


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

 Projekt współfinansowany przez
 Unię Europejską w ramach
 Europejskiego Funduszu
 Społecznego

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


Nazwa przedmiotu		Kod ECTS	
Inżynieria oprogramowania		11.3.2079	
Nazwa jednostki prowadzącej przedmiot			
Instytut Informatyki			
Studia			
wydział	kierunek	poziom	pierwszego stopnia
Wydział Matematyki, Fizyki i Informatyki	Informatyka	forma	stacjonarne
		moduł	wszystkie
		specjalnościowy	wszystkie
		specjalizacja	wszystkie
Nazwisko osoby prowadzącej (osób prowadzących)			
prof. UG, dr hab. Viktoriia Onyshchenko; mgr Łukasz Mielewczyk; mgr Michał Zakrzewski			
Formy zajęć, sposób ich realizacji i przypisana im liczba godzin		Liczba punktów ECTS	
Formy zajęć		3 Udział w zajęciach dydaktycznych, objętych planem studiów: 30h Praca własna studenta: 45h RAZEM: 75h	
Wykład, Ćw. laboratoryjne			
Sposób realizacji zajęć			
zajęcia w sali dydaktycznej			
Liczba godzin			
Wykład: 15 godz., Ćw. laboratoryjne: 15 godz.			
Termin realizacji przedmiotu			
2024/2025 letni			
Status przedmiotu		Język wykładowy	
obowiązkowy		polski	
Metody dydaktyczne		Forma i sposób zaliczenia oraz podstawowe kryteria oceny lub wymagania egzaminacyjne	
<ul style="list-style-type: none"> - Dyskusja - Gry symulacyjne - Praca w grupach - Projektowanie doświadczeń - Rozwiązywanie zadań - Samodzielne wykonanie miniprojektów - Wykład z prezentacją multimedialną 		Sposób zaliczenia	
		Zaliczenie na ocenę	
		Formy zaliczenia	
		<ul style="list-style-type: none"> - wykonanie pracy zaliczeniowej - projekt lub prezentacja - 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.	
			końcowej
		kolokwium	50%
		50%	40%
		50%	40%
		0%	5%
		0%	5%
		0%	5%
Sposób weryfikacji założonych efektów uczenia się			

zakładany efekt kształcenia	egzamin	kolokwium	projekt	referat	raport	aktywność w dyskusji	obserwacja postawy studenta
Wiedza							
K_W04		X	X				
P_W01			X				
P_W02			X				
P_W03			X				
Umiejętności							
K_U05			X				X
K_U07			X				X
P_U01			X				X
P_U02			X				X
P_U03			X				X
Kompetencje							
K_K02			X				X
							P_K01
		X					X

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, dziedziczenie, 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 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.

Jaskiewicz A.: Inżynieria oprogramowania. Helion, 1997.

Uzupełniająca:

Booch G., Rumbaugh J., Jacobson I.: UML podręcznik 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 uczenia się

K_W04: ma uporządkowaną wiedzę w zakresie inżynierii oprogramowania, specyfikacji, walidacji i weryfikacji oprogramowania oraz narzędzi wspomagających proces wytwarzania oprogramowania

K_U04: 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_U06: potrafi projektować, tworzyć, uruchamiać i testować programy przy wykorzystaniu dedykowanych narzędzi oraz adekwatnych wzorców

K_K02: rozumie potrzebę i docenia zalety pracy zespołowej, rozumie konieczność systematycznej pracy nad zespołowymi projektami informatycznymi, jest gotów do aktywnego działania w zespole

Wiedza

P_W01: Student zna różne modele cyklu życia oprogramowania; (K_W04)

P_W02: Student zna metody wytwarzania oprogramowania; (K_W04)

P_W03: Student posiada wiedzę odnośnie podstaw wytwarzania dokumentacji, analizy wymagań, projektowania, testowania oprogramowania. (K_W04)

Umiejętności

P_U01: Student potrafi zarządzać analizą wymagań; (K_U04, K_U06)

P_U02: Student potrafi tworzyć specyfikację ze zdefiniowanymi metrykami; (K_U04)

P_U03: Student potrafi utworzyć projekt oprogramowania za pomocą języka UML. (K_U04, K_U06)

Kompetencje społeczne (postawy)

P_K01: Student posiada podstawowe umiejętności komunikacji w zespole informatycznym (K_K02)

Kontakt

vonyshchenko@inf.ug.edu.pl