

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

<b>Nazwa przedmiotu</b>		<b>Kod ECTS</b>	
Inżynieria oprogramowania		11.3.0747	
<b>Nazwa jednostki prowadzącej przedmiot</b>			
Faculty of Mathematics, Physics and Informatics			
<b>Studia</b>			
<b>wydział</b>	<b>kierunek</b>	<b>poziom</b>	<b>pierwszego stopnia</b>
Wydział Matematyki, Fizyki i Informatyki	Informatyka	<b>forma</b>	stacjonarne
		<b>moduł</b>	wszystkie
		<b>specjalnościowy</b>	wszystkie
		<b>specjalizacja</b>	wszystkie
<b>Nazwisko osoby prowadzącej (osób prowadzących)</b>			
dr inż. Emilia Lubecka; dr Magdalena Godlewska; dr Paweł Pączkowski			
<b>Formy zajęć, sposób ich realizacji i przypisana im liczba godzin</b>		<b>Liczba punktów ECTS</b>	
<b>Formy zajęć</b>		2 Przedmiot w wymiarze 15h wykładu i 15h laboratorium + praca własna studenta.	
Wykład, Ćw. laboratoryjne			
<b>Sposób realizacji zajęć</b>			
zajęcia w sali dydaktycznej			
<b>Liczba godzin</b>			
Ćw. laboratoryjne: 15 godz., Wykład: 15 godz.			
<b>Cykl dydaktyczny</b>			
2016/2017 letni			
<b>Status przedmiotu</b>		<b>Język wykładowy</b>	
obowiązkowy		polski	
<b>Metody dydaktyczne</b>		<b>Forma i sposób zaliczenia oraz podstawowe kryteria oceny lub wymagania egzaminacyjne</b>	
<ul style="list-style-type: none"> <li>- Dyskusja</li> <li>- Rozwiązywanie zadań</li> <li>- Samodzielne wykonanie miniprojektów</li> <li>- Wykład z prezentacją multimedialną</li> </ul>		<b>Sposób zaliczenia</b>	
		<ul style="list-style-type: none"> <li>- Zaliczenie na ocenę</li> <li>- Zaliczenie (zał)</li> </ul>	
		<b>Formy zaliczenia</b>	
		<ul style="list-style-type: none"> <li>- ustalenie oceny zaliczeniowej na podstawie ocen cząstkowych otrzymywanych w trakcie trwania semestru</li> <li>- kolokwium</li> </ul>	
		<b>Podstawowe kryteria oceny</b>	
		Przedmiot kończy się pisemnym kolokwium zaliczeniowym, zaliczenie od 51% punktów. Przed przystąpieniem do kolokwium konieczność zaliczenia ćwiczeń laboratoryjnych. Ocena z ćwiczeń na podstawie wyników z prac domowych oraz aktywności na zajęciach.	
<b>Sposób weryfikacji założonych efektów kształcenia</b>			

zakładany efekt kształcenia	egzamin	kolokwium	projekt	referat	raport	aktywność w dyskusji	obserwacja
Wiedza							
K_W02		x					
K_W03		x					
K_W09		x					
K_W12							x
Umiejętności							
K_U03			x				
K_U04			x				
K_U06			x				
K_U19			x				
K_U22			x				
Kompetencje							
K_K06							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. Obiektość 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.

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.

### Efekty kształcenia (obszarowe i kierunkowe)

K\_W02: ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie programowania, algorytmów i złożoności, architektury systemów komputerowych, systemów operacyjnych, technologii sieciowych, języków i paradygmatów programowania, grafiki i komunikacji człowiek-komputer, sztucznej inteligencji, 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 struktury danych,  
 K\_U19: tworzy, ocenia i realizuje plan testowania, K\_U22: posługuje się wzorcami projektowymi  
 K\_K06: potrafi formułować opinie na temat podstawowych zagadnień informatycznych

### Wiedza

Student:

- zna różne modele cyklu życia oprogramowania;
- zna metody wytwarzania oprogramowania;
- posiada wiedzę odnośnie podstaw wytwarzania 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

elubecka@inf.ug.edu.pl