość wytwarzanego oprogramowania

Jak realizowana jest praktyka whole team (jedność zespołu) ?

- sit together - zespół pracuje razem o ile to możliwe przebywając w jednym pomieszczeniu
- team continuity - zachowanie ciągłości składu osobowego
- on-site customer - w skład zespołu wchodzi klient

Kolokwium 2018

40-hour week, team continuity

Jak realizowana jest praktyka kondycja zespołu ?

- 40-hour week - brak nadgodzin
- slack - zezwolenie na pewną swobodę wykorzystania czasu pracy
- sustainable pace - stałe tempo pracy

Kolokwium 2018

Za pomocą jakich zasad dbamy o dobrą kondycję zespołu

Kolokwium 2018

40-hour week, team continuity

Jak realizowana jest praktyka informative workplace (Informatywne miejsce pracy)?

Informacje istotne dla zespołu są udostępnione bezpośrednio w miejscu pracy zespołu, w sposób umożliwiający wszystkim zainteresowanym nieograniczony dostęp do nich.

Jak realizowana jest praktyka wartość odczuwalna (customer visible functionality) ?

Historie użytkownika rozważa się z punktu widzenia wartości biznesowej, ryzyka, kosztów wytworzenia uzupełnionych o testy akceptacyjne.

Jak realizowana jest praktyka iteracyjne planowanie (planning game) ?

- Dyskutowane i estymowane są historyjki oraz każda z iteracji, które wchodzą w zakres danego release
- Na podstawie historyjek tworzy się listę zadań i następuje podjęcie zobowiązania przez programistów (sign-up)

Jak realizowana jest praktyka pair programming ? Kim jest driver a kim pilot ?

- Praca w dwuosobowych zespołach, często zmieniających się
- Jedna osoba z zespoły zapisuje kod (driver), druga dba o zgodność z projektem, przejrzystość, spójność itp. tworzonego kodu (pilot).

Jak realizowana jest praktyka szybkie prototypowanie (spike) ?

Niewiadome i ryzyka w projekcie rozwiązuje się poprzez szybkie prototypowanie (max kilka dni)

Jak realizowana jest praktyka wspólna własność kodu źródłowego (collective code ownership) ?

Prawo do mydyfikowania całego kodu źródłowego utrzymywanego przez zespół mają wszyscy jego członkowie, a nie tylko specjalizujący się w danym fragmencie programiści.

Jak realizowana jest praktyka pojedyncza linia oprogramowania (single code base) ?

Polega na utrzymywaniu pojedynczej bazy kody źródłowego, wspólnej dla wszystkich linii produktu. Robi się to po to aby uniknąć błędów, które często pojawiają się przy wprowadzaniu tej samej poprawki do wielu linii produktu. Nowe funkcjonalności wprowadza się poprzez rozwiązania architektoniczne takie jak mp. konfigurowalne włączanie/wyłączanie funkcjonalności w zależności od odbiorcy końcowego lub platformy docelowej. (np. Firefox ma różne pluginy).

Kolokwium 2018

W jaki sposób można pogodzić praktykę single code base z koniecznością istnienia różnych wersji oprogramowania dla różnych odbiorców końcowych ?

Jak realizowana jest praktyka wysoka jakość wytwarzanego oprogramowania ?

- Wykorzystanie testów jednostkowych i akceptacyjnych
- Często konsolidacja całego produktu (continous integration)
- Testowe wdrożenia (deployment)
- Aby to zapewnić budowanie systemu powinno nie zabierać więcej niż kilka minut

3.22. Slajd 38

Co oznacza TDD i test first ?

- Test first - podejście mówiące, że najpierw tworzymy testy, a potem kod
- TDD (Test driven development) - określanie funkcjonalności poprzez planowanie testów

Jakie rodzaje testów przewiduje metodyka XP ?

- Testy jednostkowe (unit tests)
- Testy akceptacyjne (acceptance tests)
- Testy interaktywne lub wizualne (visual tests)

Kolokwium 2018

Rodzaje testów w XP

3.23. Slajdy 39-40

Omów testy jednostkowe

- Tworzone są przez programistów
- Są automatyczne
- Dotyczą każdego elementu do którego można przekazać dane w celu ich przetworzenia
- Walidują postęp prac

Jakie są etapy tworzenia kodu w TDD ?

1. Tworzenie kodu
2. Implementacja
3. Refaktoryzacja do standardów, potem znów test

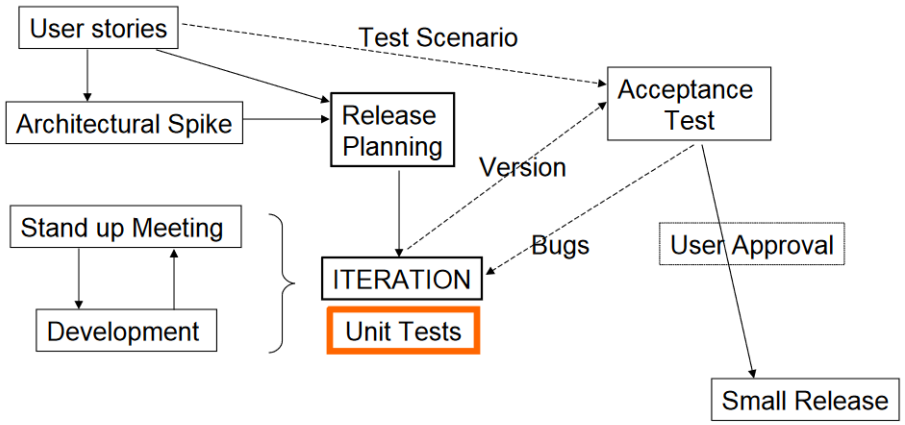
3.24. Slajd 41

Omów testy akceptacyjne

- Są projektowane przez klienta
- Kodowane są przez programistę gdy klient nie jest w stanie ich samodzielnie wyrazić
- Opisują oczekiwania klienta wobec powstającego produktu
- Nazywane są czasem testami funkcjonalnymi

4 Wykład - Testowanie w XPM

4.1. Slajd 1



4.2. Slajd 2

Co rozumiemy przez jednostkę w kontekście programowania ekstremalnego ?

Jednostka to komponent, klasa, metoda, do testowania jednostek nie jest konieczna integracja projektu.

4.3. Slajd 3

Obrazek Eclipse z JUnit...

4.4. Slajdy 4,5

Za co odpowiada dyrektywa JUnit:
@Test public void metoda ()

Uruchomienie testu metody

Za co odpowiada dyrektywa JUnit:
@Test (expected = ExceptionClass) public void metoda ()

Uruchomienie testu metody i sprawdzenie, czy wygenerowała wyjątek ExceptionClass

Za co odpowiada dyrektywa JUnit:
@Test (timeout=20) public void metoda ()

Uruchomienie testu metody i pomiar max czasu wykonania (milisekundy).

Za co odpowiada dyrektywa JUnit:
@Ignore public void metoda ()

Wyłączenie z testu (np. metody mock)

Za co odpowiada dyrektywa JUnit:
@Before public void metoda ()...

Uruchomienie metody przed każdym testem (dla przygotowania środowiska testu – otwarcie pliku, połączenia itp.)

Za co odpowiada dyrektywa JUnit:
@After public void metoda ()...

Uruchomienie metody po każdym teście (sprzątanie)

Za co odpowiada dyrektywa JUnit:
@BeforeClass public void metoda ()...

Uruchomienie raz dla całego procesu testowania (TestSuite) – dla inicjalizacji testowej wersji „systemu", ustawiania wartości statycznych.

Kolokwium 2018

Do czego typowo w testach tworzonych w JUnit używana jest adnotacja @BeforeClass ?

Za co odpowiada dyrektywa JUnit:
@AfterClass public void metoda ()...

Uruchomienie metody jednorazowo po testach

4.5. Slajd 6

Do czego służy w JUnit asercja:
fail(komunikat)

Bezwarunkowe zakończenie testu niepowodzeniem

Do czego służy w JUnit asercja:
assertTrue([komunikat], boolean wyrażenia)

Sprawdzenie wyrażenia

Do czego służy w JUnit asercja: assertsEquals([komunikat], expected, actual)

Porównanie dwóch wartości

Do czego służy w JUnit asercja: assertNull([komunikat], object), assertNotNull([komunikat], object)

Sprawdzenie istnienia obiektu

Do czego służy w JUnit asercja: assertSame([komunikat], expected, actual), assertNotSame([komunikat], expected, actual)

Sprawdzenie identyczności referencji

4.6. Slajd 7

Przykład testu JUnit:

```
import org.junit.*;
import static org.junit.Assert.*
public class Testy { // Klasa testu

@BeforeClass
public static void initTestowanie() {Testowanie.stat = 2;}
@AfterClass
public static void closeTestowanie(){Testowanie.stat = 0;}

@Test
public void test() {
      Testowanie te1 = new Testowanie();
      assertEquals(1, (int)te1.MetodaTestowana(2));
}
```

4.7. Slajd 8

Czym są tzw. Test Suites w JUnit ?

Test suite to połączenie wielu testów w jeden proces.

Jakie są dyrektywy używane przez Test Suites ?

- @RunWith(Suite.class) - określa, że mamy do czynienia z TestSuite a nie pojedynczy testem
- @SuiteClasses({ PierwszyTest.class, DrugiTest.class }) - otrzymując listę klas testów powoduje uruchomienie ich w kolejności wyszczególnienia

Kolokwium 2018

Dlaczego podczas tworzenia programów zgodnie z techniką TDD ważne jest używanie Test Suites ?

- Umożliwia to automatyzację wykonywania wszystkich testów
- Umożliwia to łatwe dodawanie kolejnych testów (funkcjonalności) do całego projektu
- Umożliwia sprawdzenie czy nowo dodana funkcjonalność nie psuje zaimplementowanych dotychczas

4.8. Slajdy 9,10,11

Do czego służą dyrektywy:
@RunWith(Parameterized.class)
@Parameters

Cel: uruchomienie serii testów kodu (np. metody) każdorazowo z różnymi parametrami.

- @RunWith(Parameterized.class) - przekazanie parametru do konstruktora klasy testu
- @Parameters - umieszczenie w klasie metody zwracającej tablicowaną kolekcję parametrów i opatrzonej tą dyrektywą

Przykład kodu w wykładzie...

4.9. Slajd 12

Po co stosowany jest Runner w JUnit ?

Stosowany do uruchomienia testów poza IDE lub w specjalnym trybie

4.10. Slajdy 13-20

Na tych slajdach są dyrektywy i asercje, ale tym razem dla NUnit (C#)...

4.11. Slajd 21

Czym jest obiekt mock w testach jednostkowych ?

Mock stosujemy w testach do zasymulowania zachowania klasy jeszcze nie zaimplementowanej. Mock implementuje (wręcz – symuluje) ten sam interfejs, co klasa, jaką zastępuje – podmieniamy ten interfejs w klasie testowanej. Jest konfigurowalny (można przed testem ustalić jego zachowanie).

Kolokwium 2018

W jakim celu w testach jednostkowych używa się Mock Objects?

4.12. Slajdy 22-28

Szersze omówienie stosowania obiektów mock w JUnit i NUnit

5 Wykład cz.2

5.1. Slajd 1

Czym są Testy wizualne w XP ?

Komplet wytycznych (zapisanych na przykład w formie listy pozycji do sprawdzenia) umożliwiający sekwencyjne przeprowadzenie wizualnej kontroli funkcjonowania systemu. Kontrola ta prowadzana jest głównie z punktu widzenia aktorów (użytkowników systemu) i dotyczy oceny interfejsów systemu. W praktyce sprowadza się ona do kontroli ergonomii interfejsów graficznych, lub estetyki prezentowanych za ich pośrednictwem treści.

5.2. Slajd 2

Pod jakim warunkiem można wykonać modyfikację architektury w XP ?

Jeśli modyfikacja architektury ułatwi przejście danej iteracji i nie zepsuje wyników testów uzyskanych na poprzednich, należy ją wykonać.
Metryka poprawności: Testy jednostkowe (Unit tests)

5.3. Slajd 3

Jaki jest warunek aktualizacji kodu we wspólnym repozytorium ?

Warunek aktualizacji kodu w repozytorium: poprawne wykonanie testu jednostki + poprawne wykonanie testu ewentualnej nowej funkcjonalności jednostki

5.4. Slajdy 4-6

Czym jest i co zawiera dokument standardów kodowania (coding standards) ?

Specyfikacja standardu powinna być dokumentem krótkim, nie zawierającym zbędnych szczegółów - lecz jednoczenie nie pozostawiającym niedomówień w dziedzinie utrzymywanej konwencji nazewnictwa, formatowania bloków, czy komentowania kodu.

5.5. 7,8

Na czym polega refaktoryzacja ? Wymień kilka zabiegów stosowanych w refaktoryzacji.

Refaktoryzacja to zabiegi umożliwiające utrzymanie czytelnej struktury kodu. Do zabiegów należą przykładowo:

- Korekta nazewnictwa klas, atrybutów, metod
- Przeniesienie metody do innej klasy
- Stworzenie nowej metody na podstawie kodu wydzielonego z innej
- Wyodrębnienie interfejsu z klasy
- Wydzielenie wartości wyrażenia i zdefiniowanie w postaci stałej

Kolokwium 2018

Wymień 4 przykłady zabiegów wykonywanych podczas refaktoryzacji.

5.6. Slajdy 9,10

Za co odpowiada w parze pilot, a za co driver ?

- Pomocnik (pilot) dostarcza wiedzę pochodzącą z dokumentacji narzędzi, opracowania merytoryczne rozwiązań zadań, dba o zgodność z projektem, przejrzystość i spójność tworzonego fragmentu oprogramowania.
- Programista (driver) koduje zapisując kod źródłowy.

5.7. Slajd 11

Coś o krzywej uczenia...

5.8. Slajd 12

Czym są stand-up meeting ?

Spotkanie robocze (stand-up):

- Codzienne, traktowane jako otwarcie dnia prac
- Określenie postępu prac - każda para przedstawia tu raport
- Wykrycie trudnych (wymagających czasu) problemów
- Koordynacja wysiłków dla rozwiązania trudnych problemów

5.9. Slajd 13

Single release, że ma zalety, bo jest ciąga integracja, walidacja biznesowa itd. i, że trzeba zintegrować wszystkie testy i przygotować dla klienta do pokazania.

5.10. Slajd 14-16

> Czym jest ciągła integracja (continuous integration) ?

Ciągła integracja polega na regularnej konsolidacji (łączeniu modułów) i uruchamianiu zestawu testów wielokrotnie w ciągu dnia. Zwykle robi to jedna osoba na oddzielnej maszynie specjalnie dla tego celu. Programista po wykonaniu każdego nowego fragmentu programu łączy go z systemem.

5.11. Slajd 17

> Na czym polega zasada 40 godzinny tydzień pracy ? Czy rzeczywiście musi być to 40 godzin ?

Jedną z charakterystycznych cech programowania ekstremalnego jest definiowanie sztywnej marszruty czasowej dla programisty, będącą konkretną, nienaruszalną granicą obciążenia grupy. Zwyczajowo nazywamy ją 40-godzinnym tygodniem pracy (40-hour week). Nie jest ważne, ile godzin w tygodniu pracy programisty faktycznie zostało przyjęte jako norma (nie musi to być akurat 40 godzin). Ważne jest, aby norma po wyznaczeniu była utrzymywana ściśle i konsekwentnie.

5.12. Slajd 18

> Przez jakie stany przechodzi zadanie w procesie jego realizacji ?

TASK -> DESIGN -> TEST CODE -> TASK CODE -> DEBUG -> BUILD (DONE)

5.13. Slajd 19

> Czy w XP ryzykiem jest opóźnienie realizacji projektu ?

W małym stopniu, ponieważ zadania są dzielone na krótkie terminy, cały czas mamy kontrolę nad wydaniami i walidacją ze strony klienta.

5.14. Slajd 20

> Czy w XP ryzykiem jest anulowanie projektu ?

Może się to zdarzyć w powodu czynników niezależnych od stosowania samej metodyki. Dlatego na od klienta wymaga się aby definiował możliwe najmniejsze wartości biznesowe dla poszczególnych release, prowadzi to do mniejszej liczby wydań nieprodukcyjnych i ryzyko anulowania jest dużo mniejsze.

5.15. Slajd 21

Czy w XP ryzykiem jest problem rozszerzalności wersji ?

W XP rozbudowa systemu jest ułatwiona ze względu na wysokie standardy wytwarzanego oprogramowania i ciągłą refaktoryzację.

5.16. Slajd 22

Czy w XP ryzykiem jest problem niskiej jakości kodu ?

W XP jest kładziony bardzo duży nacisk na testowanie kodu, co wprowadza wysoką jakość produkowanego kodu.

5.17. Slajd 23

Czy w XP ryzykiem są nieporozumienia biznesowe ?

Klient traktowany jest jako część zespołu i cały czas ma wpływ na wymagania. Ponadto nie istnieje formalna specyfikacja wymagań przed rozpoczęciem projektu. Nieporozumienia biznesowe stanowią niskie ryzyko w XP.

5.18. Slajd 24

Czy w XP ryzykiem są zmiany biznesowe ?

W XP krótki cykl iteracji redukuje skutki wszystkich zmian biznesowych.

5.19. Slajd 25

Czy w XP ryzykiem jest niska wartość biznesowa produktu ?

W XP problem ten jest redukowany poprzez realizację najważniejszych dla klienta wartości biznesowych jako pierwsze.

5.20. Slajd 26

Czy w XP ryzykiem są rotacje kadrowe ?

XP wzmacnia kolaborację w zespole, ponadto krótkie czasy trwania realizacji zadań redukują to ryzyko.

5.21. Slajd 27

Jakie są cechy wspólne XP i SCRUM ?

- Samodzielność zespołu
- Zaangażowanie klienta końcowego
- Iteracyjne planowanie
- Regularne dostarczanie działającego oprogramowania

Kolokwium 2018

Na czym polega 'zwinność' w XP ?

5.22. Slajd 28

Jakie są różnice pomiędzy XP i SCRUM ?

- SCRUM dotyczy strony zarządzającej pozostawiając rozwiązania inżynierskie (projektowanie, kodowanie, zarządzanie konfiguracją itp.) zespołowi
- XP dotyczy posunięć inżynierskich lecz nie dostarcza precyzyjnych narzędzi do zarządznia projektem

5.23. Slajd 29

Kontrowersje związane z XP (Pair programming, brak dokumentacji...)

6 Wykład - Projektowanie w XPM

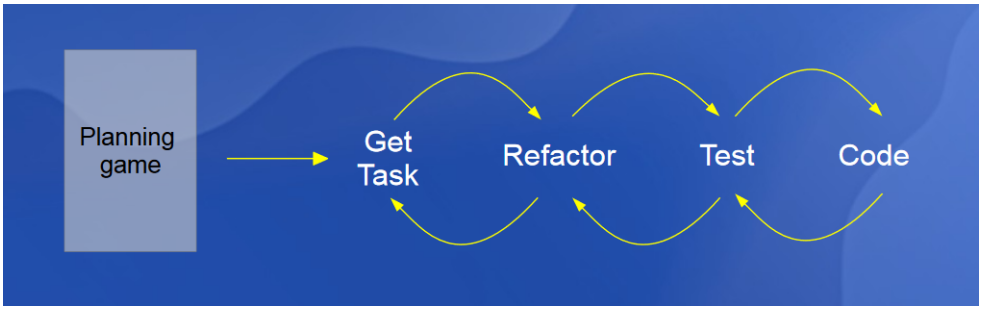
6.1. Slajd 1

> Czym jest projektowanie zwinne w XP ?

Polega na projektowaniu tylko w niezbędnym zakresie (just enough and no more) z wykorzystaniem architectural spike. Tworzona jest metafora systemu oraz wysokopoziomowa mapa dla całego systemu. Następnie projektowanie odbywa się w sposób ewolucyjny w ramach implementacji poszczególnych zadań. Jest wpisane w cyk deweloperski realizacji zadania.

6.2. Slajd 2

> Czym jest cykl deweloperski realizacji zadania ?



6.3. Slajd 3

> Wymień podstawowe zasady projektowania zwinnego w XP

SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation, Dependency inversion)substitution, Interface segregation, Dependency inversion)

- Zasada pojedynczej odpowiedzialności (Single Responsibility Principle)
- Zasada otwarte-zamknięte (Open-Closed Principle)
- Zasada podstawienia Liskov (Liskov Substitution Principle)
- Zasada segregacji interfejsów (Interface Segregation Principle)
- Zasada odwracania zależności (Dependency Inversion Principle)

6.4. Slajdy 4-8

Na czym polega zasada pojedynczej odpowiedzialności (Single responsibility) ?

- Żadna klasa nie może być modyfikowana z więcej niż jednego powoduwięcej niż jednego powodu
- Jeśli klasa odpowiada za więcej niż jeden obszar to powoduje związanie tych obszarów – obszar to powoduje związanie tych obszarów – klasy powinny podzielić między siebie obszary klasy powinny podzielić między siebie obszary odpowiedzialnościodpowiedzialnośc

6.5. Slajdy 9-12

Na czym polega zasada otwarte-zamknięte (Open closed)?

Składniki oprogramowania (klasy, moduły, funkcje, itp.) powinny być otwarte na rozbudowę, ale zamknięte dla modyfikacjirozbudowę, ale zamknięte dla modyfikację

- Otwarcie na rozbudowę – musi istnieć sposób stosunkowo prostej rozbudowy zachowań (funkcjonalności) modułu
- Zamknięcie dla modyfikacji – rozbudowa zachowań modułu nie może skutkować zmianą kodu źródłowego ani binariów tego modułu

6.6. Slajdy 13-15

Na czym polega zasada podstawienia Barbary Liskov (Liskov Substitution Reinciple)?

Zasada podstawienia Liskov - musi istnieć możliwość zastępowania typów bazowych ich możliwość zastępowania typów bazowych ich podtypami.podtypami.

6.7. Slajd 16

Na czym polega zasada segregacji interfejsów (Interface Segregation Principle)?

- Eliminowanie „grubych” interfejsów
- Grube interfejsy należy dzielić - małe grupy metod tworzą poszczególne interfejsy
- Klient nie powinien być zmuszany do zależności od metod, których nie używazależności od metod, których nie używa

Kolokwium 2018

Podaj nazwę zasady, która w akronimie SOLID odpowiada literze ’I’ oraz krótko opisz na czym ona polega.

6.8. Slajd 17-20

Na czym polega zasada odwracania zależności (Dependency Inversion Principle)?

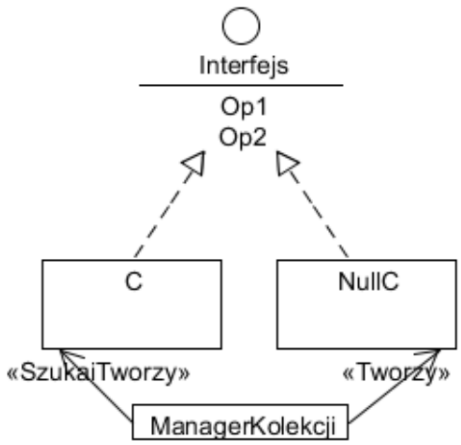
Moduły wysokopoziomowe nie powinny zależeć od modułów niskopoziomowych. Obie grupy od modułów niskopoziomowych. Obie grupy modułów powinny zależeć od abstrakcjimodułów powinny zależeć od abstrakcji

7 Wykład - Wzorce projektowe w XPM

7.1. Slajdy 1,2

Na czym polega wzorzec Null Object w Agile ?

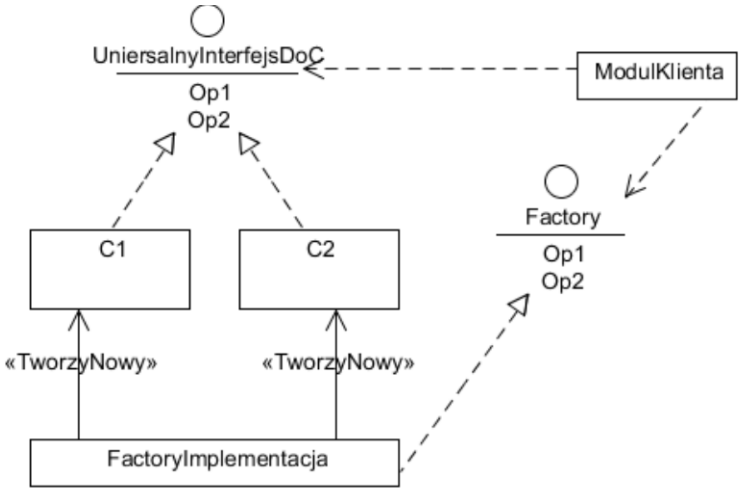
- Założenie: klasa C dostarcza funkcjonalność poprzezinterfejs I. Instancje C są przechowywane w kolekcjach.
- Oprócz klasy C tworzymy dodatkowa klasę NullC implementującą interfejs I, lecz przygotowaną tak aby implementującą interfejs I, lecz przygotowaną tak aby instancje tej klasy były zwracane po niepowodzeniu wyszukiwania w kolekcji obiektów C (zamiast zwracania wartości null)
- Wówczas kod zlecający wyszukiwanie nie spowoduje błędu, gdy nie zostanie sprawdzony przypadek zwracania null



7.2. Slajdy 3,4

Na czym polega wzorzec Factory w Agile ?

- Wzorzec faktory jest projektach zwinnych stosowany w celu spełniania reguły DIP (odwracania zależności)
- Stosowany, gdy spodziewana jest duża liczba wariantów klas wyprowadzonych z abstrakcyjnego wzorca.klas wyprowadzonych z abstrakcyjnego wzorca.
- Moduł tworzący instancje nie tworzy ich bezpośrednio, lecz korzysta z interfejsu Factory implementowanegoprzez przygotowaną do tego klasę.



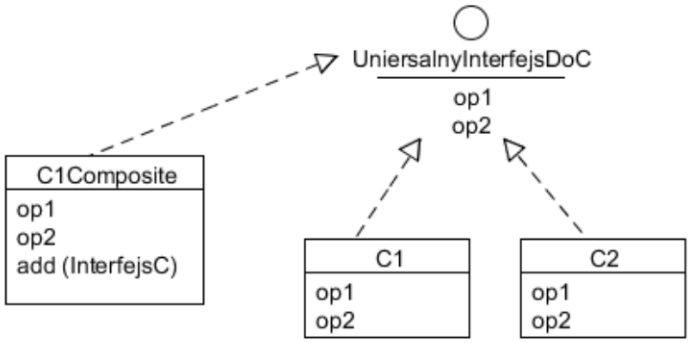
Kolokwium 2018

Wzorzec Agile zapewniający zasadę DIP

7.3. Slajdy 5,6

Na czym polega wzorzec Composite w Agile ?

- Popularny w Agile szczególnie w sytuacjach, gdy metody poszczególnych licznych klas wyprowadzonych realizują podobne operacje (np. samo-rysowanie obiektu).
- Wzorzec zakłada istnienie klasy katalogującej obiekty (przeważnie w postaci kolekcji).
- Instancja katalogująca posiada możliwość rejestrowania instancji klas wyprowadzonych
- Klasa katalogująca implementuje ten sam interfejs co kasy wyprowadzone
- Dodatkowo klasa katalogująca posiada metody realizujące iteracyjne wywoływanie odpowiednich metod obiektów katalogowanych



7.4. Slajd 7

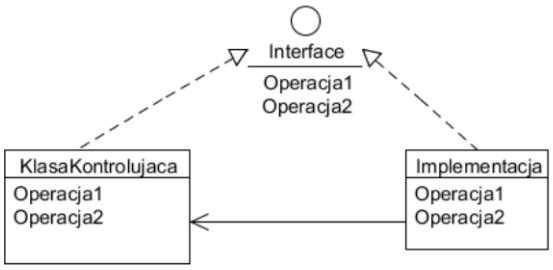
Na czym polega wzorzec Observer w Agile ?

Ogólnie działa to jak Listener w Javie w Swingu, monitorujemy stan obiektu i reagujemy na jego zmianę.

7.5. Slajd 8

Na czym polega wzorzec Proxy w Agile ?

Stosowany do kontroli funkcjonowania instancji innej klasy. Typowe zastosowanie to kontrola czasu wykonywania operacji przez instancję kontrolowaną.

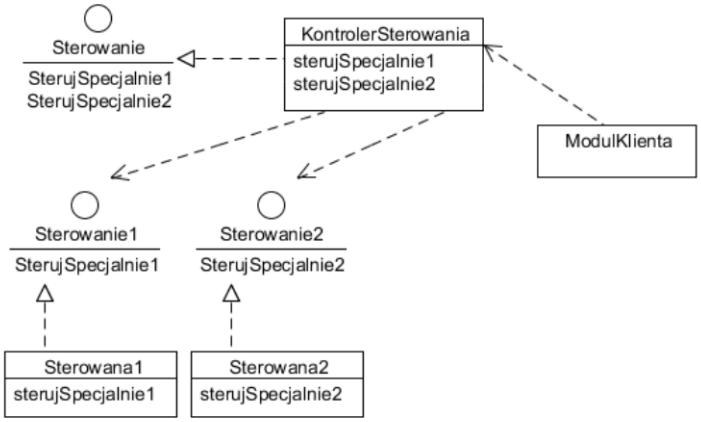


7.6. Slajdy 9,10

Na czym polega wzorzec Agile dla Abstract Server ?

- Kolejne rozwiązanie eliminujące naruszenia DIP (dependency inversion principle).

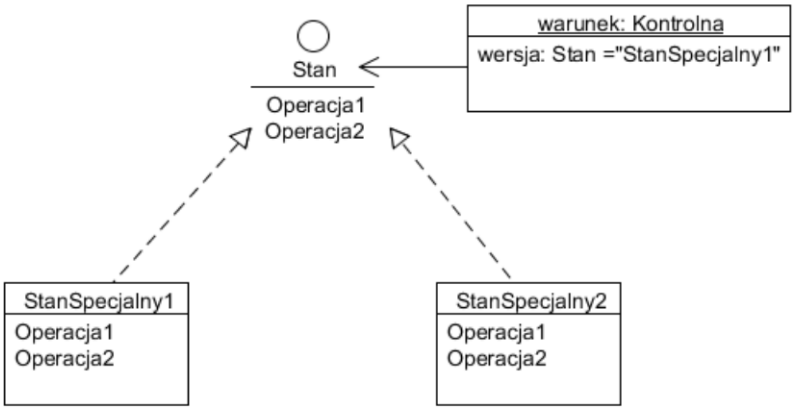
- Tworzymy uniwersalny interfejs opisujący czynności, za pomocą których steruje się instancjami klas o podobnych właściwościach. Rezygnujemy zatem z podobnych właściwościach. Rezygnujemy zatem z uzupełniania interfejsów dla specjalnych obiektów sterowanych (na rzecz jednego – uniwersalnego)
- Niwelowanie skutków stosowania reguły segregacji interfejsów – interfejsy będą agregowane przy pomocy„abstrakcyjnego serwera”



7.7. Slajdy 11,12

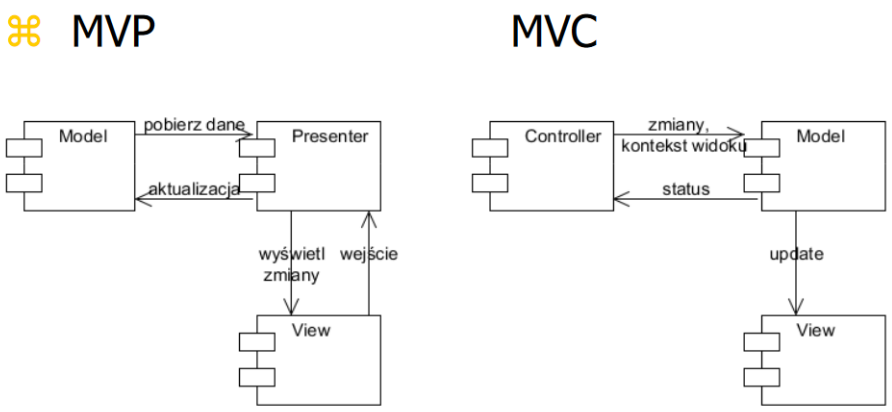
Na czym polega wzorzec State w Agile ?

- Bardzo popularny w Agile, definiuje skończoną maszynę stanów dla operacji realizowanych przez system
- Wzorzec zakłada posiadanie wielu implementacji i przełączanie działania systemu pomiędzy nimi
- Warunki przeniesione są do oddzielnych klas (powstają klasy warunku implementujące interfejsy, w których określone są metody wykonywane na różne sposoby)
- Tworzona jest klasa kontrolna, która otrzymuje nowąinstancję którejś z klas warunku (jako jedna z opcji). Z niej będą wołane odpowiednie implementacje metod



7.8. Slajdy 13-15

Czym się różni MVC od MVP ?



- W MVC kontroler obsługuje zdarzenia, manipuluje modelem, który zawiera logikę biznesową, widok wyświetla dane z modelu. Jeden kontroler obsługuje kilka widoków.
- W MVP dane z modelu są przekazywane do presentera a nie bezpośrednio do widoku, presenter przekazuje je do widoku. Jeden presenter odnosi się do jednego widoku.

Kolokwium 2018

Które elementy wzorca MVP można testować automatycznie ?

Model, presenter. (?)

8 Pozostałe pytania z kolokwium 2018

Kolokwium 2018

Które diagramy UML są najczęściej wykorzystywane podczas projektowania zwinnego w XPM ?

Diagramy klas (?)

Kolokwium 2018

Uzasadnij czy w projektach wykonywanych zgodnie z metodyką XPM można oceniać wydajność pracy programisty na podstawie ilości zrealizowanych przez niego zadań (tasków) rejestrowanych na task board.

Raczej nie, pracuje się w parach więc trudno o jednoznaczne określenie ile kto zrobił.