METODYKI PROGRAMOWANIA

Kod przedmiotu: IO-MP2

Rodzaj przedmiotu: kierunkowy, obieralny

Specjalność: Inżynieria oprogramowania

Wydział: Informatyki

Kierunek: Informatyka

Poziom studiów: drugiego stopnia – VII poziom PRK

Profil studiów: praktyczny

Forma studiów: stacjonarna/niestacjonarna

Rok: 1

Semestr: 1

Formy zajęć i liczba godzin:

Forma stacjonarna

wykłady – 15

laboratorium – 35

Forma niestacjonarna

wykłady – 10

laboratorium – 18

Zajęcia prowadzone są w języku polskim.

Liczba punktów ECTS: 3

Osoby prowadzące:

wykład:
laboratorium:

---

1. Założenia i cele przedmiotu:

Proces realizacji współczesnego oprogramowania wymaga systematycznego i zdyscyplinowanego podejścia. Podejście to jednak powinno być jednocześnie elastyczne, podatne na zmiany specyfikacji, wczesną detekcję błędów oraz pracę grupową. Wytwarzaniem oprogramowania sterują współczesne metodyki, pozwalające na pogodzenie tych pozornie rozbieżnych celów. Celem modułu Metodyki programowania jest przekazanie wiedzy na temat współczesnych metodyk tworzenia oprogramowania oraz wyrobienie umiejętności ich skutecznego wykorzystania w praktyce pracy programisty. Moduł łączy wiedzę z zakresu metodyk programowania z umiejętnościami ich praktycznego stosowania.

2. Określenie przedmiotów wprowadzających wraz z wymaganiami wstępnymi:

Przedmioty wprowadzające to: Programowanie obiektowe.

3. Opis form zajęć

a) *Wykłady*

- **Treści programowe:**
  - Metodyki paradygmaty programowania.
  - Paradygmat strukturalny, obiektowy, funkcyjny, deklaratywny, obszary zastosowań, wady, zalety.
  - Klasyczne i zwinne metodyki programowania.
  - Dokumentowanie prac projektowych.
  - Metryki jakości oprogramowania.
  - Metody testowania systemów informatycznych, metodyki sterowane testami.
  - Inżynieria odwrotna.
  - Repozytoria kodu, systemy kontroli wersji.
  - Integracja ciągła integracja i ciągła inspekcja kodu.
- **Metody dydaktyczne:**
  - Wykład prowadzony metodą tradycyjną z wykorzystaniem rzutnika multimedialnego, z wykorzystaniem materiałów udostępnianych studentom w postaci elektronicznej.
- **Forma i warunki zaliczenia:**
  - Pozytywna ocena testu egzaminacyjnego.
- **Wykaz literatury podstawowej:**
  1. S. Haridi, P. Van Roy: Programowanie. Koncepcje, techniki i modele. Helion.
  2. A. Graham: Metody obiektowe w teorii i praktyce. WNT.
  3. U. Nilsson, J. Małuszyński: Logic, programming and Prolog. Wiley.
  4. A. Elssamadisy, Agile. Wzorce wdrażania praktyk zwinnych, Helion.
  5. Martin R.C.: Czysta architektura. Struktura i design oprogramowania. Przewodnik dla profesjonalistów. Gliwice: Helion, copyright 2018.
  6. Mariusz Chrapko, Scrum. O zwinnym zarządzaniu projektami, Helion, Wyd II
- **Wykaz literatury uzupełniającej:**
  1. J. Suereth: Scala od podszewki. Helion.
  2. K. Beck, TDD. Sztuka tworzenia dobrego kodu, Helion.

b) *Laboratorium*

- **Treści programowe:**
  - Analiza zakresu stosowalności istniejących paradygmatów programowania, ćwiczenia praktyczne.
  - Dziedzina problemu, specyfikacja wymagań a wybór paradygmatu programowania.
  - Paradygmaty programowania a metodyki programowania.
  - Klasyczne i zwinne metodyki programowania, wady zalety, ćwiczenia praktyczne.
  - Metody testowania systemów informatycznych, narzędzia wspierające proces testowania.
  - Automatyzacja testów, środowiska, narzędzia.
  - Wykrywanie i śledzenie błędów czasu wykonywania programu.

- Praktyczne aspekty wykorzystania repozytoriów kodu i systemów kontroli wersji.
- Integracja ciągła integracja i ciągła inspekcja kodu.

- **Metody dydaktyczne:**
  - Prezentacja treści i dyskusja moderowana.
  - Metoda problemowa – studium przypadku, burza mózgów.
  - Metoda laboratoryjna −ćwiczenia laboratoryjne z wykorzystaniem komputerów.
- **Forma i warunki zaliczenia:**
  - Pozytywna ocena realizacji wskazanych zadań programistycznych.
  - Pozytywna ocena aktywności studenta podczas zajęć, w tym ocena biegłości w tworzeniu programów warstwy serwerowej i posługiwaniu się odpowiednimi narzędziami programistycznymi.
- **Wykaz literatury podstawowej:**
  1. A. Roman, Testowanie i jakość oprogramowania. Metody, narzędzia, techniki, Helion.
  2. W. Gajda, Git. Rozproszony system kontroli wersji, Helion.
  3. *Rajani R.: Testowanie kodu w praktyce. Gliwice: Helion, cop. 2018*

- **Wykaz literatury uzupełniającej:**
  - Jak w przypadku wykładu.

4. Opis sposobu wyznaczania punktów

a. forma stacjonarna

Forma zajęć	Formy aktywności studenta	Średnia liczba godzin na zrealizowanie aktywności
Wykład	Kontakt z nauczycielem	15
	Przygotowanie do egzaminu	10
Laboratorium	Kontakt z nauczycielem	35
	Przygotowanie do pracy kontrolnej	5
	Samodzielne rozwiązywanie zadań	10

Całkowita ilość godzin aktywności studenta	**75**
Liczba punktów ECTS dla modułu/przedmiotu	**3**

b. forma niestacjonarna

Forma zajęć	Formy aktywności studenta	Średnia liczba godzin na zrealizowanie aktywności
Wykład	Kontakt z nauczycielem	10
	Przygotowanie do egzaminu	15
Laboratorium	Kontakt z nauczycielem	18
	Przygotowanie do pracy kontrolnej	15
	Samodzielne rozwiązywanie zadań	17

Całkowita ilość godzin aktywności studenta	75
Liczba punktów ECTS dla modułu/przedmiotu	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 – 50
   - Liczba punktów ECTS – 2,0

b) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.
   - Liczba godzin kontaktowych – 35
   - 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 – 28
   - Liczba punktów ECTS – 1,2

b) liczba godzin dydaktycznych (tzw. kontaktowych) i liczba punktów ECTS na zajęciach o charakterze praktycznym.
   - Liczba godzin kontaktowych – 18
   - 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-MP2_W1	Student zna koncepcję podstawowych paradygmatów programowania, rozumie ich zakres zastosowań, zna i rozumie pojęcie metodyki programowania.	IIK_W03 IIK_W04
IO-MP2_W2	Student zna i podstawowe metodyki programowania, potrafi ocenić ich wady i zalety, potrafi dobrać rodzaj metodyki do specyfiki realizowanego projektu.	IIK_W05 IIK_W08
IO-MP2_U1	Student potrafi wykorzystywać w praktyce poznane paradygmaty i metodyki programowania.	IIK_U06 IIK_U12 IIK_U15
IO-MP2_U2	Student potrafi wykorzystywać narzędzia wspomagające proces tworzenia, testowania i integracji współdzielonego kodu.	IIK_U06 IIK_U07 IIK_U15
IO-MP2_K1	Student posiada kompetencje w zakresie metodycznej pracy grupowej nad projektem, rozumie rolę jakości oprogramowania.	IIK_K01 IIK_K02 IIK_K03

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-MP2_W1	*v*		Egzamin
IO-MP2_W2	*v*	*v*	Egzamin
IO-MP2_U1	*v*	*v*	Prace kontrolne
IO-MP2_U2		*v*	Prace kontrolne
IO-MP2_K1		*v*	Prace kontrolne, ocena aktywności

8. Kryteria uznania osiągnięcia przez studenta efektów uczenia się.

Efekt przedmiotowy (Symbol)	Efekt jest uznawany za osiągnięty, gdy student:
IO-MP2_W1	Poprawnie odpowiada na ponad 50% pytań testu egzaminacyjnego
IO-MP2_W2	Poprawnie odpowiada na ponad 50% pytań testu egzaminacyjnego
IO-MP2_U1	Osiąga ponad 50% punktów w pracach kontrolnych.
IO-MP2_U2	Osiąga ponad 50% punktów w pracach kontrolnych.
IO-MP2_K1	Osiąga ponad 50% punktów w pracach kontrolnych.