KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Inżynieria oprogramowania	**1010331461010330109**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**(brak)**	**3 / 6**
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: **2** Ćwiczenia: **-** Laboratoria: **-** Projekty/seminaria: **1**	**4**

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	**4 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr hab. inż. Barbara Begier
email: Barbara.Begier@put.poznan.pl
tel. 665-3724
Wydział Elektryczny
ul. Piotrowo 3A 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	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. Ma uporządkowaną i podbudowaną metodologicznie wiedzę w zakresie inżynierii oprogramowania (pierwszej części przedmiotu).
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 i związaną z tym odpowiedzialność za podejmowane decyzje. K_K01: rozumie potrzebę i zna możliwości ciągłego dokształcania się (studia drugiego i trzeciego stopnia, studia podyplomowe, kursy), podnoszenia kompetencji językowych, zawodowych, osobistych i społecznych.

Cel przedmiotu:

Celem drugiej części przedmiotu jest zapoznanie studentów z metodami zapewniania i oceny jakości oprogramowania oraz poznanie zwinnych metodyk wytwarzania oprogramowania. Celem zajęć projektowych jest pogłębienie praktycznej znajomości standardu UML 2.0 oraz nabycie umiejętności opracowania wymaganych artefaktów.

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 opracować dokumentację dotyczącą realizacji zadania inżynierskiego i przygotować tekst zawierający omówienie wyników realizacji tego zadania. Umie opracować plan testów i drzewo jakości wyrobu programowego. - [K_U03]

Kompetencje społeczne:

1. Ma świadomość ważności dokładnego wykonania oprogramowania, zachowania standardów notacyjnych, przestrzegania poprawności językowej i terminowego oddania prac. - [K_K07]
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 pisemnego. Student wykazuje umiejętność tworzenia podstawowych konstrukcji w UML, znajomość zasad i praktyk postępowania w metodykach twardych i zwinnych oraz zna kryteria i miary oceny jakości oprogramowania.

Zaliczenie zajęć projektowych odbywa się na podstawie ocen cząstkowych, wystawianych oddzielnie za każdy opracowany dokument oraz diagram w UML.

Treści programowe

Wykład. Jakość wyrobu programowego i jej charakterystyki według standardów ISO 9126 oraz ISO 25010. Polityki jakości w procesie wytwarzania oprogramowania. Planowanie testowania. Charakterystyka metod zwinnych, zasady wyrażone w Agile Manifesto. Przegląd metodyk zwinnych: XP (eXtreme Programming), TDD (Test Driven Development), AMDD (Agile Model Driven Develeopment), FDD (Feature Driven Development), BDD (Behavior Driven Development), Scrum. Rola czynnika ludzkiego w produkcji oprogramowania. Satysfakcja użytkownika z wyrobu programowego, model EUCS (End User Computing Satisfaction).

Projekt. Opracowanie modelu zachowań przyszłego systemu. Opracowanie planu testowania oraz drzewa jakości tworzonego oprogramowania.

Literatura podstawowa:

1. Martin R., Martin M., Agile. Programowanie zwinne. Zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C?, Helion, Gliwice 2008.

2. Wrycza St., Marcinkowski B., Wyrzykowski K., Język UML 2.0 w modelowaniu systemów informatycznych, Helion, Gliwice 2005.

Literatura uzupełniająca:

1. Begier B., Inżynieria oprogramowania - problematyka jakości, Wydawnictwo Politechniki Pozn., Poznań 1999.

2. Hnatkowska B., Huzar Z., Inżynieria oprogramowania. Metody wytwarzania i wybrane zagadnienia, PWN, Warszawa 2008.

3. Pilone D., Pitman N., UML 2.0. Almanach, Helion, Gliwice 2007.

4. Subieta K., Wprowadzenie do inżynierii oprogramowania, Wydawnictwo PJWSTK, Warszawa 2002.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. Uczestnictwo w wykładach	30
2. Uczestnictwo w zajęciach projektowych	15
3. Przygotowanie projektu	20
4. Konsultacje i egzamin	10
5. Przygotowanie do egzaminu	25

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	100	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	55	2
Zajęcia o charakterze praktycznym	35	1