KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Inżynieria oprogramowania	**1010514351010510082**

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

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

Godziny	Liczba punktów
Wykłady: **16** Ćwiczenia: **-** Laboratoria: **16** Projekty/seminaria: **-**	**3**

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

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

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Bartłomiej Prędki
email: bartlomiej.predki@put.poznan.pl
tel. (0-61) 61 6652932
Instytut Informatyki
ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z podstaw programowania, narzędzi informatyki, algorytmów i struktur danych, programowania obiektowego, architektury systemów komputerowych, systemów baz danych.
2	**Umiejętności:**	Powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu programowania oraz umiejętność pozyskiwania informacji ze wskazanych źródeł.
3	**Kompetencje społeczne**	Powinien również rozumieć konieczność poszerzania swoich kompetencji / mieć gotowość do podjęcia współpracy w ramach zespołu Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu:

1. Przekazanie studentom podstawowej wiedzy z inżynierii oprogramowania, w zakresie organizacji przebiegu przedsięwzięcia programistycznego, określania wymagań, modelowania systemów, projektowania oprogramowania, zapewniania jakości, w tym niezawodności oprogramowania, refaktoryzacji, dokumentowania oprogramowania, projektowania API, konserwacji i ponownego wykorzystania oprogramowania, narzędzi wspomagających wytwarzanie oprogramowania, w tym narzędzi zarządzania wersjami.
2. Rozwijanie u studentów umiejętności rozwiązywania prostych problemów z zakresu projektowania, budowy i testowania oprogramowania, wykorzystania narzędzi wspomagających wytwarzanie oprogramowania, modyfikowania u wykorzystania komponentów programistycznych.
3. Kształtowanie u studentów umiejętności efektywnej pracy jako analityk/projektant/programista w zespole programistycznym pracującym zgodnie z klasycznymi lub żwawymi (agile) metodykami.

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

1. ma uporządkowaną, podbudowaną teoretycznie wiedzę ogólną w zakresie inżynierii oprogramowania. - [K_W4]

2. ma szczegółową wiedzę nt. inżynierii oprogramowania - [K_W5]

3. ma podstawową wiedzę o cyklu życia systemów informatycznych programowych - [K_W7]

4. zna podstawowe metody, techniki i narzędzia stosowane przy rozwiązywaniu prostych zadań informatycznych z zakresu inżynierii oprogramowania. - [K_W8]

5. zna podstawowe pojęcia z zakresu ekonomii odnoszące się do inwestycji informatycznych i projektów informatycznych takie, jak zwrot z inwestycji, koszty stałe i koszty zmienne, ryzyko finansowe, przychód a zysk, zysk a przepływy pieniężne (ang. cash flow) - [K_W11]

6. ma wiedzę o trendach rozwojowych i najistotniejszych nowych osiągnięciach w inżynierii oprogramowania - [K_W6]

7. zna i rozumie zasady określania wymagań, modelowania systemów, projektowania i refaktoryzacji oprogramowania, zapewniania jakości w tym niezawodności oprogramowania, testowania oprogramowania, dokumentowania oprogramowania, wykorzystania narzędzi wspomagających wytwarzanie oprogramowania. - [-]

Umiejętności:

1. potrafi ocenić - przynajmniej w podstawowym zakresie - różne aspekty ryzyka związanego z przedsięwzięciem informatycznym - [K_U10]

2. potrafi stworzyć model obiektowy prostego systemu (np. w języku UML) - [K_U14]

3. potrafi efektywnie uczestniczyć w inspekcji oprogramowania - [K_U16]

4. ma umiejętność systematycznego przeprowadzania testów funkcjonalnych i innych rodzajów testów - [K_U17]

5. potrafi sformułować wymagania pozafunkcjonalne dla wybranych charakterystyk jakościowych - [K_U19]

6. potrafi - zgodnie z zadaną specyfikacją - zaprojektować oraz zrealizować prosty system informatyczny, używając właściwych metod, technik i narzędzi - [K_U21]

7. ma umiejętność posługiwania się przynajmniej jednym z najbardziej popularnych systemów zarządzania wersjami - [K_U21]

8. potrafi posługiwać się technikami informacyjno-komunikacyjnymi wykorzystywanymi przy realizacji przedsięwzięć informatycznych - [K_U6]

9. potrafi sformułować specyfikację funkcjonalną w formie przypadków użycia - [K_U18]

10. potrafi zaprojektować dobry interfejs użytkownika dla różnych klas systemów informatycznych - [K_U25]

11. potrafi określić i opisać wymagania wobec prostych systemów informatycznych - [K_U21]

12. potrafi wybrać właściwy model i metodykę realizacji przedsięwzięcia programistycznego - [K_U21]

13. potrafi modelować systemy w języku UML - [K_U14]

14. potrafi projektować oprogramowanie z wykorzystaniem wzorców projektowych - [-]

15. potrafi refaktoryzować oprogramowanie - [-]

16. potrafi projektować i dokumentować API - [K_U25]

17. potrafi zapewnić niezawodność oprogramowania w tym przeprowadzić testy oprogramowania - [K_U17]

18. potrafi dokumentować oprogramowanie - [K_U18-19]

19. potrafi konserwować i ponownie wykorzystywać istniejące oprogramowanie - [-]

Kompetencje społeczne:

1. zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych lub też do poważnej utraty zdrowia, a nawet życia - [K_K4]

2. potrafi pracować jako efektywny członek zespołu programistycznego - [K_K5]

Sposoby sprawdzenia efektów kształcenia

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów:

- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach;

b) w zakresie ćwiczeń:

- na podstawie oceny bieżącego postępu realizacji zadań,

Ocena podsumowująca:

Sprawdzanie założonych efektów kształcenia realizowane jest przez:

- ocenę przygotowania studenta do poszczególnych sesji zajęć laboratoryjnych (sprawdzian ?wejściowy

Treści programowe

Program przedmiotu obejmuje następujące zagadnienia:

- Wprowadzenie w tym podstawowe modele cyklu życia i podstawowe rodzaje metodyk wytwarzania oprogramowania
- Analiza/modelowanie systemów z wykorzystaniem języka UML
- UML jako narzędzie projektowania i dokumentowania oprogramowania
- Projektowanie oprogramowania
- Wzorce projektowe
- Refaktoryzacja
- Projektowanie i dokumentowanie API
- Niezawodność oprogramowania ? unikanie błędów, odporność na błędy
- Niezawodność oprogramowania ? testowanie
- Konserwacja i ponowne wykorzystanie oprogramowania, dokumentacja techniczna i użytkowa
- Narzędzia inżynierii oprogramowania. Zarządzanie zmianami i konfiguracją

Program zajęć laboratoryjnych obejmuje następujące zagadnienia:

- Zintegrowane środowiska programistyczne, np. Eclipse
- Standardy kodowania
- Dokumentowanie oprogramowania, np. javadoc
- Modelowanie systemów w języku UML
- Metoda kart CRC
- Projektowanie oprogramowania z wykorzystaniem wzorców projektowych
- Implementacja kodu na podstawie diagramów UML
- Dokumentowanie kodu w języku UML
- Inspekcje kodu
- Testowanie oprogramowania w tym testy jednostkowe
- Narzędzia zarządzania wersjami, np. svn, mercurial

Cześć wymienionych wyżej treści programowych realizowana jest w ramach pracy własnej studenta.

Metody dydaktyczne:

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, studium przypadków,
2. ćwiczenia laboratoryjne: rozwiązywanie zadań, ćwiczenia praktyczne, dyskusja, praca w zespole, pokaz multimedialny, warsztaty, demonstracja,

Literatura podstawowa:

1. A. Jaszkiewicz, Inżynieria oprogramowania, Helion, 1997.
2. G. Booch, J. Rambaugh, I. Jacobson, UML przewodnik użytkownika, WNT, 2000.
3. M. Fowler, K. Scott, UML w kropelce, Oficyna Wydawnicza LTP, 2002
4. Sommerville Ian Inżynieria oprogramowania, WNT, 2003.
5. UML. Inżynieria oprogramowania. Wydanie IIUML. Inżynieria oprogramowania. Wydanie II, P. Stevens, Helion, Gliwice, 2007
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku, WNT, 2008
7. Sacha, Inżynieria oprogramowania, PWN, 2010.

Literatura uzupełniająca:

1. S. Maguire, Niezawodność oprogramowania, Helion, 2002
2. J. W. Cooper, Java. Wzorce projektowe, Helion, 2001
3. Pressman Roger S. Praktyczne podejście do inżynierii oprogramowania, WNT, 2004.
4. Warmer Jos, Kleppe Anneke, Inżynieria oprogramowania OCL precyzyjne modelowanie w UML, WNT, 2003.
5. Kernighan W. Brian, Pike Rob Inżynieria oprogramowania. Lekcja oprogramowania, WNT, 2002.
6. R.C. Martin, M. Martin, Agile, programowanie zwinne, Helion, 2008.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)

1. udział w zajęciach laboratoryjnych:	16
2. przygotowanie do ćwiczeń laboratoryjnych:	10
3. zadania domowe (w ramach pracy własnej)	10
4. udział w konsultacjach związanych z realizacją procesu kształcenia, w szczególności ćwiczeń laboratoryjnych / projektu	2
5. przygotowanie do sprawdzianów / kolokwium	5
6. udział w wykładach	16
7. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi (10 stron tekstu naukowego = 1 godz.), 100 stron	10
8. przygotowanie do zaliczenia wykładów	5

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	74	3
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	34	1
Zajęcia o charakterze praktycznym	36	1