**Nazwa przedmiotu:**

Zaawansowane metody wytwarzania oprogramowania

**Koordynator przedmiotu:**

Rajmund Kożuszek

**Status przedmiotu:**

Fakultatywny dowolnego wyboru

**Poziom kształcenia:**

Studia II stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Przedmioty techniczne - podstawowe

**Kod przedmiotu:**

ZMWO

**Semestr nominalny:**

3 / rok ak. 2021/2022

**Liczba punktów ECTS:**

4

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

1. liczba godzin kontaktowych – 49 godz., w tym
obecność na wykładach 26 godz.,
obecność na seminariach 4 godz.
obecność na spotkaniach projektowych 11 godz.
obecność podczas zaliczania projektów 2 godz.,
obecność na warsztatach 4 godz.,
obecność na konsultacjach (do seminarium, projektów, sprawdzianów) 2 godz.,
2. praca własna studenta – 68 godz., w tym
przygotowanie do seminariów 10 godz.,
przygotowanie projektów 40 godz.,
przygotowanie do wykładów i sprawdzianów 18 godz.
Łączny nakład pracy studenta wynosi 117 godz., co odpowiada 4 pkt. ECTS.

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

1,68 pkt. ECTS, co odpowiada 49 godz. kontaktowym

**Język prowadzenia zajęć:**

polski

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

1,95 pkt. ECTS, co odpowiada 2 godz. zaliczania projektów plus 40 godz. przygotowania projektów plus 11 godzin spotkań projektowych plus 4 godz. warsztatów

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

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

**Wymagania wstępne:**

Student zna języki programowania oraz elementarne struktury danych i biblioteki. Student potrafi pisać i uruchamiać programy komputerowe, w tym w co najmniej jednym języku obiektowym. Student posiada podstawową wiedzę i umiejętności z inżynierii oprogramowania, w tym z inżynierii wymagań, modelowania i projektowania oprogramowania, weryfikacji i pielęgnacji oprogramowania.

**Limit liczby studentów:**

24

**Cel przedmiotu:**

Ukształtowanie wśród studentów zrozumienia konieczności zapewnienia wysokiej jakości oprogramowania. Zapoznanie z nowoczesnymi technikami produkcji oprogramowania wysokiej jakości. Rozwój umiejętności w zakresie projektowania i implementacji oprogramowania wysokiej jakości.

**Treści kształcenia:**

Wykład:
Wprowadzenie (2)
Przedstawienie celu i zakresu przedmiotu oraz sprawy organizacji zajęć. Cechy oprogramowania wysokiej jakości. Normy jakości oprogramowania.
Metryki oprogramowania (2)
Metryki obiektowe wpływające na jakość i złożoność projektu oraz procesu testowania. Metryki Chindamber-Kemerer, MOOD, Bashir-Goehl.
Nowoczesne metody wytwarzania i ewolucji oprogramowania (8)
Wytwarzanie zorientowane aspektowo (AOD). Identyfikacja aspektów i ich przeplatanie (ang. weaving). Języki programowania aspektowego (C++ i Java) i ich kompilatory. Slicing kodu - odmiany slicingu, zastosowania. Refaktoryzacja kodu, wykrywanie i 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 kodu do wzorców projektowych.
Testowanie(6)
Test Driven Development. Automatyzacja testowania. Testy jednostkowe. Testy z użyciem obiektów zastępczych (mock objects). Testowanie oparte o wyrocznie i testowanie bez wyroczni. Testowanie mutacyjne, kombinatoryczne. Testowanie interfejsów graficznych i aplikacji internetowych. Testowanie oprogramowania mobilnego, oprogramowania w chmurach, oprogramowania wbudowanego. Kryteria testowania, ocena jakości testów. Testowanie a specyfikacje formalne. Generacja testów z modeli, specyfikacji i kodu.
Nowe podejścia do projektowania oprogramowania (4)
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.
Formalne metody specyfikacji oprogramowania (2)
Podstawowe mechanizmy, operatory, przykłady specyfikacji, zastosowania. Transformacje specyfikacji formalnych do kodu.
Sprawdziany (2)
Projekt:
Programowanie aspektowe
Metryki, Analiza statyczna kodu, Wykrywanie zapachów w kodzie, Refaktoryzacja kodu, Testowanie refaktoryzowanego kodu
Wybrane zagadnienia testowania.
Seminaria:
Wybrane problemy wytwarzania i testowania oprogramowania, nowoczesne metody i narzędzia.
Warsztaty – zajęcia zintegrowane
MDE (Model Driven Engineering) – tworzenie metamodelu, budowa modeli zgodnych z metamodelami, transformacje modeli, testowanie transformacji

**Metody oceny:**

Liczebność zespołów na zajęciach projektowych: 8-12 studentów.
Warsztaty: 2 spotkania po 2 godziny, maksymalnie 16 osób na zajęciach
Seminaria: 2 spotkania po 2 godziny, 10-16 studentów w grupie

**Egzamin:**

nie

**Literatura:**

1. M. Fowler i inni: Refaktoryzacje. Ulepszanie struktury istniejącego kodu, WNT 2006
2. S.H. Kan: Metryki i modele w inżynierii jakości oprogramowania, PWN 2006
3. A. Roman: Testowanie i jakość oprogramowania. Modele, techniki, narzędzia, PWN 2017
4. A. Harry: Formal Methods Fact File VDM and Z, Wiley 1996
5. Aspect-Oriented Software Development web site - http://www.aosd.net
6. J. Kerievsky: Refaktoryzacja do wzorców projektowych, Helion, 2005
7. Specyfikacje OMG (MDA, MOF, UML, QVT) http://www.omg.org/
8. Materiały wykładowcy, aktualne publikacje i materiały z Internetu
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
• 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/
• R.V. Binder: Testowanie systemów obiektowych. Modele, wzorce, narzędzia, WNT 2003

**Witryna www przedmiotu:**

https://usosweb.usos.pw.edu.pl/kontroler.php?\_action=katalog2/przedmioty/pokazPrzedmiot&prz\_kod=103A-INxxx-MSP-ZMWO

**Uwagi:**

(-)

## Charakterystyki przedmiotowe