**Nazwa przedmiotu:**

Sztuka wytwarzania oprogramowania

**Koordynator przedmiotu:**

Rajmund Kożuszek

**Status przedmiotu:**

Fakultatywny ograniczonego wyboru

**Poziom kształcenia:**

Studia II stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Przedmioty techniczne - podstawowe

**Kod przedmiotu:**

SWO

**Semestr nominalny:**

1 / rok ak. 2021/2022

**Liczba punktów ECTS:**

4

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

1. Liczba godzin kontaktowych - 50 godz., w tym
a. obecność na wykładach 30 godz.
b. obecność na zajęciach laboratoryjnych 16 godz.
c. udział w konsultacjach związanych z realizacją przedmiotu 4 godz.
2. Praca własna studenta - 60 godz., w tym:
a. analiza literatury, dokumentacji, materiałów wykładowych związana z przygotowaniem do kolejnych wykładów: 20 godz.
b. instalacja oprogramowania, analiza dokumentacji, eksperymenty, związane z przygotowaniem się do laboratorium: 30 godz.
c. przygotowanie do kolokwiów: 10 godz.
Łączny nakład pracy studenta wynosi 110 godz, co odpowiada 4 pkt. ECTS

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

2 pkt. ECTS, 50 godz. kontaktowych

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

1,75 pkt. ECTS, 30 + 16 = 46 godz. realizacji ćwiczeń

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład: | 30h |
| Ćwiczenia: | 0h |
| Laboratorium: | 15h |
| Projekt: | 0h |
| Lekcje komputerowe: | 0h |

**Wymagania wstępne:**

Programowanie obiektowe, Systemy operacyjne

**Limit liczby studentów:**

150

**Cel przedmiotu:**

Przedmiot ten ma na celu nauczenie podstaw technik i dobrych praktyk stosowanych w przemysłowym wytwarzaniu oprogramowania. Nacisk położony jest na pracę projektanta kodu i programisty rozbudowanego systemu informatycznego, ale zakres przedmiotu wykracza poza rozwiązywanie problemów programistycznych, ukazując kodowanie jako element większej całości. Student po zakończeniu przedmiotu będzie mógł wykazać się wiedzą z zakresu stosowanych rozwiązań inżynieryjnych i metodyk pracy wykorzystywanych w procesie wytwarzania i utrzymania złożonych systemów informatycznych. Student zdobędzie umiejętności z zakresu ustandaryzowanego rozwiązywania częstych problemów inżynieryjnych, oceny jakości kodu i metod jej poprawy, pracy nad istniejącym kodem oraz technik stosowanych dla długofalowego utrzymania wysokiej wiarygodności i jakości oprogramowania. Student zapozna się także z różnorodnymi narzędziami i kompleksowymi środowiskami wspomagającymi proces wytwarzania oprogramowania oraz technikami pracy w zespole programistycznym.

**Treści kształcenia:**

WYKŁADY:
1. Wprowadzenie (2 godz.)
Regulamin i informacje o przedmiocie. Potok wytwarzania oprogramowania - powtórzenie i rozwinięcie, od edytora, poprzez repozytorium kodu do narzędzi do ciągłego integrowania i dostarczania oprogramowania (CI/CD). Omówienie typowych narzędzi, git, Jenkins.
2. Styl kodowania (4 godz.)
Techniki pracy w zespole programistów (rewizje kodu i mentor), rola testów w procesie wytwarzania oprogramowania, testy jednostkowe. Zaawansowane techniki stosowane w programowaniu obiektowym - SOLID. Czysty kod (Clean Code, Lean Development). Tworzenie oprogramowania w oparciu o testy (TDD).
3. Obiektowe wzorce projektowe (4 godz.)
Zalety stosowania wzorców projektowych i poprawne ich aplikowania. Antywzorce. Omówienie wzorców: kompozyt, most, fabryki, singleton, obserwator, model-widok-kontroler, dekorator, adapter, iterator, strategia, wizytator, wielometoda, komenda, fasada i inne.
4. Współbieżne wzorce projektowe (6 godz).
Stosowanie wzorców do rozwiązywania problemów współbieżnych. Asynchroniczność a wielowątkowość i zrównoleglenie wykonania. Pętla zdarzeń, asynchroniczna obsługa urządzeń i asynchroniczne przetwarzanie danych. Skalowalność. Komunikacja międzywątkowa i międzyprocesowa.
5. Monitorowanie aplikacji (1 godz.)
Logowanie i śledzenie pracy programu. Automatyczne reagowanie na błędy.
6. Analiza aplikacji (2 godz.)
Debugowanie, profilowanie i badanie uruchomionej aplikacji. Automatyzacja wykrywania problemów czasu wykonania.
7. Analiza kodu (2 godz.)
Analiza statyczna kodu, metryki kodu, pomiar pokrycia, ocena jakości kodu i testów.
8. Refaktoring (4 godz.)
Wykrywanie “zapachów” kodu, techniki poprawy jakości kodu. Utrzymywanie dużych projektów i praca z kodem zastanym.
9. Systemy heterogeniczne (2 godz.)
Tworzenie aplikacji, gdzie różne moduły są implementowane w różnych językach programowania, przykład - łączenie Pythona i C++, Javy i C++, R i C++. Różne techniki łączenia systemów, problemy architektoniczne i techniczne.
10. Zasoby internetowe (1 godz.)
Konferencje i czasopisma poświęcone programowaniu, standardy języków programowania. Umiejętne wykorzystanie dostępnych zasobów.
LABORATORIA:
1. Zapoznanie się z podstawowymi narzędziami. Zaawansowane wykorzystanie narzędzia git. Obsługa przygotowanego potoku. Praca w parach i rewizja kodu.
2. Przygotowanie własnego projektu i środowiska dla niego. Zestawienie potoku CI/CD dla projektu, z testami i walidacją.
3. Profilowanie i debugowanie złożonej aplikacji. Poszukiwanie przyczyn błędów i wykrywanie miejsc potencjalnej optymalizacji.
4. Refaktoring istniejącego kodu. Praca z zastanym kodem. Walidacja działania z wykorzystaniem istniejącego potoku.

**Metody oceny:**

Realizacja przedmiotu obejmuje następujące formy zajęć:
- wykład - 2 godziny tygodniowo, na którym, oprócz tradycyjnego omawiania zagadnień z wykorzystaniem slajdów, będą prezentowane przykłady tworzenia kodu źródłowego kodu w czasie rzeczywistym
- zajęcia laboratoryjne w wymiarze 4 godz.; w ramach zajęć studenci samodzielnie, pod opieką prowadzącego, w grupach dwuosobowych będą realizować wskazane ćwiczenia.
Sprawdzanie założonych efektów kształcenia realizowane jest przez:
- ocenę umiejętności i wiedzy wykazanych w trakcie realizacji zadań laboratoryjnych,
- ocenę wiedzy wykazanej na dwóch kolokwiach pisemnych.

**Egzamin:**

nie

**Literatura:**

1. Andrew Hunt, David Thomas - Pragmatyczny programista. Od czeladnika do mistrza. Helion 2014
2. Robert C. Martin - Czysty kod. Podręcznik dobrego programisty. Helion 2010
3. Robert C. Martin - Mistrz czystego kodu. Kodeks postępowania profesjonalnych programistów. Helion 2013
4. Erich Gamma et al. - Wzorce projektowe. Helion 2010
5. Martin Fowler - Refaktoryzacja. Ulepszanie struktury istniejącego kodu. Helion 2011

**Witryna www przedmiotu:**

https://usosweb.usos.pw.edu.pl/kontroler.php?\_action=katalog2/przedmioty/pokazPrzedmiot&prz\_kod=103A-INxxx-ISP-SWO

**Uwagi:**

(-)

## Charakterystyki przedmiotowe