Podstawy modelowania programów - opis przedmiotu

Informacje ogólne

Nazwa przedmiotu	Podstawy modelowania programów
Kod przedmiotu	11.3-WI-INFP-PMP
Wydział	Wydział Informatyki, Elektrotechniki i Automatyki
Kierunek	Informatyka
Profil	ogólnoakademicki
Rodzaj studiów	pierwszego stopnia z tyt. inżyniera
Semestr rozpoczęcia	semestr zimowy 2019/2020

Informacje o przedmiocie

Semestr	5
Liczba punktów ECTS do zdobycia	7
Typ przedmiotu	obieralny
Język nauczania	polski
Sylabus opracował	• dr inż. Łukasz Hładowski

Formy zajęć

Forma zajęć	Liczba godzin w semestrze (stacjonarne)	Liczba godzin w tygodniu (stacjonarne)	Liczba godzin w semestrze (niestacjonarne)	Liczba godzin w tygodniu (niestacjonarne)	Forma zaliczenia
Wykład	30	2	18	1,2	Egzamin
Laboratorium	30	2	18	1,2	Zaliczenie na ocenę
Projekt	15	1	9	0,6	Zaliczenie na ocenę

Cel przedmiotu

- ukształtowanie podstawowych umiejętności w zakresie modelowania oprogramowania

- zapoznanie studentów z praktycznym zastosowaniem modelowania programów dla prostych systemów informatycznych

- zapoznanie studentów z prawidłowym sposobem praktycznej implementacji rozwiązania prostego problemu przy wykorzystaniu podstawowych wzorców projektowych

Wymagania wstępne

Programowanie obiektowe, Inżynieria oprogramowania

Zakres tematyczny

Zagadnienia wprowadzające. Tło i historia współczesnych technik modelowania. Zunifikowany proces cyklu życia aplikacji. Analiza i projektowanie systemowe. Paradygmat obiektowości. Modelowanie obiektowe i jego rola w projektowaniu systemów informatycznych. Diagramy Klasa-Odpowiedzialność-Współpraca (ang. Class-Responsibility-Collaboration, CRC). Procesy wytwarzania oprogramowania. Wstęp do notacji i diagramów Zunifikowanego Języka Modelowania (UML). Geneza i cele powstania UML. Zakres i struktura warstwowa UML. Modelowanie strukturalne. Podstawowe pojęcia i elementy obiektowości. Klasy, obiekty, abstrakcja, enkapsulacja, dziedziczenie, polimorfizm, komunikacja, relacje i powiązania między obiektami. Statyczne diagramy strukturalne: klas i obiektów. Modelowanie asocjacji między klasami. Agregacja, kompozycja, generalizacja, specjalizacja, zależności i realizacje. Pakiety i podsystemy. Typy, interfejsy i klasy implementacji Diagramy implementacyjne: komponentów i wdrożenia. Wymagania i ich specyfikacja. Diagramy przypadków użycia. Analiza i związki między przypadkami użycia: zawieranie, rozszerzenie, grupowanie i uogólnienie. Modelowanie behawioralne. Diagramy sekwencji i kolaboracji. Role, komunikaty i bodźce. Interakcje i kolaboracje. Analiza stanów systemu. Diagramy stanów i aktywności. Przejścia przepływu. Decyzje. Współbieżność. Sygnały i komunikacja. Wzorce projektowe. Formułowanie problemów programistycznych. Przegląd najczęściej stosowanych wzorców konstrukcyjnych, strukturalnych i behawioralnych. Tworzenie i testowanie wzorców. Zagadnienia praktyczne. Praca nad przypadkami użycia. Ogólne spojrzenie na projektowanie, wdrażanie i testowanie. Przegląd wybranych narzędzi projektowania w UML-u.

Metody kształcenia

wykład: burza mózgów, dyskusja, zajęcia praktyczne, wykład konwencjonalny

laboratorium: burza mózgów, praca z dokumentem źródłowym, dyskusja, praca w grupach, zajęcia praktyczne, wykład konwencjonalny

projekt: burza mózgów, praca z dokumentem źródłowym, dyskusja, praca w grupach, zajęcia praktyczne, wykład konwencjonalny

Efekty uczenia się i metody weryfikacji osiągania efektów uczenia się

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć
Rozumie potrzebę wykonywania testów jednostkowych i potrafi dokonać ich implementacji dla prostych przypadków oraz umie wykorzystać podstawowe narzędzia programistyczne używane do takich testów.	• K_W19 • K_K01	• projekt	• Wykład • Projekt

Opis efektu	Symbole efektów	Metody weryfikacji	Forma zajęć
Potrafi zaimplementować fragment prostego systemu w wybranym języku programowania przy wykorzystaniu wzorców projektowych i właściwych technik programowania zorientowanego obiektowo.	• K_W19 • K_K01	• kolokwium • projekt	• Projekt
Posługuje się językiem UML do opisu i formułowania problemów programistycznych.	• K_W19 • K_K01	• kolokwium • projekt • sprawdzian	• Wykład • Laboratorium • Projekt
Potrafi zaimplementować proste wzorce projektowe w wybranym języku programowania. Zna wady i zalety wykorzystania wybranego wzorca oraz potrafi zaproponować rozwiązanie alternatywne	• K_W19 • K_K01	• kolokwium • projekt • sprawdzian	• Wykład • Laboratorium • Projekt
Potrafi dostrzec i omówić wady i zalety zaproponowanego rozwiązania programistycznego na etapie jego modelowania w UML.	• K_W19 • K_U01 • K_K01	• projekt	• Projekt
Zna, rozumie i stosuje podstawowe zasady programistyczne inżynierii oprogramowania.	• K_W19 • K_K01	• kolokwium • projekt • sprawdzian	• Wykład • Laboratorium • Projekt
Umie wybrać narzędzia wspomagające budowę oprogramowania.	• K_W19 • K_U01 • K_K01	• bieżąca kontrola na zajęciach • sprawdzian	• Laboratorium
Potrafi dostrzec i omówić wady i zalety zaproponowanego rozwiązania programistycznego poprzez analizę kodu źródłowego. Potrafi dokonać refaktoryzacji prostego kodu.	• K_W19 • K_K01	• projekt	• Wykład • Projekt

Warunki zaliczenia

Wykład - warunkiem zaliczenia jest uzyskanie pozytywnej oceny z egzaminu przeprowadzonego w formie zaproponowanej przez prowadzącego.

Laboratorium - warunkiem zaliczenia jest uzyskanie pozytywnych ocen ze wszystkich zadań przewidzianych do realizacji w ramach programu laboratorium.

Projekt - warunkiem zaliczenia jest uzyskanie pozytywnych ocen ze wszystkich zadań projektowych, przewidzianych do realizacji w ramach zajęć projektowych.

Składowe oceny końcowej = wykład: 40% + laboratorium: 20% + projekt: 40%

Literatura podstawowa

1. Martin R.C.: Czysty kod. Podręcznik dobrego programisty, Helion, Gliwice 2010
2. Beck K.: TDD. Sztuka tworzenia dobrego kodu, Helion, Gliwice 2014
3. Freeman E., Freeman E., Bates B., Sierra K.:Wzorce projektowe – Rusz głową!, Helion, Gliwice 2010
4. Larman C.: UML i wzorce projektowe. Analiza i projektowanie obiektowe oraz iteracyjny model wytwarzania aplikacji. Wydanie III, Helion, Gliwice 2011
5. Cheesman J., Daniels J.: Komponenty w UML, WNT, Warszawa, 2004

Literatura uzupełniająca

1. Martin R.C., Martin M.: Agile. Programowanie zwinne: zasady, wzorce i praktyki zwinnego wytwarzania oprogramowania w C#, Helion 2008
2. Seidl, M., Scholz, M.: UML @ Classroom: An Introduction to Object-Oriented Modeling, Springer 2015
3. Way J.: Laravel Testing Decoded, Leanpub 2013

Uwagi

Zmodyfikowane przez dr inż. Łukasz Hładowski (ostatnia modyfikacja: 06-05-2019 11:16)

Wygenerowano automatycznie z systemu SylabUZ