SYLABUS

DOTYCZY CYKLU KSZTAŁCENIA *2025-2029*

ROK AKADEMICKI 2027/2028

1. PODSTAWOWE INFORMACJE O PRZEDMIOCIE

Nazwa przedmiotu	*Inżynieria oprogramowania*
Kod przedmiotu*	
nazwa jednostki prowadzącej kierunek	*Instytut Informatyki, Wydział Nauk Ścisłych i Technicznych*
Nazwa jednostki realizującej przedmiot	*Instytut Informatyki, Wydział Nauk Ścisłych i Technicznych*
Kierunek studiów	*Informatyka i ekonometria*
Poziom studiów	*studia pierwszego stopnia*
Profil	*praktyczny*
Forma studiów	*studia stacjonarne*
Rok i semestr/y studiów	*rok III, sem. V*
Rodzaj przedmiotu	*przedmiot kierunkowy*
Język wykładowy	*polski*
Koordynator	*dr inż. Przemysław Pardel*
Imię i nazwisko osoby prowadzącej / osób prowadzących	*dr inż. Przemysław Pardel, mgr inż. Ewa Żesławska*

1.1. Formy zajęć dydaktycznych, wymiar godzin i punktów ECTS

Semestr (nr)	Wykł.	Ćw.	Konw.	Lab.	Sem.	ZP	Prakt.	Inne (jakie?)	**Liczba pkt. ECTS**
5	15			30					3

1.2. Sposób realizacji zajęć

zajęcia w formie tradycyjnej

1.3. Forma zaliczenia przedmiotu (z toku)

zaliczenie z oceną

2. WYMAGANIA WSTĘPNE

- Znajomość dowolnego języka obiektowego
- Umiejętność z zakresu tworzenia aplikacji z wykorzystaniem języka obiektowego
- Podstawowe umiejętności z zakresu testowania oprogramowania

3. CELE, EFEKTY UCZENIA SIĘ, TREŚCI PROGRAMOWE I STOSOWANE METODY DYDAKTYCZNE

3.1. Cele przedmiotu

C1	Zapoznanie studentów z zagadnieniami inżynierii oprogramowania
C2	Zapoznanie studentów z etapami cyklu rozwoju oprogramowania
C3	Zapoznanie studentów z podstawowymi zagadnieniami różnych metodologii zarządzania projektami
C4	Zapoznanie studentów z metodami analizy oprogramowania
C5	Zapoznanie studentów z metodami testowania, utrzymania i rozwoju oprogramowania

3.2. Efekty uczenia się dla przedmiotu

EK (efekt uczenia się)	Treść efektu uczenia się zdefiniowanego dla przedmiotu	Odniesienie do efektów kierunkowych
EK_01	Zna proces inżynierii oprogramowania, jego etapy i potrafi je zastosować w projekcie informatycznym stosując się do zasad ochrony własności intelektualnej i przemysłowej	K_W05, K_W08
EK_02	Zna zagadnienia związane z procesem analizy oprogramowania, potrafi tworzyć diagramy UML, potrafi zrealizować na ich podstawie oprogramowanie	K_W05, K_U08, K_U09, K_U10
EK_03	Potrafi projektować i implementować systemy informatyczne posługując się narzędziami wspomagającymi proces wytwarzania oprogramowania na różnych jego etapach.	K_U10
EK_04	Potrafi testować tworzone systemy komputerowe, rozumie rolę testów jednostkowych	K_U09,

3.3. Treści programowe

A. Problematyka wykładu

1. Wprowadzenie do inżynierii oprogramowania
   a. definicja i zakres inżynierii oprogramowania,
   b. różnice między programowaniem a inżynierią oprogramowania,
   c. warstwy inżynierii oprogramowania: proces, metody, narzędzia.
   d. cele i Kluczowe Rezultaty (OKR) w inżynierii oprogramowania: jak łączyć wizję z praktyką.
2. Proces tworzenia oprogramowania
   a. modele cyklu życia oprogramowania (kaskadowy, przyrostowy, spiralny, agile / kanban / Scrum).
3. Rola dokumentacji, planowania i kontroli jakości.
4. Inżynieria wymagań.
   a. identyfikacja wymagań funkcjonalnych i niefunkcjonalnych,
   b. techniki zbierania wymagań (wywiady, ankiety, obserwacja, user stories),
   c. specyfikacja i walidacja wymagań.
5. Analiza oprogramowania
   a. modelowanie systemu (diagramy UML – przypadki użycia, klasy, sekwencje),
   b. analiza domeny i identyfikacja głównych komponentów,
   c. techniki dekompozycji problemu.
6. Projektowanie oprogramowania
   a. architektura systemów,
   b. wzorce projektowe (kreacyjne, strukturalne, behawioralne – przykłady),
   c. dobre praktyki projektowania (SOLID, DRY, KISS).
7. Testowanie oprogramowania
   a. rodzaje testów (jednostkowe, integracyjne, systemowe, akceptacyjne),
   b. metodologia TDD (Test Driven Development) – założenia i praktyka,
   c. automatyzacja testów.
8. Utrzymanie i ewolucja oprogramowania
   a. refaktoryzacja i zarządzanie zmianami,
   b. techniczny dług,
   c. znaczenie dokumentacji i kontroli wersji w cyklu życia systemu.

B. Problematyka ćwiczeń , konwersatoriów, laboratoriów, zajęć praktycznych

1. Wprowadzenie i organizacja pracy zespołowej / OKR w praktyce: formułowanie celów i mierzenie rezultatów w zespołach Agile a. podstawy zarządzania projektem b. OKR w praktyce: formułowanie celów i mierzenie rezultatów w zespołach Agile
2. Modele cyklu życia oprogramowania – case study a. analiza przykładowego projektu w różnych modelach (waterfall vs agile).
3. Tworzenie wymagań dla analizowanego systemu informatycznego a. opracowanie wymagań funkcjonalnych i niefunkcjonalnych dla prostego systemu, b. tworzenie user stories i acceptance criteria.
4. Analiza wymagań – UML a. Język UML: diagram przypadków użycia
5. Modelowanie domeny a. Język UML: diagram klas b. opracowanie diagramów klas i diagramów sekwencji.
6. Projektowanie architektury a. Język UML: diagramy komponentów, pakietów
7. Analiza funkcjonalności systemu; a. Język UML: diagram sekwencji
8. Projektowanie procesów biznesowych; a. Język UML: diagram aktywności
9. Testowanie aplikacji
10. Dokumentacja techniczna projektowanego systemu a. dokumentacja kodu b. przygotowanie wersji aplikacji i release notes

3.4. Metody dydaktyczne

Wykład: wykład z prezentacją multimedialną, dyskusja
Laboratorium: projekt, warsztaty

4. METODY I KRYTERIA OCENY

4.1. Sposoby weryfikacji efektów uczenia się

Symbol efektu	Metody oceny efektów uczenia się (np.: kolokwium, egzamin ustny, egzamin pisemny, projekt, sprawozdanie, obserwacja w trakcie zajęć)	Forma zajęć dydaktycznych (w, ćw, ...)
EK_01 EK_02	kolokwium / sprawozdanie indywidualne	wykład, laboratorium
EK_03 EK_04	projekt / kolokwium	laboratorium

4.2 Warunki zaliczenia przedmiotu (kryteria oceniania)

1. Zaliczenie przedmiotu odbywa się na podstawie dwóch składowych:
   a. zaliczenia wykładu,
   b. oceny z laboratorium.
2. Zaliczenie laboratorium obejmuje:
   a. projekt indywidualny,
   b. kolokwium zaliczeniowe
3. Projekt indywidualny polega na opracowaniu rozwiązania konkretnego problemu z zakresu inżynierii oprogramowania oraz osobistej prezentacji wyników projektu.
4. Student zalicza laboratorium, jeżeli:
   a. uzyska ocenę co najmniej 3.0 z projektu,

b. ocenę co najmniej 3.0 z kolokwium zaliczeniowego.

5. Student zalicza przedmiot, jeśli:
   a. otrzymał z laboratorium ocenę co najmniej 3.0.
   b. zdobył min 50% punktów z testu

Ocenę dst. z kolokwium/projektu student uzyskuje w przypadku uzyskania minimum połowy możliwych do zdobycia punktów. Uzyskane punkty odpowiadają skali: do 50% - ocena 2.0; od 51% do 69% - ocena 3.0; od 70% do 79% - ocena 3.5; od 80% do 87% - ocena 4.0; od 88% do 94% - ocena 4.5; od 95% do 100% - ocena 5.0.

Taka forma zaliczeń pozwala na pełną weryfikację założonych efektów uczenia się.

5. CAŁKOWITY NAKŁAD PRACY STUDENTA POTRZEBNY DO OSIĄGNIĘCIA ZAŁOŻONYCH EFEKTÓW W GODZINACH ORAZ PUNKTACH ECTS

Forma aktywności	Średnia liczba godzin na zrealizowanie aktywności
Godziny kontaktowe wynikające z planu studiów	45
Inne z udziałem nauczyciela akademickiego (udział w konsultacjach, egzaminie)	2
Godziny niekontaktowe – praca własna studenta (przygotowanie do zajęć, egzaminu, napisanie referatu itp.)	28
SUMA GODZIN	75
SUMARYCZNA LICZBA PUNKTÓW ECTS	**3**

6. PRAKTYKI ZAWODOWE W RAMACH PRZEDMIOTU

wymiar godzinowy	-
zasady i formy odbywania praktyk	-

7. LITERATURA

Literatura podstawowa:

- Ian Sommerville – Inżynieria oprogramowania (wyd. polskie, Helion)
- Roger S. Pressman – Inżynieria oprogramowania. Podejście praktyczne (wyd. polskie, PWN
- Steve McConnell – Kod kompletny (Code Complete, wyd. polskie, Helion)
- Frederick P. Brooks – Mityczny osobomiesiąc (wyd. polskie, Helion)

Literatura uzupełniająca:

- Jeff Sutherland – Scrum. O zwinnym zarządzaniu projektami (wyd. polskie, Helion)
- John Doerr – Rozmierz to! Jak Google, Bono i Gates definiują cele i osiągają sukcesy (wyd. polskie, MT Biznes)

Akceptacja Kierownika Jednostki lub osoby upoważnionej