KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Ewolucja i pielęgnacja oprogramowania

Przedmiot

Kierunek studiów
Informatyka

Rok/semestr
1/1

Studia w zakresie (specjalność)
Software Engineering

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład	Laboratoria	Inne (np. online)
30	30	
Ćwiczenia	Projekty/seminaria	

Liczba punktów ECTS

6

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:
dr inż. Bartosz Walter

Odpowiedzialny za przedmiot/wykładowca:

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać wiedzę dotyczącą modeli i procesów rozwoju oprogramowania, oraz podstawowe umiejętności w zakresie programowania (przynajmniej dotyczące czytania i rozumienia kodu). Powinien również posiadać umiejętność pozyskiwania informacji ze wskazanych źródeł oraz mieć gotowość do podjęcia współpracy w ramach zespołu.

Cel przedmiotu

Celem tego predmiotu jest przekazanie studentom wiedzy dotyczącej procesów ewolucji systemów informatycznych, rodzajów zmian związanych z pielęgnacją oraz możliwej reakcji na ewolucję poprzez zaplanowane i świadome czynności pielęgnacyjne. Studenci po ukończeniu przedmiotu powinny umieć ocenić pielęgnowalność systemu informatycznego, zastosować w nim zmiany oraz zweryfiować ich poprawność, a także uczestniczyć w przeglądach oprogramowania i stosować przekształcenia refaktoryzacyjne.

Przedmiotowe efekty uczenia się

Wiedza

1. Posiada ugruntowaną wiedzę teoretyczną dotyczącą cyklu rozwojowego systemu informatycznego

2. Posiada wiedzę na temat wybranych metod, języków i notacji modelowania oprogramowania

3. Posiada wiedzę na temat wzorców projektowych oraz dobrych praktyk w zakresie projektowania oprogramowania

4. Zna wybrane wybrane metryki i metody pomiaru niektórych charakterystyk oprogramowania (np. rozmiaru, złożoności)

Umiejętności

1. Potrafi przeprojektować system informatyczny, zastosować w nim poprawkę lub zaktualizować go.

2. Potrafi dokonać oceny jakości projektu systemu informatycznego i ocenić jej wpływ na koszt i łatwość jego funkcjonowania i rozwoju.

Kompetencje społeczne

1. Potrafi współpracować w grupie

2. Potrafi poszerzać swoją wiedzę, korzystając z dostępnych źródeł i dokonując ich selekcji

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Wiedza przedstawiona w ramach wykładu weryfikowana jest poprzez realizację podczas wykładu, w ramach współpracy w grupach, dwóch zadań projektowych oraz egzamin końcowy -- test wielokrotnego wyboru sprawdzający stopień zrozumienia i przyswojenia treści wykładowych. Oceny uzyskane z tych form są uśredniane z wagami 30% i 70%. Próg zaliczeniowy wynosi 50%. Zagadnienia zaliczeniowe zostaną przedstawione podczas ostatniego wykładu w ramach przedmiotu.

Umiejętności nabyte w ramach laboratorium weryfikowane są poprzez realizację w grupach 3-4 projektów, dotyczących poszczególnych zagadnień omawianych w trakcie zajęć. Próg zaliczeniowy wynosi 50%.

Treści programowe

1. Wykład: Przegląd modeli ewolucji oprogramowania. Pomiary i metryki dotyczące ewolucji artefaktów programowych. Rodzaje czynności pielęgnacyjnych. Podejścia do oceny pielęgnowalności. Metody restrukturyzacji i refaktoryzacji zastanego oprogramowania. Obserwacja i analiza zmian w repozytoriach kodu.

2. Laboratorium: Obserwacja ewolucji oprogramowania. Zbieranie i analiza wartości metryki dotyczących ewolucji oprogramowania. Zaburzenia w pielęgnacji oprogramowania. Pielęgnacja systemów informatycznych w interacyjnym cyklu życia oprogramowania. Techniki refaktoryzacyjne.

Metody dydaktyczne

.

Literatura

Podstawowa

1. T. Mens, S. Demeyer: Software Evolution. Springer Science and Business Media, 2008.

2. R. C. Martin: Clean code. A Handbook of agile software craftmanship. Prentice Hall, 2008

3. J. Visser i in.: Oprogramowanie łatwe w utrzymaniu. Edycja Java. Helion, 2016.

Uzupełniająca

1. M. Fowler. Refaktoryzacja. Ulepszanie struktury istniejącego kodu. Helion, 2019.

2. Pryadershi Tripathy, Kshirasagar Naik: Software evolution and maintenance. A practitioner's approach. Addison-Wesley, 2015.

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	150	6,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	61	2,0
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)[1]	89	4,0

[1] niepotrzebne skreślić lub dopisać inne czynności