KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Nazwa przedmiotu	Kod ECTS
Podstawy inżynierii oprogramowania	11.3.0737

Nazwa jednostki prowadzącej przedmiot

Faculty of Mathematics, Physics and Informatics

Studia

wydział	kierunek		
Wydział Matematyki, Fizyki i Informatyki	Informatyka	poziom	pierwszego stopnia
		forma	niestacjonarne (zaoczne)
		moduł specjalnościowy	wszystkie
		specjalizacja	wszystkie

Nazwisko osoby prowadzącej (osób prowadzących)

dr Robert Fidytek; dr Magdalena Godlewska

Formy zajęć, sposób ich realizacji i przypisana im liczba godzin

Formy zajęć

Wykład, Ćw. laboratoryjne

Sposób realizacji zajęć

zajęcia on-line, zajęcia w sali dydaktycznej

Liczba godzin

Ćw. laboratoryjne: 10 godz., Wykład: 10 godz.

Liczba punktów ECTS

2
Przedmiot w wymiarze 10h wykładu i 10h laboratorium + praca własna studenta

Cykl dydaktyczny

2016/2017 letni

Status przedmiotu

obowiązkowy

Język wykładowy

polski

Metody dydaktyczne

- Dyskusja
- Praca w grupach
- Rozwiązywanie zadań
- 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
- egzamin pisemny (dłuższa wypowiedź pisemna / rozwiązanie problemu)

Podstawowe kryteria oceny

Laboratorium:
Ocena zależy od sumy punktów zdobytych za zrealizowane zadania.
Wykład:
Wynik zaliczenia laboratorium - 50% oceny.
Egzamin pisemny lub pisemne zadanie zaliczeniowe - 50% oceny.

Sposób weryfikacji założonych efektów kształcenia

zakładany efekt kształcenia	egzamin pisemny/zadanie zaliczeniowe	zadania indywidualne	zadania wykonywane w grupie	obserwacja
	Wiedza			
K_W02	x			
K_W03	x			
K_W09	x			
K_W12				x
	Umiejętności			
K_U03		x	x	
K_U04			x	
K_U06		x	x	
K_U19			x	
K_U22			x	
	Kompetencje			
K_K01				x

Określenie przedmiotów wprowadzających wraz z wymogami wstępnymi

A. Wymagania formalne

Aktywny udział w zajęciach.

B. Wymagania wstępne

Języki programowania, algorytmy i struktury danych, bazy danych.

Cele kształcenia

Celem przedmiotu jest zapoznanie studentów z podstawowymi problemami, metodami, technikami i narzędziami produkcji oprogramowania wysokiej jakości. Nacisk jest położony na obiektowe podejście do projektowania i modelowanie systemu w UML.

Treści programowe

1. Proces tworzenia Systemów Informatycznych
2. Pojęcie procesu i projektu (budowy) oprogramowania. Modele cyklu życia oprogramowania (model kaskadowy, realizacja kierowana dokumentami, V-model, prototypowanie, model przyrostowy, spiralny)
3. Określanie wymagań i analiza. Rodzaje i role notacji wykorzystywanych w fazie analizy, obiektowe i strukturalne metody analizy.
4. UML 2. – standard modelowania obiektowego: poziomy modelu i typy diagramów. Przykłady użycia (use case)- elementy diagramów, organizowanie. Model logiczny - diagramy klas, diagramy behaweoralne (sekwencji, stanów, aktywności), Model implementacyjny – diagram komponentów (komponenty, podsystemy, porty) i model fizyczny (diagram montażowy).
5. Testowanie i inspekcja. Testowanie ukierunkowane na wyszukiwanie defektów – funkcjonalne, strukturalne (złożoność cyklomatyczna McCabe, pokrycie kodu - ścieżki niezależne, pokrycie danych, testowanie pętli). Testowanie obiektów. Estymacja kosztów. Metoda punktów funkcyjnych, model COCOMO 81 i COCOMO II. Model CMM. Miary niezawodności oprogramowania i techniki programowania dla systemów o dużej niezawodności.

Wykaz literatury

1. I. Sommerville: Inżynieria oprogramowania, WNT 2003
2. Szejko St. (red). Metody wytwarzania oprogramowania. MIKOM, 2002.
3. Jaszkiewicz A.: Inżynieria oprogramowania. Helion, 1997.
4. Materiały umieszczone na platformie udukacyjnej.

Uzupełniająca:

5. Booch G., Rumbaugh J., Jacobson I.: UML podręcznk użytkownika. WNT, 2001
6. Dumnicki R., Kasprzyk A., Kozłowski M.: Analiza i projektowanie obiektowe. Helion, 1998
7. Eriksson H-E, Penker M.: UML Toolkit. Wiley Computer Publishing, John Wiley & Sons, Inc. 1998.
8. Pressman R.S.: Software Engineering. A Practitioner's Approach. McGraw-Hill, Inc. 1992.

Efekty kształcenia (obszarowe i kierunkowe)	Wiedza
K_W02 ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie programowania, algorytmów i	Srudent ma uporządkowaną wiedzę dotycząca metryk procesu tworzenia oprogramowania.

złożoności, architektury systemów komputerowych, systemów operacyjnych, technologii sieciowych, języków i paradygmatów programowania, baz danych, inżynierii oprogramowania, języków formalnych,
K_W03: zna podstawowe metody projektowania, analizowania i programowania algorytmów,
K_W09: ma wiedzę na temat inżynierii oprogramowania, projektowania, wzorców projektowych, wykorzystania API, narzędzi i środowisk wytwarzania oprogramowania, cyklu życia projektu informatycznego, specyfikacji oprogramowania, walidacji i weryfikacji, utrzymywania oprogramowania,
K_W12: zna podstawowe zasady bezpieczeństwa i higieny pracy w zawodzie informatyka
K_U03 potrafi pracować indywidualnie i w zespole informatyków, w tym także potrafi zarządzać swoim czasem oraz podejmować zobowiązania i dotrzymywać terminów,
K_U04 potrafi porozumiewać się przy użyciu różnych technik w środowisku zawodowym oraz w innych środowiskach, w tym w języku angielskim oraz z wykorzystaniem narzędzi informatycznych,
K_U06 projektuje, analizuje pod kątem poprawności i złożoności obliczeniowej oraz programuje algorytmy; wykorzystuje podstawowe techniki algorytmiczne i struktur danych,
K_U19 tworzy, ocenia i realizuje plan testowania,
K_U22 posługuje się wzorcami projektowymi
K_K01 zna ograniczenia własnej wiedzy i rozumie potrzebę dalszego kształcenia
K_K06 potrafi formułować opinie na temat podstawowych zagadnień informatycznych

Student zna techniki i narzędzia do produkcji oprogramowania wysokiej jakości.
Student zna podstawy tworzenia dokumentacji projektów informatycznych.

Umiejętności

Student umie tworzyć dokumentacę projektów informatycznych.
Student umie projektować systemy informatyczne za pomocą języka UML.
Student umie wykorzystać narzędzia do produkcji oprogramowania wysokiej jakości.

Kompetencje społeczne (postawy)

Student zna ograniczenia własnej wiedzy i rozumie potrzebę dalszego kształcenia.
Student potrafi formułować własne opinie na temat projektowanych systemów.

Kontakt

robert.fidytek@inf.ug.edu.pl