



## KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Inżynieria oprogramowania

### Przedmiot

Kierunek studiów

Bioinformatyka

Studia w zakresie (specjalność)

Poziom studiów

pierwszego stopnia

Forma studiów

stacjonarne

Rok/semestr

3/5

Profil studiów

ogólnoakademicki

Język oferowanego przedmiotu

polski

Wymagalność

obligatoryjny

### Liczba godzin

Wykład

15

Ćwiczenia

Laboratoria

30

Projekty/seminaria

Inne (np. online)

### Liczba punktów ECTS

4

### Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:

dr inż. Mirosław Ochodek

email: [Miroslaw.Ochodek@cs.put.poznan.pl](mailto:Miroslaw.Ochodek@cs.put.poznan.pl)

tel. 61 665 2944

Wydział Informatyki i Telekomunikacji

ul. Piotrowo 3, 60-965 Poznań

Odpowiedzialny za przedmiot/wykładowca:

dr hab. inż. Piotr Zielniewicz

email: [pzielniewicz@cs.put.poznan.pl](mailto:pzielniewicz@cs.put.poznan.pl)

Wydział Informatyki i Telekomunikacji

ul. Piotrowo 3, 60-965 Poznań

### Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z podstaw programowania, narzędzi informatyki, algorytmów i struktur danych, programowania obiektowego. Ponadto powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu programowania oraz umiejętność pozyskiwania informacji ze wskazanych źródeł.

### 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 testowania 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 zwinnymi (Agile) metodykami.

### Przedmiotowe efekty uczenia się

#### Wiedza

1. Ma podstawową wiedzę w zakresie zarządzania projektami informatycznymi.
2. Ma podstawową wiedzę w zakresie inżynierii wymagań (wymagania funkcjonalne, przypadki użycia, wymagania pozafunkcjonalne).
3. Ma podstawową wiedzę w zakresie modelowania i projektowania oprogramowania.
4. Ma podstawową wiedzę w zakresie metod weryfikacji i walidacji oprogramowania.

#### Umiejętności

1. Potrafi specyfikować wymagania funkcjonalne i pozafunkcjonalne.
2. Potrafi stworzyć prototyp interfejsu użytkownika aplikacji.
3. Potrafi tworzyć modele obiektowe w notacji UML (model klas, maszyny stanowej, sekwencji).
4. Potrafi tworzyć przypadki testowe i dokonywać ich automatyzacji (testy jednostkowe).
5. Potrafi zorganizować pracę w zespole wytwarzającym oprogramowanie.
6. Potrafi posługiwać się zintegrowanym środowiskiem programistycznym.

#### Kompetencje społeczne

1. Potrafi współdziałać i pracować w zespole programistycznym.

### Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów: na podstawie odpowiedzi na pytania oraz udział w quizach w trakcie zajęć wykładowych

b) w zakresie laboratorium: na podstawie oceny bieżącego postępu realizacji zadań w trakcie laboratorium

Ocena podsumowująca:

W zakresie oceny efektów kształcenia dotyczących nabytych umiejętności oraz kompetencji społecznych (głównie ocena z zajęć laboratoryjnych):



a) ocena końcowa składa się z oceny realizacji miniprojektu (50%) oraz kolokwium (50%). Kolokwium składa się zarówno z pytań testowych (wielokrotnego wyboru) oraz zadań o charakterze otwartym (opracowanie diagramu UML, napisanie fragmentu kodu dot. testowania jednostkowego).

W zakresie efektów kształcenia dotyczących nabytej wiedzy:

a) w trakcie wykładów studenci rozwiązują quizy oraz krótkie zadania o charakterze problemowym lub biorą udział w quizie. Za dostarczenie akceptowalnego rozwiązania (w zależności od jego formy i charakteru) student otrzymuje 1%.

b) test wyboru obejmujący pytania jednokrotnego wyboru (jedna prawidłowa odpowiedź) oraz pytania z możliwie jedną lub wieloma poprawnymi odpowiedziami (typ pytania jest jawnie wskazany w teście). Za udzielenie poprawnej odpowiedzi na pytanie student otrzymuje 1 punkt. Punkty przeliczane są na skalę procentową.

Na podstawie uzyskanych punktów procentowych (z testu wyborów oraz w trakcie wykładu) wyznaczana jest ocena końcowa według skali:

- $\geq 90\%$  - 5,0
- $<80\%$ , 90%) - 4,5
- $<70\%$ , 80%) - 4,0
- $<60\%$ , 70%) - 3,5
- $<50\%$ , 60%) - 3,0
- mniej niż 50% - 2,0

### **Treści programowe**

Program przedmiotu obejmuje następujące zagadnienia:

- Wprowadzenie, w tym znaczenie i rola wytwarzania oprogramowania we współczesnym świecie, wizja projektu informatycznego, konsekwencje błędów w oprogramowaniu, zakres tematyczny inżynierii oprogramowania
- Systemy zarządzania wersjami (Git)
- Wymagania funkcjonalne i pozafunkcjonalne oraz modelowanie procesów biznesowych
- Modelowanie i analiza oprogramowania (w tym notacja UML)
- Prototypowanie oprogramowania
- Projektowanie oprogramowania
- Metodyki zarządzania projektami (Scrum)



- Testowanie oprogramowania (jednostkowe oraz akceptacyjne)

Program zajęć laboratoryjnych obejmuje następujące zagadnienia:

- Narzędzia zarządzania konfiguracją oprogramowania (Git)
- Prototypowanie interfejsu użytkownika
- Dokumentowanie wymagań (przypadki użycia, opowieści użytkowników)
- Modelowanie systemów w notacji UML
- Testowanie oprogramowania - testy jednostkowe
- Praktyczna realizacja mini-projektu
- Wzorce projektowe

### Metody dydaktyczne

Metody dydaktyczne obejmują:

- a) wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, studium przypadków.
- b) ćwiczenia laboratoryjne: rozwiązywanie zadań, ćwiczenia praktyczne, dyskusja, praca w zespole, pokaz multimedialny, warsztaty, demonstracja oraz realizacja mini-projektu.

### Literatura

Podstawowa

1. I. Sommerville, Inżynieria oprogramowania, Wydawnictwo Naukowe PWN, 2020
2. A. Jaskiewicz, Inżynieria oprogramowania, Helion, 1997.
3. K. Schwaber, J. Sutherland, The Scrum Guide: Przewodnik po Scrumie: Reguły Gry, <http://www.scrumguides.org>, (dostępny online), 2017.

Uzupełniająca

1. Wzorce projektowe w języku Java: [https://www.tutorialspoint.com/design\\_pattern](https://www.tutorialspoint.com/design_pattern)
2. Kopczyńska, Sylwia, Jerzy Nawrocki, and Mirosław Ochodek. An Empirical Study on Catalog of Non-functional Requirement Templates: Usefulness and Maintenance Issues. Information and Software Technology (2018).
3. Nawrocki, Jerzy, et al. Agile requirements engineering: A research perspective. International Conference on Current Trends in Theory and Practice of Informatics. Springer, Cham, 2014.



**Bilans nakładu pracy przeciętnego studenta**

	Godzin	ECTS
Łączny nakład pracy	100	4,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	45	2,0
Praca własna studenta (studia literaturowe, praca w ramach mini-projektu, przygotowanie się do testu wiedzy) <sup>1</sup>	55	2,0

<sup>1</sup> niepotrzebne skreślić lub dopisać inne czynności