



KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Ewolucja i pielęgnacja oprogramowania [S2Inf1E-IO>EPO]

Przedmiot

Kierunek studiów

Informatyka/Computing

Rok/Semestr

1/2

Studia w zakresie (specjalność)

Inżynieria oprogramowania

Profil studiów

ogólnoakademicki

Poziom studiów

drugiego stopnia

Język oferowanego przedmiotu

angielski

Forma studiów

stacjonarne

Wymagalność

obligatoryjny

Liczba godzin

Wykład

30

Laboratorium

30

Inne

0

Ćwiczenia

0

Projekty/seminaria

15

Liczba punktów ECTS

6,00

Koordynatorzy

dr hab. inż. Bartosz Walter prof. PP
bartosz.walter@put.poznan.pl

Wykładowcy

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 przedmiotu 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 zweryfikować 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:

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 interakcyjnym cyklu życia oprogramowania. Techniki refaktoryzacyjne.

Tematyka zajęć

1. Wykład: Przegląd modeli ewolucji oprogramowania. Pomiary i metryki dotyczące ewolucji artefaktów programowych. Rodzaje czynności pielęgnacyjnych. Podejścia do oceny pielęgnowalności za pomocą przykrych zapachów. Klony kodu. Metody refaktoryzacji istniejącego oprogramowania. Obserwacja i analiza zmian w repozytoriach kodu za pomocą Google BigQuery oraz Github API. Model pielęgnowalności oprogramowania SIG. Czysty kod - przegląd i przykłady.
2. Laboratorium: Obserwacja i analiza ewolucji oprogramowania. Analiza wartości wybranych metryk dotyczących ewolucji oprogramowania. Zaburzenia w pielęgnacji oprogramowania. Pielęgnacja systemów informatycznych w iteracyjnym cyklu życia oprogramowania. Wykonywanie przekształceń refaktoryzacyjnych z wykorzystaniem narzędzi oraz ręcznie. Analiza przykrych zapachów (code smells) w celu priorytetyzacji działań pielęgnacyjnych
3. Projekt: aktywny udział w wybranym projekcie open source

Metody dydaktyczne

1. Wykład: prezentacja multimedialna, dyskusja
2. Zajęcia laboratoryjne: prezentacja poparta podanymi przykładami, programowanie oprogramowania i zadania projektowe w grupach, dyskusja

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 craftsmanship. 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	6,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	75	3,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwίων/egzaminu, wykonanie projektu)	75	3,00