# 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 – 25**

**Forma niestacjonarna**

**wykłady – 15**

**laboratorium – 20**

**Zajęcia prowadzone są w języku polskim.**

**Liczba punktów ECTS: 3**

**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. Jak tworzyć i dostarczać w pełni skalowane, skuteczne rozwiązania zarówno w przypadku małych, jak i dużych, złożonych projektach jednocześnie dbając o minimalizowanie zużycia energii oraz zasobów.
Celem przedmiotu jest również zapoznanie studentów z narzędziami i środowiskiem wytwarzania oprogramowania oraz tworzenia modelu na potrzeby opracowywanego projektu. Omówienie sekwencji diagramów w języku UML i informacji które przekazują.

## 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. Podstawowe działania w procesie projektowania: od zebrania wymagań do wdrożenia i ewolucji produktu informatycznego. Stała obecność i komunikacja z przedstawicielem biznesu i szybkie reagowanie na potrzebę zmiany rozwiązania wraz z jego tworzeniem.
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. Minimalna ilość diagramów na potrzeby uznania pracy twórczej za projekt.
7. Mechanizm rozszerzeń UML, narzędzia związane z językiem UML,
8. Diagnostyka oprogramowania; Podstawowe wiadomości o testowaniu: strategia testowania, przygotowania testów, rodzaje testów, automatyzacja procesów testowania
9. Automatyzacja prac projektowych z wykorzystaniem sztucznej inteligencji (AI) w:
    - usprawnianiu operacji zadań,
    - usprawnianiu sposobów interakcji przepływami,
    - możliwości zautomatyzowania powtarzalnych, czasochłonnych zadań.
10. Narzędzia wspomagające projektowanie.

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

**Wykaz literatury uzupełniającej:**

1. D. Astels, G. Mills, M. Novak: *eXtreme programming: Teoria i praktyka prowadzenia projektów informatycznych,* Helion, 2002,
2. B. Wiszniewski, B. Bereza-Jarociński: *Teoria i praktyka testowania programów*, PWN, Warszawa, 2007,
3. J. Cadle, D. Yeates: *Zarządzanie procesem tworzenia systemów informacyjnych*, WNT, Warszawa, 2004,
4. J. Highsmith: *APM - Agile Project Management, jak tworzyć innowacyjne produkty*, PWN, Warszawa 2007,
5. Jaszkiewicz: *Inżynieria oprogramowania*, WNT, Warszawa, 2003, ISBN: 83-7197-007-2,

***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,

3. Analiza i model środowiska, model kontekstowy,
4. Diagram przypadków użycia, 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. Samodzielne wykonanie projektu, wykorzystującego język modelowania UML, dla przykładowych pozyskanych wymagań.

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

- **Wykaz literatury uzupełniającej:**

1. D. Astels, G. Mills, M. Novak: *eXtreme programming: Teoria i praktyka prowadzenia projektów informatycznych*, Helion, 2002,
2. B. Wiszniewski, B. Bereza-Jarociński: *Teoria i praktyka testowania programów*, PWN, 2007,
3. M. Flasiński: *Zarządzanie projektami informatycznymi*, PWN, Warszawa, 2007,
4. J. Highsmith: *APM - Agile Project Management, jak tworzyć innowacyjne produkty*, PWN, Warszawa 2007,
5. A. Jaszkiewicz: *Inżynieria oprogramowania*, WNT, Warszawa, 2003, ISBN 83-7197-007-2,

## 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 rozwiązywanie zadań domowych | 2 |
| | Przygotowanie do sprawdzianu | 3 |
| Laboratorium | Kontakt z nauczycielem | 25 |
| | Samodzielne studiowanie wskazanej literatury | 5 |
| | Przygotowanie do pracy kontrolnej | 9 |
| Konsultacje | Kontakt z nauczycielem | 3 |
| Zal./Egzamin | Kontakt z nauczycielem | 3 |

| | |
|---|---|
| Całkowita ilość godzin aktywności studenta | 75 |
| Liczba punktów ECTS dla modułu | 3 |

### b. forma niestacjonarna

| Forma zajęć | Formy aktywności studenta | Średnia ilość godzin na zrealizowanie aktywności |
|---|---|---|
| Wykład | Kontakt z nauczycielem | 15 |
| | Samodzielne rozwiązywanie zadań domowych | 7 |
| | Przygotowanie do sprawdzianu | 7 |
| Laboratorium | Kontakt z nauczycielem | 20 |
| | Samodzielne studiowanie wskazanej literatury | 5 |
| | Przygotowanie do pracy kontrolnej | 15 |
| Konsultacje | Kontakt z nauczycielem | 3 |
| Zal./Egzamin | Kontakt z nauczycielem | 3 |

| | |
|---|---|
| Całkowita ilość godzin aktywności studenta | 75 |
| Liczba punktów ECTS dla modułu | 3 |

## 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 – 56
- Liczba punktów ECTS – 2,2

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

- Liczba godzin kontaktowych – 25
- Liczba punktów ECTS – 1,6

### b. forma niestacjonarna

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

- Liczba godzin kontaktowych – 41
- Liczba punktów ECTS – 1,6

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

- Liczba godzin kontaktowych – 20
- Liczba punktów ECTS – 1,6

## 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<br>K_W07<br>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<br>K_W07<br>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<br>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<br>K_W12 |
| IO_W05 | Zna metody testowania programów komputerowych (testy jednostkowe, systemowe, akceptacyjne) oraz narzędzia do automatyzacji diagnostyki oprogramowania. | K_W07<br>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<br>K_U02<br>K_U04<br>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<br>K_U07<br>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<br>K_U02<br>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, posługiwać się debuggerem, usuwać błędy oprogramowania i je konfigurować. Potrafi tworzyć testy oprogramowania (najmniej jednostkowe CPPUnit, JUnit itp.) oraz je automatyzować. | K_U01<br>K_U02<br>K_U11<br>K_U20<br>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<br>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<br>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<br>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 (np. StarUML). |
| 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) |
| 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 |
| IO_U01 - IO_U04 | **sprawozdanie częściowe** |
| IO_U05 | Ocenia się postęp w realizacji zadania projektowego - **sprawozdanie całościowe**<br>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. |