**Nazwa przedmiotu:**

Zaawansowane metody wytwarzania oprogramowania

**Koordynator przedmiotu:**

dr inż. Ilona Bluemke, dr inż. Anna Derezinska

**Status przedmiotu:**

Fakultatywny ograniczonego wyboru

**Poziom kształcenia:**

Studia II stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Przedmioty techniczne - zaawansowane

**Kod przedmiotu:**

ZMWO

**Semestr nominalny:**

3 / rok ak. 2015/2016

**Liczba punktów ECTS:**

4

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

Bilans nakładu pracy przeciętnego studenta wygląda następująco:
-udział w wykładach: 15 x 2 godz. = 30 godz.,
-przygotowanie do kolejnych wykładów i realizacji laboratorium (przejrzenie materiałów z wykładu i dodatkowej literatury, próba rozwiązania miniproblemów sformułowanych na wykładzie, konsultacje): 10 godzin (w tym 1 h konsultacji)
-realizacja zadań laboratoryjnych: (15h zajęć laboratoryjnych, oraz obejmuje także zapoznanie się z instrukcjami laboratoryjnymi, przygotowanie programów i ich testów oraz modeli przed laboratoriami, przygotowanie rozwiązań zadań, zainstalowanie części oprogramowania i opanowanie umiejętności wykorzystania go przy realizacji zadań, dokończenie zadań po laboratorium oraz przygotowanie kolejnych sprawozdań), 15 + (3\*15+10) = 70
-przygotowanie do 2 kolokwiów i 2 kartkówek (rozwiązanie zadań przedkolokwialnych, udział w konsultacjach): 10 godz. (w tym 1 godz. konsultacji)
Łączny nakład pracy studenta wynosi zatem: 30 + 10 + 70 + 10 = 120 godz., co odpowiada ok. 4 punktom ECTS.

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

30 + 15 + 1 + 1 = 47 godz., co odpowiada ok. 2 punktom ECTS (zakładam oprócz wykładu i laboratorium 2 godziny konsultacji w semestrze)

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

nakład pracy związany z zajęciami o charakterze praktycznym wynosi 70 godzin co odpowiada ok. 2.5 punktom ECTS.

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład: | 30h |
| Ćwiczenia: | 0h |
| Laboratorium: | 15h |
| Projekt: | 0h |
| Lekcje komputerowe: | 0h |

**Wymagania wstępne:**

Przedmiot IOP, umiejętność programowania w Javie

**Limit liczby studentów:**

24

**Cel przedmiotu:**

Celem przedmiotu jest zapoznanie studentów z zaawansowanymi metodami stosowanymi w produkcji oprogramowania wysokiej jakości oraz jego ewolucji. Omawiane są różne formalne metody specyfikacji oprogramowania pozwalające na uzyskanie wysokiej niezawodności. Prezentowane są nowoczesne podejścia do projektowania np. MDA, design by contract, metodyki lekkie. Omawiane są różne metody ponownego użycia np. projektowanie aspektowe, komponentowa inżynieria oprogramowania, wzorce projektowe i architektoniczne. Studenci poznają także zaawansowane techniki testowania oprogramowania obiektowego. Uczą się stosowania metryk oprogramowania w projektowaniu i testowaniu. Przedstawiane są techniki ponownej inżynierii kodu poprawiające jego jakość. Wykład jest skorelowany z zajęciami laboratoryjnymi. W ramach zajęć laboratoryjnych studenci stosują poznane metody do rozwiązania konkretnych problemów. Poznają także narzędzia do specyfikacji formalnych i automatyzacji testowania.

**Treści kształcenia:**

Treść wykładu:
Wprowadzenie (2)
Przedstawienie celu i zakresu przedmiotu oraz sprawy organizacji zajęć (zaliczenie, laboratorium). Cechy oprogramowania wysokiej jakości.
Formalne metody specyfikacji i weryfikacji oprogramowania (4)
Język Z - podstawowe mechanizmy, operatory, przykłady specyfikacji, zastosowania. Modele zachowania (np. statecharts) i ich zastosowanie w weryfikacji własności czasowych i niezawodnościowych. Narzędzia do specyfikacji i weryfikacji. Transformacje specyfikacji formalnych do kodu i modeli.
Nowe podejścia do projektowania oprogramowania (6)
MDE (Model Driven Engineering) - wytwarzanie oparte na modelach. Meta-modelowanie, hierarchia OMG. Profile UML. MDA (Model Driven Architecture) - model niezależny od platformy (PIM) i model zależny platformowo (PSM). Transformacje modeli (standard QVT - Query View Transformation), łączenie modeli.
CBSE - komponentowa inżynieria oprogramowania, modele komponentowe.
Testowanie (3)
Testowanie a specyfikacje formalne. Test Driven Development. Automatyzacja testowania. Testowanie za pomocą narzędzi typu „zarejestruj – odtwórz”, skrypty testujące, punkty weryfikacyjne. Testy jednostkowe. Testy z użyciem obiektów zastępczych (mock objects).
Nowoczesne metody wytwarzania i ewolucji oprogramowania (10)
Wytwarzanie zorientowane aspektowo (AOD). Identyfikacja aspektów i ich przeplatanie (ang.weaving). Języki programowania aspektowego (C++ i Java) i ich kompilatory.
Wzorce architektoniczne - przykłady zastosowań.
Slicing kodu - odmiany slicingu, zastosowania
Refaktoryzacja kodu - usuwanie "brzydkich zapachów" np. duplikaty kodu, długie metody, duże klasy, złożone instrukcje warunkowe. Przekształcenia refaktoryzacyjne. Weryfikacja poprawności refaktoryzacji - projektowanie zgodne z umową (ang. design by contract) warunki wstępne i końcowe, niezmienniki, testowanie jednostkowe, regresyjne. Refaktoryzacja do wzorców projektowych.
Metryki oprogramowania (2)
Metryki obiektowe wpływające na jakość i złożoność projektu oraz procesu testowania. Metryki Chindamber-Kemerer, MOOD, Bashir-Goehl.
Zakres laboratorium:
Celem laboratorium jest pogłębienie zdobytej wiedzy i jej praktyczne zastosowanie. Laboratorium obejmuje zadania z zakresu formalnych specyfikacji, testowania, programowania aspektowego, refaktoryzacji kodu. Studenci będą wykorzystywać specjalizowane narzędzia. Szczegółowa treść ćwiczeń laboratoryjnych jest określana przez prowadzących i może być różna dla różnych zespołów laboratoryjnych.

**Metody oceny:**

kolokwia, oceny za ćwiczenia laboratoryjne

**Egzamin:**

nie

**Literatura:**

M. Fowler i inni: Refaktoryzacje. Ulepszanie struktury istniejącego kodu, WNT 2006
· S.H. Kan: Metryki i modele w inżynierii jakości oprogramowania, PWN 2006
· R.V. Binder: Testowanie systemów obiektowych. Modele, wzorce, narzędzia, WNT 2003
· A. Harry: Formal Methods Fact File VDM and Z, Wiley 1996
· Aspect-Oriented Software Development web site - http://www.aosd.net
· J. Kerievsky, Refaktoryzacja do wzorców projektowych, Helion, 2005
· Specyfikacje OMG (MDA, MOF, UML, QVT) http://www.omg.org/
Literatura uzupełniająca:
· L. Maciaszek : Practical Software Engineering, Pearson 2005
· M. Flower i inni:. Architektura systemów zarządzania przedsiębiorstwem, Wzorce
projektowe, Helion 2005
· J.Z.Gao, H.-S.J. Tsao, Ye Wu: Testing and quality Assurance for Component Based
Software, Artech House 2003
· C. Szyperski: Oprogramowanie komponentowe. Obiekty to za mało, WNT 2001
· B. Wiszniewski, B. Bereza-Jarociński: Teoria i praktyka testowania programów, PWN 2006
· G. Kiczales and others: Aspect-Oriented Programming, Proceedings of the European
Conference on Object-Oriented Programming (ECOOP), Finland, LNCS 1241
· http://www.mockobjects.com/
· Materiały wykładowcy i strony internetowe przedmiotu

**Witryna www przedmiotu:**

https://studia.elka.pw.edu.pl/priv/12L/ZMWO.A/

**Uwagi:**

Uwagi realizacyjne:
Do zajęć laboratoryjnych są potrzebne narzędzia wspomagające testowanie i specyfikacje formalne, kompilatory języków aspektowych, narzędzia do refaktoryzacji kodu, narzędzia CASE do modelowania i transformacji modeli. Podczas wykładu konieczny jest rzutnik i notebook z zainstalowanym oprogramowaniem do pokazu narzędzi.

## Efekty przedmiotowe

### Profil ogólnoakademicki - wiedza

**Efekt zmwo-W01:**

Student, który zaliczył przedmiot potrafi wskazać echy oprogramowania wysokiej jakości

Weryfikacja:

kolokwium

**Powiązane efekty kierunkowe:** K\_W12, K\_W05

**Powiązane efekty obszarowe:** T2A\_W08, T2A\_W04

**Efekt zmwo-W02:**

Student, który zaliczył przedmiot potrafi przeprowadzić analizę kodu programu z wykorzystaniem metryk oprogramowania i statycznych analizatorów kodu, wykorzystać wyniki analizy w ewolucji oprogramowania

Weryfikacja:

Kolokwium, zadanie laboratoryjne

**Powiązane efekty kierunkowe:** K\_W05, K\_W07, K\_W08

**Powiązane efekty obszarowe:** T2A\_W04, T2A\_W05, T2A\_W07

**Efekt zmwo-W03:**

student, który zaliczył przedmiot, potrafi wykonać refaktoryzacje programu obiektowego oraz sprawdzić poprawność refaktoryzacji programu metodami testowania jednostkowego i funkcjonalnego z wykorzystaniem odpowiednich narzędzi wytwarzania oprogramowania

Weryfikacja:

kolokwium, zadanie laboratoryjne

**Powiązane efekty kierunkowe:** K\_W05, K\_W07, K\_W08

**Powiązane efekty obszarowe:** T2A\_W04, T2A\_W05, T2A\_W07

**Efekt zmwo-W04:**

student, który zaliczył przedmiot, potrafi określić i wykorzystać do analizy programu graf zależności programu oraz wycinek kodu (ang. slice)

Weryfikacja:

kolokwium

**Powiązane efekty kierunkowe:** K\_W05, K\_W08

**Powiązane efekty obszarowe:** T2A\_W04, T2A\_W07

**Efekt zmwo-W05:**

student, który zaliczył przedmiot, potrafi stosować meta-modelowanie, ocenić zgodność modelu z metamodelem, zaprojektować i zmodyfikować proste meta-modele z wykorzystaniem odpowiednich narzędzi CASE

Weryfikacja:

kartkówka, kolokwium, zadanie laboratoryjne

**Powiązane efekty kierunkowe:** K\_W02, K\_W05, K\_W07, K\_W08

**Powiązane efekty obszarowe:** T2A\_W03, T2A\_W04, T2A\_W05, T2A\_W07

**Efekt zmwo-W06:**

Student, który zaliczył przedmiot potrafi zaprojektować i wykonać proste transformacje modeli w zadanym języku transformacji z wykorzystaniem odpowiednich narzędzi CASE

Weryfikacja:

kolokwium, zadanie laboratoryjne

**Powiązane efekty kierunkowe:** K\_W02, K\_W05, K\_W07, K\_W08

**Powiązane efekty obszarowe:** T2A\_W03, T2A\_W04, T2A\_W05, T2A\_W07

**Efekt zmwo-W07:**

student, który zaliczył przedmiot, potrafi napisać prostą specyfikację w notacji Z, wskazać typy systemów, które powinny być formalnie specyfikowane

Weryfikacja:

kolokwium

**Powiązane efekty kierunkowe:** K\_W02, K\_W05, K\_W07

**Powiązane efekty obszarowe:** T2A\_W03, T2A\_W04, T2A\_W05

**Efekt zmwo-W08:**

Student, który zaliczył przedmiot potrafi zaprojektować aspekt, wskazać zastosowania programowania aspektowego

Weryfikacja:

kolokwium, kartkówka, zadanie laboratoryjne

**Powiązane efekty kierunkowe:** K\_W05, K\_W07, K\_W08

**Powiązane efekty obszarowe:** T2A\_W04, T2A\_W05, T2A\_W07

### Profil ogólnoakademicki - umiejętności

**Efekt zmwo-U01:**

student, który zaliczył przedmiot, potrafi przeprowadzić analizę kodu programu z wykorzystaniem metryk oprogramowania i statycznych analizatorów kodu, wykorzystać wyniki analizy w ewolucji oprogramowania

Weryfikacja:

kolokwium, zadanie laboartoryjne

**Powiązane efekty kierunkowe:** K\_U05, K\_U09, K\_U10, K\_U11

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09, T2A\_U11, T2A\_U15, T2A\_U16

**Efekt zmwo-U02:**

Student, który zaliczył przedmiiot potrafi wykonać refaktoryzacje programu obiektowego oraz sprawdzić poprawność refaktoryzacji programu metodami testowania jednostkowego i funkcjonalnego z wykorzystaniem odpowiednich narzędzi wytwarzania oprogramowania

Weryfikacja:

kolokwium, laboratorium

**Powiązane efekty kierunkowe:** K\_U05, K\_U08, K\_U10, K\_U11

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09, T2A\_U12, T2A\_U15, T2A\_U16

**Efekt zmwo-U03:**

.... stosować meta-modelowanie, zaprojektować i zmodyfikować proste meta-modele, zaprojektować i wykonać proste transformacje modeli

Weryfikacja:

kolowium, laboratorium

**Powiązane efekty kierunkowe:** K\_U05, K\_U08, K\_U13

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09, T2A\_U12, T2A\_U18

**Efekt zmwo-U04:**

Student, który zaliczył przedmiot potrafi zaprojektować i uruchomić aspekt korzystając z narzędzi do programowania aspektowego

Weryfikacja:

kolokwium, laboratorium

**Powiązane efekty kierunkowe:** K\_U05, K\_U08, K\_U10

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09, T2A\_U12, T2A\_U15

**Efekt zmwo-U05:**

student, który zaliczył przedmiot, potrafi określić i wykorzystać do analizy programu graf zależności programu oraz wycinek kodu (ang. slice)

Weryfikacja:

kolokwoium

**Powiązane efekty kierunkowe:** K\_U05

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09

**Efekt zmwo-U06:**

student, który zaliczył przedmiot, potrafi napisać prostą specyfikację w notacji Z

Weryfikacja:

kolokwium

**Powiązane efekty kierunkowe:** K\_U05

**Powiązane efekty obszarowe:** T2A\_U07, T2A\_U09