KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Narzędzia inżynierii programowania [S1MiKC2>NIP]

Przedmiot

Kierunek studiów
Mikroelektronika i komunikacja cyfrowa

Rok/Semestr
1/2

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
0

Laboratorium
20

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

2,00

Koordynatorzy

dr inż. Łukasz Kułacz
lukasz.kulacz@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student powinien posiadać podstawową wiedzę z zakresu programowania, w szczególności pojęcia zmiennych, klas i funkcji. Powinien posiadać umiejętność implementacji prostych programów i dostrzegać ryzyko związane z tworzeniem niedopracowanego oprogramowania.

Cel przedmiotu

Celem przedmiotu jest przekazanie studentom podstawowej wiedzy dotyczącej narzędzi wykorzystywanych podczas tworzenia oprogramowania. Narzędzia te dotyczą przede wszystkim przechowywania i kontroli wersji kodu w lokalnych i zdalnych repozytoriach, współpracy nad tym samym kodem, technik zapewnienia jakości poprzez testy o różnym poziomie szczegółowości i skomplikowania, podstaw metod ciągłej integracji i ciągłego wdrażania oraz rozszerzeń do edytorów kodu ułatwiających tworzenie dobrej jakości oprogramowania. W ramach zajęć poruszane są również podstawowe zagadnienia związane z wykorzystaniem profilera oraz debuggera kodu.

Przedmiotowe efekty uczenia się

Wiedza:
Student ma podstawową wiedzę teoretyczną i praktyczną w zakresie repozytoriów oprogramowania,

testów oprogramowania, automatyzacji procesu testowania i zasad tworzenia poprawnego i czytelnego kodu. Student zna podstawowe narzędzia ułatwiające prace nad oprogramowaniem.

Umiejętności:
Student potrafi efektywnie wykorzystać do pracy lokalne repozytorium oprogramowania, a także synchrozniować efefy swojej pracy poprzez zdalne repozytorium. Student potrafi współpracować z innymi twórcami oprogramowania poprzez zdalne repozytoria i łączyć z nimi swoje oprogramowanie. Student potrafi przygotować podstawowe testy do kodu i przetestować swój kod zarówno lokalnie jak i poprzez automatyzację połączoną z zdalnym repozytorium oprogramowania. Student potrafi również wykorzystać podstawowe narzędzia do ułatwienia pracy nad tworzeniem oprogramowania i tym samym poprawić jakość tworzonego kodu. Student potrafi użyć profiler do oceny działania kodu oraz debugger do analizy działania kodu.

Kompetencje społeczne:
Student ma świadomość możliwości i ograniczeń podczas tworzenia oprogramowania, a w szczególności konieczność zapewnienia dobrej jakości kodu. Rozumie potencjalny wpływ niepoprawnie przygotowanego oprogramownia. Jest świadomy wyzwań i zagrożeń związanych z współpracą wielu programistów nad tym samym kodem.

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

W zakresie laboratorium weryfikowanie założonych efektów kształcenia realizowane jest przez: ocenę merytoryczną zadań indywidualnych lub grupowych wykonywanych w ramach zajęć lub w postaci zadań do wykonania po zajęciach, aktywność na zajęciach. Zadania w postaci poleceń do wykonania poprzez implementacje określonych w zadaniu funkcjonalności, dla których przypisana jest liczba punktów do uzyskania. Przejęta skala ocen:
2.0 dla <0%; 50%>
3.0 dla (50%; 60%>
3.5 dla (60%; 70%>
4.0 dla (70%; 80%>
4.5 dla (80%; 90%>
5.0 dla (90%; 100%>

Treści programowe

W ramach przedmiotu student zapozna się z różnymi dostępnymi narzędziami wspomagającymi tworzenie kodu i prowadzenie projektów teleinformatycznych zgodnie z dobrymi praktykami. Narzędzia te mają zarówno umożliwiać prace wieloplatformową, jak i w zespole wieloosobowym. W szczególności student pozna narzędzia do kontroli wersji i wybrane repozytoria, systemy automatycznego testowania, koncepcję mikrousług i CI/CD, testowania jakości kodu i pokrycia kodu testami.

Tematyka zajęć

Laboratoria:
1. Podstawy systemu git. (1 jednostka zajęć)
2. Współpraca z wykorzystaniem systemu git. (2 jednostki zajęć)
3. Przygotowanie testów jednostkowych i integracyjnych. (2 jednostki zajęć)
4. Przygotowanie testów funkcjonalnych i wydajności. (2 jednostki zajęć)
5. Kontrola jakości kodu i pokrycie kodu testami. (1 jednostka zajęć)
6. Narzędzia ułatwiające pracę podczas tworzenia oprogramowania. (2 jednostki zajęć)

Metody dydaktyczne

Laboratoria będą prowadzone w formie praktycznych zajęć przy komputerze, podczas których studenci będą mieli okazję samodzielnie implementować rozwiązania zgodnie z dostarczonymi instrukcjami. Część zadań przewidziane jest do realizacji w kilkuosobowych grupach. W czasie realizacji zajęć przewidziane są również konsultacje z prowadzącym, które umożliwią studentom uzyskanie wskazówek oraz omówienie trudniejszych zagadnień związanych z implementacją. Poza zajęciami studenci mają możliwość skorzystania z dodatkowych konsultacji z prowadzącym.

Literatura

Podstawowa:
1. Jakość oprogramowania. Podręcznik dla profesjonalistów. Michał Sobczak
2. TDD w praktyce. Niezawodny kod w języku Python. Harry Percival
3. Git i GitHub. Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej. Mariot Tsitoara

Uzupełniająca:
1. Testowanie i jakość oprogramowania. Modele, techniki, narzędzia. Adam Roman

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	60	2,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	20	0,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	40	1,50