KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Nazwa przedmiotu

Inżynieria oprogramowania

Kod ECTS

11.3.1363

Nazwa jednostki prowadzącej przedmiot

Instytut Informatyki

Studia

wydział	kierunek		
Wydział Matematyki, Fizyki i Informatyki	Informatyka	poziom	pierwszego stopnia
		forma	stacjonarne
		moduł specjalnościowy	wszystkie
		specjalizacja	wszystkie

Nazwisko osoby prowadzącej (osób prowadzących)

dr Adam Kostulak; dr Paweł Pączkowski

Formy zajęć, sposób ich realizacji i przypisana im liczba godzin

Formy zajęć

Wykład, Ćw. laboratoryjne

Sposób realizacji zajęć

zajęcia w sali dydaktycznej

Liczba godzin

Wykład: 15 godz., Ćw. laboratoryjne: 15 godz.

Liczba punktów ECTS

2
Przedmiot w wymiarze 15h wykładu i 15h laboratorium + praca własna studenta.

Termin realizacji przedmiotu

2020/2021 letni

Status przedmiotu

obowiązkowy

Język wykładowy

polski

Metody dydaktyczne

- Dyskusja
- Rozwiązywanie zadań
- Samodzielne wykonanie miniprojektów
- Wykład z prezentacją multimedialną

Forma i sposób zaliczenia oraz podstawowe kryteria oceny lub wymagania egzaminacyjne

Sposób zaliczenia

- Zaliczenie na ocenę
- Zaliczenie (zal)

Formy zaliczenia

- ustalenie oceny zaliczeniowej na podstawie ocen cząstkowych otrzymywanych w trakcie trwania semestru
- kolokwium

Podstawowe kryteria oceny

Przedmiot kończy się pisemnym kolokwium zaliczeniowym, zaliczenie od 51% punktów. Przed przystąpienieniem do kolokwium konieczność zaliczenia ćwiczeń laboratoryjnych. Ocena z ćwiczeń na podstawie wyników z prac domowych oraz aktywności na zajęciach.

Sposób weryfikacji założonych efektów kształcenia

zakładany efekt kształcenia	egzamin	kolokwium	projekt	referat	raport	aktywność w dyskusji	obserwacja postawy studenta
	Wiedza						
K_W04		X					
	Umiejętności						
K_U05							X
K_U07							X
	Kompetencje						

Określenie przedmiotów wprowadzających wraz z wymogami wstępnymi

A. Wymagania formalne

Brak

B. Wymagania wstępne

Języki programowania, Programowanie obiektowe

Cele kształcenia

Celem przedmiotu jest zapoznanie studentów z podstawowymi problemami, metodami, technikami i narzędziami produkcji oprogramowania wysokiej jakości. Przedstawienie różnych modeli cyklu życia oprogramowania, metod wytwarzania oprogramowania, podstaw dokumentacji, analizy wymagań, projektowania, testowania oprogramowania. Omówienie zarządzania analizą wymagań, tworzenia specyfikacji ze zdefiniowanymi metrykami. Omawiane zagadnienia przedstawiane z naciskiem na obiektowe podejście do projektowania i modelowanie systemu za pomocą języka UML.

Treści programowe

1. Wprowadzenie; motywacje dla systematycznego wytwarzania oprogramowania.
2. Cykl życia oprogramowania, fazy cyklu klasycznego, miejsce analizy i projektowania w cyklu wytwarzania.
3. Wymagania. Kategorie, pozyskiwanie, analiza i specyfikacja wymagań. Weryfikacja i walidacja; testowanie akceptacyjne.
4. Modelowanie; modelowanie świata a modelowanie systemu, miejsce modelu w analizie i projektowaniu.
5. Przypadki użycia. Diagram, interpretacja, opis ustrukturalizowany, przykłady.
6. Przegląd podstawowych zasad i pojęć paradygmatu obiektowego: obiekt, klasyfikacja, agregacja, dziedzicznie, komunikacja. Obiektowość a naturalny dla człowieka sposób postrzegania rzeczywistości.
7. Wstęp do metodyki UML: model klas, model dynamiczny. Związki pomiędzy modelami.
8. Obiekty, klasy, atrybuty, operacje. Związki pomiędzy klasami (asocjacje). Związki jako klasy. Role. Agregacja jako szczególny przypadek związku. Semantyka agregacji. Propagacja operacji jako kryterium agregacji. Przykłady.
9. Dziedziczenie: specjalizacja i generalizacja. Hierarchia dziedziczenia. Redefiniowanie właściwości obiektów w dół hierarchii dziedziczenia, przesłanianie Klasy abstrakcyjne. Obiekt jako wystąpienie klasy i wszystkich jej nadklas.
10. Jak tworzyć model klas? Przykład tworzenia modelu klas.
11. Model dynamiczny: zdarzenia, czynności, akcje. Diagramy sekwencji (interakcji). Powiązanie z m innymi modelami. Jak tworzyć diagram sekwencji. Przykład.
12. Diagram stanów. Stany: proste, złożone, współbieżne. dziedziczenie stanów. Przejścia między stanami. Powiązanie z modelem klas. Jak tworzyć diagram model dynamiczny? Przykład tworzenia modelu dynamicznego.
13. Przejście od modelu do projektu. Podstawowe fazy projektu obiektowego. Podział na podsystemy i moduły, identyfikacja współbieżności, przydział procesorów, przydział zasobów, priorytety.
14. Przejście od projektu do programu.

Wykaz literatury

Materiały wykładów (umieszczone na Portalu Edukacyjnym)
I. Sommerville: Inżynieria oprogramowania, WNT 2003
Szejko St. (red). Metody wytwarzania oprogramowania. MIKOM, 2002.
Jaszkiewicz A.: Inżynieria oprogramowania. Helion, 1997.
Uzupełniająca:
Booch G., Rumbaugh J., Jacobson I.: UML podręcznk użytkownika. WNT, 2001
Dumnicki R., Kasprzyk A., Kozłowski M.: Analiza i projektowanie obiektowe. Helion, 1998
Eriksson H-E, Penker M.: UML Toolkit. Wiley Computer Publishing, John Wiley & Sons, Inc. 1998.
Pressman R.S.: Software Engineering. A Practitioner's Approach. McGraw-Hill, Inc. 1992.

Kierunkowe efekty kształcenia

K_W04: ma uporządkowaną wiedzę w zakresie inżynierii oprogramowania, specyfikacji, walidacji i weryfikacji oprogramowania oraz narzędzi wspomagających proces wytwarzania oprogramowania
K_U05: potrafi pracować w zespole informatyków, zarządzać swoim czasem oraz podejmować zobowiązania i dotrzymywać terminy, porozumiewać się przy użyciu różnych technik w tym z wykorzystaniem dedykowanych narzędzi
K_U07: potrafi projektować, tworzyć, uruchamiać i testować programy przy wykorzystaniu dedykowanych narzędzi oraz adekwatnych wzorców

Wiedza

Student:

- zna różne modele cyklu życia oprogramowania;
- zna metody wytwarzania oprogramowania;
- posiada wiedzę odnośnie podstaw wytwrzania dokumentacji, analizy wymagań, projektowania, testowania oprogramowania.

Umiejętności

Student:

- potrafi zarządzać analizą wymagań;
- potrafi tworzyć specyfikację ze zdefiniowanymi metrykami;
- potrafi utworzyć projekt oprogramowania za pomocą języka UML.

Kompetencje społeczne (postawy)

Student:

- posiada podstawowe umiejętności komunikacji w zespole informatycznym

Kontakt

adam.kostulak@ug.edu.pl