KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu

Organizacja procesu wytwarzania oprogramowania

Przedmiot

Kierunek studiów

Informatyka

Studia w zakresie (specjalność)

Inteligentne technologie informatyczne

Poziom studiów

drugiego stopnia

Forma studiów

stacjonarne

Rok/semestr

1/1

Profil studiów

ogólnoakademicki

Język oferowanego przedmiotu

polski

Wymagalność

obligatoryjny

Liczba godzin

Wykład	Laboratoria	Inne (np. online)
15	15	
Ćwiczenia	Projekty/seminaria	

Liczba punktów ECTS

2

Wykładowcy

Odpowiedzialny za przedmiot/wykładowca:

dr inż. Marcin Szeląg

e-mail: marcin.szelag@cs.put.poznan.pl

tel. 61 665 3023

Instytut Informatyki

ul. Piotrowo 2, 60-965 Poznań

Odpowiedzialny za przedmiot/wykładowca:

Wymagania wstępne

Efekty kształcenia ze studiów I stopnia zdefiniowane w Uchwale Senatu w sprawie zatwierdzenia kierunkowych efektów kształcenia dla studiów prowadzonych na Politechnice Poznańskiej nr 42 z dnia 24 kwietnia 2017 roku, a szczególnie efekty K1st_W1-8, weryfikowane w procesie rekrutacji na studia 2 stopnia - efekty te prezentowane są w serwisie internetowym wydziału www.fc.put.poznan.pl

Cel przedmiotu

1. Przekazanie studentom wiedzy w zakresie: systemów zarządzania wersjami oprogramowania, zwinnych metod szacowania pracochłonności w procesie wytwarzania oprogramowania oraz automatyzacji budowania, ciągłej integracji i wdrażania oprogramowania.

2. Rozwijanie u studentów umiejętności wykorzystania w procesie wytwarzania oprogramowania odpowiednich technik i narzędzi.

3. Kształtowanie u studentów kompetencji w zakresie świadomego włączania do tworzonego oprogramowania zewnętrznych modułów i bibliotek, z poszanowaniem zasad ich licencjonowania.

Przedmiotowe efekty uczenia się

Wiedza

1. ma zaawansowaną i pogłębioną wiedzę z zakresu wybranych narzędzi wykorzystywanych do ciągłej integracji, budowania i wersjonowania systemów informatycznych - [K2st_W1]

2. ma zaawansowaną wiedzę szczegółową dotyczącą schematów rozwoju oprogramowania z użyciem systemu kontroli wersji Git (ang. Git workflows) - [K2st_W3]

3. ma zaawansowaną wiedzę szczegółową dotyczącą ciągłej integracji i ciągłego wdrażania oprogramowania - [K2st_W3]

4. ma zaawansowaną wiedzę szczegółową dotyczącą automatyzacji procesu budowania oprogramowania z użyciem systemu Gradle - [K2st_W3]

5. ma zaawansowaną i szczegółową wiedzę o procesie tworzenia wydania (ang. release) oprogramowania z wykorzystaniem systemu kontroli wersji Git - [K2st_W5]

6. ma zaawansowaną i szczegółową wiedzę o procesie wdrażania oprogramowania z wykorzystaniem lekkich kontenerów programowych na platformie Docker - [K2st_W5]

Umiejętności

1. potrafi efektywnie używać systemu wersjonowania Git do zarządzania konfiguracją oprogramowania - [K2st_U2]

2. potrafi budować aplikacje w oparciu o komunikujące się ze sobą mikrousługi, realizowane z użyciem kontenerów programowych na platformie Docker - [K2st_U5]

3. potrafi ocenić przydatność i możliwość wykorzystania nowych narzędzi informatycznych do ciągłej integracji i budowania oprogramowania - [K2st_U6]

4. potrafi poprawnie użyć metodę delficką do szacowania pracochłonności zadań w procesie wytwarzania oprogramowania - [K2st_U7]

5. potrafi ocenić przydatność narzędzi wykorzystywanych do budowania, ciągłej integracji i wdrażania oprogramowania - [K2st_U9]

6. potrafi zautomatyzować proces budowania i tworzenia wersji dystrybucyjnej oprogramowania z użyciem systemu Gradle - [K2st_U11]

Kompetencje społeczne

1. rozumie, że wiedza i umiejętności związane z budowaniem i wdrażaniem oprogramowania szybko stają się przestarzałe - [K2st_K1]

2. rozumie konieczność śledzenia trendów w zakresie narzędzi do wersjonowania, budowania, ciągłej integracji i wdrażania oprogramowania - [K2st_K2]

3. ma świadomość konieczności przestrzegania zasad licencji modułów i bibliotek zewnętrznych włączanych do tworzonego oprogramowania - [K2st_K4]

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty kształcenia przedstawione wyżej weryfikowane są w następujący sposób:

Ocena formująca:

a) w zakresie wykładów:

- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach,

b) w zakresie laboratoriów / ćwiczeń:

- na podstawie oceny bieżącego postępu realizacji zadań.

Ocena podsumowująca:

a) w zakresie wykładów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenę wiedzy i umiejętności wykazanych na kolokwium zaliczeniowym podczas ostatniego wykładu, składającym się z pytań zamkniętych (wielokrotnego wyboru), obejmujących łącznie cały zakres wykładu. Zaliczenie (ocena 3,0) wymaga zdobycia minimum 6 z 12 pkt.

- omówienie wyników kolokwium,

b) w zakresie laboratoriów weryfikowanie założonych efektów kształcenia realizowane jest przez:

- ocenianie ciągłe, na poszczególnych zajęciach - weryfikacja dostatecznego stopnia realizacji zadań zaplanowanych na zajęcia, adnotacja niewystarczającej realizacji zadań spowodowanej brakiem zaangażowania,

- ocena wybranego zadania dotyczącego przerobionych zagadnień, realizowanego częściowo na laboratoriach a częściowo w ramach pracy własnej,

- ocena wiedzy i umiejętności zdobytych na laboratoriach poprzez pisemny test złożony z pytań wielokrotnego wyboru, obejmujących łącznie cały zakres laboratorium. Zaliczenie (ocena 3,0) wymaga zdobycia minimum 18 z 35 pkt.

Opcjonalne uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:

- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- uwagi związane z udoskonaleniem materiałów dydaktycznych.

Treści programowe

Program wykładu obejmuje następujące zagadnienia:

Zwinne metody szacowania pracochłonności w procesie wytwarzania oprogramowania.

Zarządzanie wersjami oprogramowania - system Git, wybrane schematy rozwoju oprogramowania (ang. Git workflows).

Automatyzacja budowania aplikacji - system Gradle.

Ciągła integracja (ang. continuous integration) i ciągłe dostarczanie oprogramowania (ang. continuous delivery).

Tworzenie aplikacji w oparciu o mikrousługi (ang. microservices).

Wdrażanie oprogramowania z użyciem kontenerów programowych - system Docker.

Program laboratorium obejmuje następujące zagadnienia:

Rozproszony system wersjonowania Git - przypomnienie podstawowych poleceń, zagadnienia zaawansowane, schemat rozwoju oprogramowania Gitflow.

System Gradle - analiza gotowych przykładów skryptów budujących i ich rozszerzanie, zarządzanie zależnościami, programowanie własnych zadań, tworzenie pluginów.

Ciągła integracja oprogramowania - konfiguracja serwera Jenkins

Tworzenie aplikacji z wykorzystaniem kontenerów programowych w systemie Docker - tworzenie obrazów, uruchamianie kontenerów, narzędzie docker-compose, docker stack, skalowanie aplikacji, orkiestracja.

Metody dydaktyczne

1. wykład: prezentacja multimedialna, demonstracja, dyskusja.

2. ćwiczenia laboratoryjne: słowne wprowadzenie, notatka elektroniczna, prezentacja multimedialna, zadania praktyczne typu krok po kroku (ang. tutorial), otwarte zadania praktyczne.

Literatura

Podstawowa

1. Pro Git, 2nd edition, Scott Chacon, Ben Straub, 2014 (https://git-scm.com/book/en/v2).

2. Agile: metodyki zwinne w planowaniu projektów, Mike Cohn, Helion, 2018 (tytuł oryginału: Agile Estimating and Planning).

3. Gradle User Manual (https://docs.gradle.org/current/userguide/userguide.html).

4. Jenkins User Documentation (https://jenkins.io/doc/).

5. Docker: praktyczne zastosowania, Sean P. Kane, Karl Matthias, Helion, 2017 (tytuł oryginału: Docker: Up & Running).

Uzupełniająca

1. Continuous Integration, M. Fowler, 2006, http://www.martinfowler.com/articles/continuousIntegration.html.

2. Building and Testing with Gradle, T. Berglund, M. McCullough, O'Reilly Media, 2011.

3. Docker Documentation (https://docs.docker.com/).

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	50	2,0
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	32	1,3
Praca własna studenta (przygotowanie do zajęć laboratoryjnych, wykonanie zadania domowego (mini-projektu), przygotowanie do kolokwiów, zapoznanie się ze wskazaną literaturą)[1]	18	0,7

[1] niepotrzebne skreślić lub dopisać inne czynności