INŻYNIERIA OPROGRAMOWANIA

Kod przedmiotu: IO

Rodzaj przedmiotu: kierunkowy; obowiązkowy

Wydział: Informatyki

Kierunek: Informatyka

Poziom studiów: pierwszego stopnia – VI poziom PRK

Profil studiów: praktyczny

Forma studiów: stacjonarna/niestacjonarna

Rok: 2

Semestr: 4

Formy zajęć i liczba godzin:

Forma stacjonarna

wykłady – 25

laboratorium – 20

Forma niestacjonarna

wykłady – 15

laboratorium – 15

Zajęcia prowadzone są w języku polskim.

Liczba punktów ECTS: 4

Osoby prowadzące:

wykład:
laboratorium:

---

1. Założenia i cele przedmiotu:

Celem przedmiotu jest przekazanie studentom wiedzy na temat przedsięwzięć informatycznych, podstawowych zasad projektowania oprogramowania, ciężkich i zwinnych metodyk projektowania, zasad realizacji podstawowych etapów projektowania: zbierania wymagań i tworzenia specyfikacji, walidacji i testowania oprogramowania, wdrażania, utrzymania i ewolucji oprogramowania.
Celem przedmiotu jest również zapoznanie studentów z zasadami korzystania z API, narzędziami i środowiskiem wytwarzania oprogramowania, a także z procesami wytwarzania oprogramowania i zarządzania przedsięwzięciem programistycznym.

2. Określenie przedmiotów wprowadzających wraz z wymaganiami wstępnymi:

3. Opis form zajęć

a) Wykłady

- **Treści programowe (tematyka zajęć):**

1. Pojęcie przedsięwzięcia informatycznego. Zagadnienia i pojęcia związane z wytwarzaniem oprogramowania i wszystkich artefaktów, powstałych w trakcie realizacji projektu

informatycznego,
2. Podstawowe modele procesu tworzenia oprogramowania: kaskadowy, przyrostowy, spiralny, ewolucyjny, prototypowanie. Rola dokumentacji w procesie projektowania,
3. Znaczenie metodyk ciężkich i zwinnych w procesie budowy systemu informatycznego. Podstawowe kryteria doboru modelu procesu wytwarzania oprogramowania,
4. Cztery podstawowe działania w procesie projektowania: od zebrania wymagań do wdrożenia i ewolucji produktu informatycznego,
5. Proces zbierania wymagań i tworzenia specyfikacji projektu, standaryzacja metod specyfikacji i dokumentowania projektu informatycznego,
6. Realizacja ogólnego projektu, język UML: historia i geneza języka, koncepcja modeli UML, elementy, diagramy, związki, modelowanie funkcjonalności, diagram klas, diagram obiektów, diagram komponentów, diagram pakietów, diagramy interakcji, rodzaje komunikatów, diagramy stanu i czynności,
7. Mechanizm rozszerzeń UML, narzędzia związane z językiem UML,
8. Tworzenie kodu, zasady integracji kodu, inspekcja kodu,
9. Diagnostyka oprogramowania; Podstawowe wiadomości o testowaniu: strategia testowania, przygotowania testów, rodzaje testów,
10. Projektowanie kierowane testami,
11. Automatyzacja wykonywania testów,
12. Narzędzia wspomagające projektowanie,
13. Zarządzania konfiguracją oprogramowania,
14. Współczesne problemy, metody i narzędzia zarządzania całym cyklem wytwarzania oprogramowania.

- **Metody dydaktyczne**:
  Wykład prowadzony metodą tradycyjną z wykorzystaniem rzutnika multimedialnego wraz z prezentacją diagramów i narzędzi projektowych.
- **Forma i warunki zaliczenia:**
  Warunkiem zaliczenia wykładu jest zdanie sprawdzianu w formie testowej. Sprawdzian powinien uwzględniać przede wszystkim treści teoretyczne.

Wykaz literatury podstawowej:

1. P.Stevens: UML Inżynieria oprogramowania; Helion Gliwice 2007
2. Gaddis T.: Projektowanie oprogramowania dla zupełnie początkujących. Gliwice: Helion, 2020.
3. I. Sommerville: Inżynieria oprogramowania; WNT Warszawa 2003
4. Martin R.C.: Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów. Gliwice: Helion, copyright 2018.
5. W. Dąbrowski, A. Stasiak, M. Wolski: Modelowanie systemów w języku UML 2.1, PWN, Warszawa 2007

Wykaz literatury uzupełniającej:

1. K. Subieta: Wprowadzenie do inżynierii programowania, Wydawnictwo PJWSTK, Warszawa 2002, ISBN 83-89244-00-4.
2. D. Astels, G. Mills, M. Novak: eXtreme programming: Teoria i praktyka prowadzenia projektów informatycznych, Helion, 2002,
3. B. Wiszniewski, B. Bereza-Jarociński: Teoria i praktyka testowania programów, PWN, Warszawa, 2007,
4. J. Cadle, D. Yeates: Zarządzanie procesem tworzenia systemów informacyjnych, WNT, Warszawa, 2004,
5. J. Highsmith: APM - Agile Project Management, jak tworzyć innowacyjne produkty, PWN, Warszawa 2007,
6. Jaszkiewicz: Inżynieria oprogramowania, WNT, Warszawa, 2003, ISBN: 83-7197-007-2,
7. J. Górski (red.): Inżynieria oprogramowania. Mikom, Warszawa, 2000, ISBN 83-7279-028-0,
8. K. Beck, A. Cynthia: Wydajne programowanie – Extreme Programming, Mikom, 2005.

b) Ćwiczenia laboratoryjne

- **Treści programowe (tematyka zajęć):**
  1. Zebranie i analiza wymagań – diagram przypadków użycia,
  2. Zapoznanie się i porównanie narzędzi projektowania, m.in.: StarUML, VisualParadigm, Enterprise Architect,
  3. Analiza i model środowiska, model kontekstowy,
  4. Diagram klas, diagram obiektów; identyfikacja klas z wykorzystaniem kart CRC, identyfikacja związków i zależności,
  5. Diagramy interakcji: sekwencji i współpracy, rodzaje komunikatów, modelowanie warunków i pętli,
  6. Diagram komponentów, diagram pakietów,
  7. Diagramy stanów i czynności,
  8. Diagramy przepływu danych,
  9. Wędrówka po kodzie, prowadzenie inspekcji kodu w oparciu o przykładowe programy,
  10. Zarządzanie konfiguracją oprogramowania, zapoznanie się z podstawowymi rozwiązaniami.
- **Metody dydaktyczne:**
  W ramach laboratorium realizowane będą małe projekty (realizowane w grupach 3-4) z użyciem narzędzi typu CASE, wspomagających zarządzanie projektem informatycznym. Studenci wykonują diagramy i zadania w ramach zajęć dydaktycznych i pracy własnej.
- **Forma i warunki zaliczenia:**
  Warunkiem zaliczenia jest realizacja wymienionych ćwiczeń laboratoryjnych wraz ze sprawozdaniami oraz wykonanie zadań typu projektowego w ramach pracy własnej w domu.

- **Wykaz literatury podstawowej:**
  1. P.Stevens: UML Inżynieria oprogramowania; Helion Gliwice 2007,
  2. I. Sommerville: Inżynieria oprogramowania, WNT, Warszawa, 2003,
  3. W. Dąbrowski, A. Stasiak, M. Wolski: Modelowanie systemów w języku UML 2.1, PWN, Warszawa 2007,

- **Wykaz literatury uzupełniającej:**
  1. K. Subieta: Wprowadzenie do inżynierii programowania, Wydawnictwo PJWSTK, Warszawa 2002, ISBN 83-89244-00-4.
  2. D. Astels, G. Mills, M. Novak: eXtreme programming: Teoria i praktyka prowadzenia projektów informatycznych, Helion, 2002,
  3. B. Wiszniewski, B. Bereza-Jarociński: Teoria i praktyka testowania programów, PWN, 2007,
  4. M. Flasiński: Zarządzanie projektami informatycznymi, PWN, Warszawa, 2007,
  5. J. Highsmith: APM - Agile Project Management, jak tworzyć innowacyjne produkty, PWN, Warszawa 2007,
  6. A. Jaszkiewicz: Inżynieria oprogramowania, WNT, Warszawa, 2003, ISBN 83-7197-007-2,
  7. M. Klein a.o.: Handbook for Real-Time Analysis, Guide to Rate Monotonic, Analysis for Real – Time Systems, Kluwer Academic Publishers, 1993,
  8. J. Górski (red.): Inżynieria oprogramowania, Mikom, Warszawa, 2000, ISBN 83-7279-028-0,
  9. A. Cockburn: Jak pisać efektywne przypadki użycia, WNT, Warszawa, 2004,
  10. R. Pressman: Software Engineering, McGraw-Hill, New York, 1997.

4. Opis sposobu wyznaczania punktów ECTS

a. forma stacjonarna

Forma zajęć	Formy aktywności studenta	Średnia ilość godzin na zrealizowanie aktywności
Wykład	Kontakt z nauczycielem	25
	Samodzielne studiowanie wskazanej literatury	15

	Samodzielne rozwiązywanie zadań domowych	5
	Przygotowanie do sprawdzianu	5
Laboratorium	Kontakt z nauczycielem	20
	Samodzielne studiowanie wskazanej literatury	15
	Przygotowanie do pracy kontrolnej	15

Całkowita ilość godzin aktywności studenta	100
Liczba punktów ECTS dla modułu	4

b. forma niestacjonarna

Forma zajęć	Formy aktywności studenta	Średnia ilość godzin na zrealizowanie aktywności
Wykład	Kontakt z nauczycielem	15
	Samodzielne studiowanie wskazanej literatury	15
	Samodzielne rozwiązywanie zadań domowych	10
	Przygotowanie do sprawdzianu	10
Laboratorium	Kontakt z nauczycielem	15
	Samodzielne studiowanie wskazanej literatury	15
	Przygotowanie do pracy kontrolnej	20

Całkowita ilość godzin aktywności studenta	100
Liczba punktów ECTS dla modułu	4

5. Wskaźniki sumaryczne

a. forma stacjonarna

a)liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich

- Liczba godzin kontaktowych – 45
- Liczba punktów ECTS – 1,8

b)liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.

- Liczba godzin kontaktowych – 20
- Liczba punktów ECTS – 2,0

b. forma niestacjonarna

a)liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich

- Liczba godzin kontaktowych – 30
- Liczba punktów ECTS – 1,2

b)liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.

- Liczba godzin kontaktowych – 15
- Liczba punktów ECTS – 2,0

6. Zakładane efekty uczenia się

Efekt przedmiotowy (Symbol)	Efekty uczenia się dla przedmiotu	Odniesienie do kierunkowych efektów uczenia się
IO_W01	zna podstawowe zagadnienia inżynierii oprogramowania, w tym modele wytwarzania oprogramowania (przyrostowy, kaskadowy, spiralny, ewolucyjny), metodyki zwinne, w tym TDD (Test Driven Development) oraz zasadnicze problemy i zalecane najlepsze praktyki projektowania. Zna podstawy standardów zarządzenia projektem informatycznym (Scrum, XP) oraz dokumentowania prac projektowych. Zna wybrane narzędzia typu CASE (StarUML, Visual Paradigm, Enterprise Architect)	K_W03 K_W07 K_W13
IO_W02	Zna metody i techniki wytwarzania oprogramowania, w tym etap określania wymagań, analizy, projektowania, implementacji, testowania, wdrożenia konserwacji oraz dokumentowania prac projektowych. Zna perspektywy pojęciową, projektową i implementacyjną oraz elementy poza dziedziną projektową (dot. doboru interfejsu użytkownika, systemu zarządzania bazami danych itp.)	K_W03 K_W07 K_W12
IO_W03	zna zasady konstrukcji przypadków użycia, identyfikacji użytkowników i otoczenia systemu biznesowego. Zna metody tworzenia modelu obiektowego, w tym techniki DDD (Data Driven Design), RDD (Responsibility Driven Design), CRC (Class, Responsibility, Collaborations) oraz techniki ekstrahowania związków generalizacji-specjalizacji i asocjacji.	K_W07 K_W12
IO_W04	zna język modelowania systemów UML, diagramy statyczne (diagramy przypadków użycia, klas, obiektów, pakietów, komponentów, wdrożeń), diagramy dynamiczne (diagramy czynności, stanów, sekwencji, współpracy), związki (powiązania, agregacji, kompozycji, uogólnienia, zależności, użycia, włączenia, rozszerzenia), komunikaty, stereotypy i inne elementy języka.	K_W07 K_W12
IO_W05	zna zasady inspekcji i refaktoryzacji kodu źródłowego. Zna zasady zarządzania konfiguracją oprogramowania oraz metody testowania programów komputerowych (testy jednostkowe, systemowe, akceptacyjne) oraz narzędzia do automatyzacji diagnostyki oprogramowania.	K_W07 K_W12
IO_U01	potrafi zastosować wybrane modele wytwarzania oprogramowania (metodyki tradycyjne i zwinne), umie zidentyfikować zasadnicze problemy projektowe oraz wykorzystać zalecane najlepsze praktyki. Zna wybrane narzędzia typu CASE oraz umie zarządzać pracami projektowymi, w tym wytworzyć poprawną dokumentację.	K_U01 K_U02 K_U04 K_U20
IO_U02	potrafi wykonywać odpowiednie czynności w poszczególnych etapach procesu wytwarzania oprogramowania, umie zastosować perspektywy projektowe oraz uzupełniać projekt dodatkowymi elementami spoza dziedziny projektowej.	K_U02 K_U07 K_U13
IO_U03	potrafi wykonać analizę biznesową przedsięwzięcia, ustalić użytkowników i otoczenie projektowanego systemu, umie wykorzystać metody DDD, RDD i CRC do utworzenia modelu obiektowego. Potrafi poprawnie analizować tekst wymagań oraz modelować związki miedzy bytami (obiektami) systemu.	K_U01 K_U02 K_U11
IO_U04	umie specyfikować wymagania dotyczące oprogramowania oraz przeprowadzić ich przegląd. Potrafi posługiwać się notacją języka UML oraz zaprojektować strukturę i funkcjonalność oprogramowania. Potrafi zaimplementować oprogramowanie w wybranym języku obiektowym (C++, Java, C#) zgodnie z dokumentacją projektową.	K_U02
IO_U05	potrafi dokonać wyboru narzędzi wspomagających budowę oprogramowania, umie wykorzystać inspekcję i refaktoryzację kodu źródłowego, posługiwać się debuggerem, usuwać błędy oprogramowania i je konfigurować. Potrafi tworzyć testy oprogramowania (najmniej jednostkowe CPPUnit, JUnit itp.) oraz je automatyzować.	K_U01 K_U02 K_U11 K_U20 K_U23

IO_K01	potrafi dostrzegać otoczenie pozainformatyczne wytwarzanego oprogramowania, w szczególności jego zgodność z normami prawnymi, a także uwarunkowaniami społecznymi czy ogólnie przyjętymi zasadami współżycia społecznego i dobrymi obyczajami.	K_U13 K_K03
IO_K02	potrafi pracować samodzielnie lub zespołowo, umie wykazać kreatywność lub przewodzić grupie, organizować realizację przedsięwzięcia z zachowaniem bezpieczeństwa, higieny i ergonomii pracy.	K_K02 K_K03
IO_K03	rozumie potrzebę ustawicznego rozwoju intelektualnego, w szczególności w zakresie szybko rozwijającej się dziedziny nowych technologii oraz dziedzinowego języka obcego.	K_U06 K_K01

7. Odniesienie efektów uczenia się do form zajęć i sposób oceny osiągnięcia przez studenta efektów uczenia się

Efekt przedmiotowy (Symbol)	Forma zajęć		Sposób sprawdzenia osiągnięcia efektu
	wykład	laboratorium	
IO_W01	*v*		Praca domowa
IO_W02	*v*		Praca kontrolna
IO_W03	*v*		Praca domowa
IO_W04	*v*		Praca kontrolna
IO_W05	*v*		Sprawdzian wiadomości
IO_U01		*v*	Sprawozdanie z realizacji projektu
IO_U02		*v*	Sprawozdanie z realizacji projektu
IO_U03		*v*	Sprawozdanie z realizacji projektu
IO_U04		*v*	Sprawozdanie z realizacji projektu
IO_U05		*v*	Sprawozdanie z realizacji projektu
IO_K01		*v*	Dyskusja
IO_K02		*v*	Obserwacja pracy studenta lub zespołu projektowego
IO_K03		*v*	Ocena doboru literatury w dokumentacji projektu

8. Kryteria uznania osiągnięcia przez studenta efektów uczenia się

Efekt przedmiotowy (Symbol)	Efekt jest uznawany za osiągnięty gdy:
IO_W01	**Opracowanie** propozycji rozwiązania określonego zadania związanego z projektowaniem systemu informatycznego, przy użyciu wybranego narzędzia wspomagającego projektowanie typu CASE (StarUML, Visual Paradigm, Enterprise Architect).
IO_W02	propozycja rozwiązania niewielkiego zadania, związanego z projektowaniem systemów informatycznych, w szczególności istotnymi są: poprawność analizy tekstu wymagań (identyfikacja bytów), model obiektowy systemu (diagramy i relacje je łączące, przepływ komunikatów, zdarzeń), obecność perspektyw (pojęciowej, projektowej i implementacyjnej) oraz informacji na temat doboru pozadziedzinowego otoczenia (projekt interfejsu użytkownika, dobór narzędzi programistycznych, wybór systemu zarządzania bazą danych, typ projektowanej aplikacji: desktopowa, webowa, mobilna itd.).
IO_W03	Propozycja rozwiązania zadania dot. analizy tekstu wymagań systemu informatycznego z wykorzystaniem technik DDD, RDD i CRC do utworzenia modelu obiektowego. Praca powinna zawierać autorskie obserwacje i wnioski.

IO_W04	Propozycję rozwiązania określonego problemu projektowego przy wykorzystaniu modelowania w notacji UML. Rozwiązanie powinno zawierać diagramy przypadków użycia, klas, aktywności, sekwencji, współpracy, stanów, pakietów, komponentów i wdrożeń. Istotna jest poprawność diagramów i relacji między nimi w odniesieniu do opisowej treści zadania
IO_W05	**Sprawdzian wiadomości** obejmujący zagadnienia formalne języka modelowania UML oraz zadania związane z wykorzystaniem procesów refaktoryzacji kodu źródłowego i procedur testowania.
IO_U01 - IO_U04	**sprawozdanie częściowe**..
IO_U05	Ocenia się postęp w realizacji zadania projektowego - **sprawozdanie całościowe** Sprawozdanie powinno mieć formę elektroniczną (PDF). Istotnym czynnikiem oceny jest dotrzymanie umówionego terminu doręczenia sprawozdania.
IO_K01	Dyskusja na temat otoczenia pozainformatycznego wytwarzanego oprogramowania projektowego, w szczególności jego zgodność z normami prawnymi, a także uwarunkowaniami społecznymi czy ogólnie przyjętymi zasadami współżycia społecznego i dobrymi obyczajami.
IO_K02	Obserwacja pracy studenta i zespołu projektowego.
IO_K03	Ocena doboru literatury w dokumentacji projektu.