KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Inżynieria oprogramowania	**1010331551010330109**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**(brak)**	**3 / 5**
Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
-	**polski**	**obligatoryjny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
I stopień	**stacjonarna**

Godziny	Liczba punktów
Wykłady: **15** Ćwiczenia: **-** Laboratoria: **15** Projekty/seminaria: **-**	**3**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
(brak)	**(brak)**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**3 100%**
nauki techniczne	**3 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Andrzej Sikorski
email: andrzej.sikorski@put.poznan.pl
tel. 6653958
Wydział Elektryczny
ul. Piotrowo 3A 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	K_W00: ma podstawową wiedzę wynikającą z programu szkoły średniej K_W05: ma uporządkowaną i podbudowaną teoretycznie wiedzę w zakresie podstawowych konstrukcji programistycznych, implementacji algorytmów, paradygmatów i stylów programowania, metod weryfikacji poprawności programów, języków formalnych, kompilatorów, platform.
2	**Umiejętności:**	K_U01: potrafi pozyskiwać informacje z literatury, baz danych i innych źródeł; potrafi integrować uzyskane informacje, dokonywać ich interpretacji, a także wyciągać wnioski oraz formułować i uzasadniać opinie.
3	**Kompetencje społeczne**	K_K02: ma świadomość ważności i rozumie pozatechniczne aspekty i skutki działalności inżyniera-informatyka oraz związaną z tym odpowiedzialność za podejmowane decyzje.

Cel przedmiotu:

Celem przedmiotu obejmującego dwa semestry jest zapoznanie studentów z inżynierskim podejściem do wytwarzania oprogramowania. Podczas pierwszego semestru tego przedmiotu celem jest poznanie zasad modelowania obiektowego zgodnie ze standardem UML 2.0, dokonanie przeglądu modeli cyklu rozwojowego oprogramowania oraz procesów podtawowych i wspomagających w produkcji oprogramowania.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. Ma uporządkowaną i podbudowaną metodologicznie wiedzę w zakresie inżynierii oprogramowania - [K_W12]
2. Orientuje się w obecnym stanie oraz najnowszych trendach rozwojowych informatyki - [K_W19]

Umiejętności:

1. Potrafi sformułować wymagania, opracować model obiektowy oraz ocenić prosty system informatyczny, uwzględniając realizowane funkcje i powiązania między elementami składowymi. - [K_U16]
2. Potrafi przygotować i przedstawić krótką prezentację poświęconą wynikom realizacji zadania inżynierskiego. - [K_U14]

Kompetencje społeczne:

1. Ma świadomość ważności i rozumie pozatechniczne aspekty i skutki działalności inżyniera-informatyka i związaną z tym odpowiedzialność za podejmowane decyzje - [K_K02]
2. Ma świadomość odpowiedzialności za pracę własną oraz gotowość podporządkowania się zasadom pracy w zespole i ponoszenia odpowiedzialności za wspólnie realizowane zadania - [K_K04]

Sposoby sprawdzenia efektów kształcenia

Treści prezentowane podczas wykładu są przedmiotem egzaminu w następnym semestrze. Po serii wykładów w bieżącym semestrze ocena zostanie wystawiona na podstawie oceny aktywności studenta na zajęciach oraz sprawdzianu.

Aktualizacja (2017)

Zaliczenie zajęć laboratoryjnych odbywa się na podstawie ocen cząstkowych wystawianych

za wykonanie zadań w zakresie programowania asynchronicznego oraz modelowania obiektowego. W odróżnieniu od lat poprzednich zamiast zajmować się tzw. "projektem" studenci wykonują konkretne zadania przygotowane przez prowadzących laboratoria.

Treści programowe

Uzasadnienie znaczenia dziedziny Inżynierii Oprogramowania w procesie wytwarzania i projektowania aplikacji. Przykłady programistyczne uzasadniające używanie metod Inżynierii.

Aktualizacja 2017: zadania asynchroniczne, wzorzec projektowy Task w języku c#.

Programowanie z wykorzystaniem technik asynchronicznych rozumiane jako alternatywa dla programowania wielowątkowego. Techniki wykorzystywania zadań asynchronicznych w oprogramowaniu wielowątkowym.

Techniki inżynierii odwrotnej (analizy kodu wykonywalnego) w kontekście wzorca projektowego fasada/adapter.

Diagramy UML - klasy i przypadku użycia.

Laboratoria (od 2017): zadania dot. programowania asynchronicznego, wykorzystanie funkcji dostępnych w .NET, tworzenie własnych usług asynchronicznych

Zadania polegające na modelowaniu z wykorzystaniem diagramów przypadków użycia oraz klas. Studenci wykonują zadania szczegółowe dot. modelowania dla konkretnych przypadków dot. programowania.

Literatura podstawowa:

1. The Unified Modeling Language User Guide (2nd Edition) Booch et al.
2. Async in C# 5.0: Unleash the Power of Async

Literatura uzupełniająca:

1. Kernighan Ritchie: Język Ansi C
2. The C++ Programming Language

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. Wykład	15
2. Laboratoria	15
3. Modelowanie systemów informatycznych, projektowanie z użyciem UML	30
4. Konsultacje, zaliczenie	10

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	70	3
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	20	1
Zajęcia o charakterze praktycznym	50	1