



## KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Inżynieria oprogramowania

### Przedmiot

Kierunek studiów

Sztuczna Inteligencja

Studia w zakresie (specjalność)

Poziom studiów

pierwszego stopnia

Forma studiów

stacjonarne

Rok/semestr

2/4

Profil studiów

ogólnoakademicki

Język oferowanego przedmiotu

angielski

Wymagalność

obligatoryjny

### Liczba godzin

Wykład

30

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 inż. Wojciech Complak

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

tel. 61 665 2983

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, architektury systemów komputerowych, systemów baz danych. 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) metodami.

### 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 uczestniczyć w spotkaniach projektowych w metodyce Scrum w roli członka zespołu deweloperskiego (Sprint Planning, Daily Scrum, Sprint Review i Sprint Retrospective).
2. Potrafi specyfikować wymagania funkcjonalne i pozafunkcjonalne.
3. Potrafi tworzyć modele obiektowe w notacji UML (model klas, maszyny stanowej, sekwencji).
4. Potrafi tworzyć przypadki testowe i dokonywać ich automatyzacji (testy jednostkowe, testy akceptacyjne oraz testy wydajnościowe).

#### Kompetencje społeczne

1. Ma świadomość tego, że narzędzia oraz biblioteki programistyczne podlegają ciągłym i częstym zmianom (np. na podstawie zmian w bibliotece JUnit, czy narzędzi do zarządzania wersjami).
2. 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.
3. potrafi identyfikować rzeczywiste problemy o znaczeniu komercyjnym, które można rozwiązać poprzez implementacje i wdrożenie systemów informatycznych.

### 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) w zależności od stopnia realizacji zadań na poszczególnych zajęciach laboratoryjnych student może otrzymać 0 albo 10 punktów. Student nieobecny na zajęciach może odrobić zajęcia w innym terminie lub za zgodą prowadzącego wykonać zadania w domu. Każdy student może uzyskać łącznie od 0 do 120 punktów.

b) w trakcie semestru studenci realizują projekt grupowy (3-5 osób) według zaleceń metodyki Scrum. Projekt składa się z dwóch sprintów (iteracji). W każdym sprincie zespół może uzyskać od 0 do  $n \cdot 100$  (gdzie  $n$  jest liczbą osób w zespole) punktów w zależności od stopnia realizacji zadań. Każdy z członków zespołu może otrzymać maksymalnie 100 punktów za sprint co daje łącznie maksymalnie 200 punktów.

Na podstawie sumy uzyskanych punktów wyznaczona zostaje ocena końcowa według następujący progów:

-  $\geq 280$  - 5,0

-  $< 250, 280$ ) - 4,5

-  $< 220, 250$ ) - 4,0

-  $< 190, 220$ ) - 3,5

-  $< 160, 190$ ) - 3,0

- mniej niż 160 - 2,0

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 25 pytań jednokrotnego wyboru (jedna prawidłowa odpowiedź) lub pytań 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
- Zarządzanie konfiguracją oprogramowania (w tym systemy zarządzania wersjami - Git i Subversion, narzędzia automatycznego budowania oprogramowania - Apache Ant oraz Apache Maven, praktyki ciągłej integracji, oraz podstawy zarządzania wersjami środowiska uruchomieniowego i konteneryzacji aplikacji)
- Wymagania funkcjonalne (w tym przypadki użycia)
- Wymagania pozafunkcjonalne (w tym norma ISO 25010)
- Modelowanie i analiza oprogramowania (w tym notacja UML)
- Projektowanie oprogramowania (w tym wzorce projektowe)
- Architektura oprogramowania
- Metodyki zarządzania projektami (Scrum oraz PRINCE2)
- Zarządzanie jakością oprogramowania (m.in. pomiar w procesie wytwarzania oprogramowania)
- Testowanie oprogramowania (jednostkowe, integracyjne, akceptacyjne, pozafunkcjonalne)

Program zajęć laboratoryjnych obejmuje następujące zagadnienia:

- Ocena ryzyka w projektach informatycznych
- Narzędzia zarządzania konfiguracją oprogramowania, np. Git, Apache Ant, Apache Maven
- Dokumentowanie wymagań funkcjonalnych z wykorzystaniem metody przypadków użycia



- Dokumentowanie wymagań pozafunkcyjnych
- Modelowanie systemów w notacji UML
- Projektowanie oprogramowania z wykorzystaniem wzorców projektowych
- Testowanie oprogramowania, w tym testy jednostkowe i wydajnościowe
- Praktyczna realizacja mini-projektu według zaleceń metodyki Scrum.

### Metody dydaktyczne

Mini-projekt realizowany jest według autorskiej metody opisanej w poniższym artykule:

Ochodek, Mirosław. "A Scrum-Centric Framework for Organizing Software Engineering Academic Courses." In Towards a Synergistic Combination of Research and Practice in Software Engineering, pp. 207-220. Springer, Cham, 2018.

Pozostałe 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.

### Literatura

Podstawowa

1. I. Sommerville, Software Engineering, 9th ed., Pearson Education, 2011.
2. K. Schwaber, J. Sutherland, The Scrum Guide, <http://www.scrumguides.org>, (dostępny online), 2020.

Uzupełniająca

1. Ochodek, Mirosław, J. Nawrocki, and K. Kwarciak. Simplifying effort estimation based on Use Case Points. Information and Software Technology 53.3 (2011): 200-213.
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	60	2,5
Praca własna studenta (studia literaturowe, praca w ramach mini-projektów, przygotowanie się do testu wiedzy) <sup>1</sup>	40	1,5

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