KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Inżynieria oprogramowania	**1010331561010330109**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**(brak)**	**3 / 6**
Ścieżka obieralności/specjalność: **Technologie informatyczne**	Przedmiot oferowany w języku: **polski**	Kurs (obligatoryjny/obieralny) **obligatoryjny**

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

Godziny	Liczba punktów
Wykłady: **30** Ćwiczenia: **-** Laboratoria: **-** Projekty/seminaria: **15**	**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%**
nauki techniczne	**4 100%**

Odpowiedzialny za przedmiot / wykładowca:

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

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Adam Meissner
email: adam.meissner@put.poznan.pl
tel. 61 665 37 24
Wydział Elektryczny
ul. Piotrowo 3A 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	Student ma uporządkowaną i podbudowaną teoretycznie wiedzę z zakresu podstaw inżynierii oprogramowania obejmującą elementy modelowania i realizacji systemów informatycznych. Znajomość Visual Paradigm. Znajomość teoretycznych podstaw informatyki w zakresie programowania obiektowego, podstawowych algorytmów. Model relacyjny.
2	**Umiejętności:**	Inżynieria oprogramowania w zakresie 5 sem. (diagramy klas, przypadków użycia, diagramy wymagań). Programowanie w językach C,C++,C#. Podstawy baz danych. Programowanie współbieżne.
3	**Kompetencje społeczne**	Umiejętność pracy w zespole. Poczucie odpowiedzialności, lojalność oraz dyskrecja w sprawach poufnych.

Cel przedmiotu:

aktualizacja 2017 nietrywialne wzorce dot. nowych platform i architektur.

Zapoznanie się z wybranymi metodami modelowania i wytwarzania systemów informatycznych oraz z metodami walidacji, weryfikacji i testowania ich poprawności.

Pozyskanie wiedzy z zakresu modelowania obiektowego, relacyjnego oraz współbieżnego.

Biegłość w posługiwaniu się diagramami UML do celów projektowania aplikacji biznesowych, inżynierskich ze szczególnym uwzględnieniem programowania współbieżnego.

Przedmiot jest zorientowany na wsparcie studenta w zakresie ćwiczeń laboratoryjnych, podczas których powinien nabrać umiejętności w zakresie zagadnień omawianych na wykładzie.

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 pozyskiwać informacje z dokumentacji technicznej narzędzi programistycznych, bibliotek, API. - [K_U16]
2. Potrafi odwzorowywać zadania programistyczne na struktury dostępne w środowiskach programistycznych. - [K_U03]

Kompetencje społeczne:

1. Ma świadomość ważności dokładnego wykonania oprogramowania, zachowania standardów notacyjnych - [K_K07]
2. Ma świadomość odpowiedzialności za pracę własną oraz gotowość podporządkowania się zasadom pracy w zespole - [K_K04]

Sposoby sprawdzenia efektów kształcenia

Egzamin pisemny. zadania UML oraz programistyczne.

Seminarium lub kolokwium w trakcie semestru.

Zadania o podwyższonym stopniu trudności proponowane w trakcie semestru.

Rozwiązywanie zadań technicznych z dziedziny przedmiotu proponowanych w treści wykładu.

Treści programowe

Aktualizacja 2017 Wzorce programowe :promise (kontynuacja programowania asynchronicznego z sem.5), barrier (MPI oraz obiekty oferowane przez platformy programistyczne), metoda Scrum.

Diagramy UML dynamiczne: interakcji, sekwencji, przebiegów czasowych, stanów.

Wzorce programowe w zakresie programowania współbieżnego. Specyfikacja w UML mechanizmów synchronizacyjnych wysokiego poziomu.

Modelowanie systemów czasu rzeczywistego.

Projektowanie relacyjne. Wzorce projektowe z zakresu modelu relacyjnego. Dzielenie relacyjne, anty-złączenie, techniki przepisywania zapytań SQL.

Formalne metody modelowania programów. Programowanie zwinne i programowanie ekstremalne. Metoda Scrum. Metody walidacji, weryfikacji i testowania oprogramowania.

Zastosowane metody kształcenia:

- wykład z prezentacją slajdów uzupełniany przykładami podawanymi na tablicy

- projekt - szczegółowe recenzowanie dokumentacji projektowych przez prowadzącego połączone z omówieniem typowych błędów oraz propozycjami ich usuwania.

Literatura podstawowa:

1. Paulish D.J., Inżynieria oprogramowania. Zarządzanie architekturocentrycznym procesem tworzenia oprogramowania. Przewodnik praktyczny, WNT, Warszawa, 2007

2. Schwaber K., Sutherland J., The Scrum Guide TM. The Definitive Guide to Scrum: The Rules of the Game, July 2016, http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Polish.pdf

3. Shore S., Warden S., Agile Development. Filozofia programowania zwinnego, Wyd. Helion, Gliwice, 2008

4. Zmitrowicz K., Jakość projektów informatycznych. Rozwój i testowanie oprogramowania, Wyd. Helion, Gliwice, 2015

Literatura uzupełniająca:

1. Jeffries R., Programowanie ekstremalne w C#, PWN, Warszawa, 2005

2. Rad N.K., Turley F., The Scrum Master Training Manual. A Guide to Passing the Professional Scrum Master (PSM) Exam, Management Plaza, 2013, https://mplaza.pm/downloads/Scrum%20Training%20Manual.pdf

3. Sutherland J., Jeff Sutherland?s Scrum Handbook, Scrum Training Institute Press, 2010, http://www.ugrad.cs.ubc.ca/~cs310/2014W1/slides/Sutherland_Scrum_Handbook.pdf

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. Wykład	30
2. Zajęcie projektowe	15
3. Praca własna	20

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	65	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	45	3
Zajęcia o charakterze praktycznym	15	1