Zaawansowana inżynieria oprogramowania

Sylabus zajęć

Informacje podstawowe

Kierunek studiów Informatyka	**Cykl dydaktyczny** 2023/24
Specjalność -	**Kod zajęć** 06INFN.320S.04807.23
Jednostka organizacyjna Wydział Matematyki i Informatyki	**Języki wykładowe** polski
Poziom studiów studia inżynierskie pierwszego stopnia	**Obligatoryjność** Fakultatywny
Forma studiów studia niestacjonarne	**Blok zajęciowy** Przedmioty specjalnościowe
Profil studiów profil ogólnoakademicki	

Koordynator zajęć	Patryk Żywica
Prowadzący zajęcia	Patryk Żywica

Okres	**Forma zajęć / liczba godzin / forma zaliczenia**	**Liczba punktów ECTS**
Semestr 6	• Wykład: 15, Zaliczenie z oceną • Laboratorium: 15, Zaliczenie z oceną	5

Cele kształcenia dla zajęć

Kod	Cel
C1	Zapoznanie studentów z zagadnieniem rzetelności i bezpieczeństwa systemów informatycznych
C2	Zapoznanie studentów z zagadnieniem powtórnego użycia kodu
C3	Omówienie inżynierii oprogramowania opartej na komponentach
C4	Omówienie inżynierii oprogramowania zorientowanej na usługi
C5	Omówienie inżynierii złożonych systemów
C6	Omówienie inżynierii oprogramowania czasu rzeczywistego
C7	Zapoznanie studentów z zagadnieniem zarządzania projektami
C8	Przedstawienie problemu zarządzania jakością i konfiguracją

Wymagania wstępne

Zaliczenie zajęć Inżynieria oprogramowania oraz wiedza z przedmiotu Podstawy programowania.

Efekty uczenia się dla zajęć

Kod	Efekty uczenia się dla zajęć w zakresie	Efekty uczenia się dla kierunku	Metody weryfikacji osiągnięcia efektów uczenia się dla zajęć
Wiedzy – Student/ka:			
W1	rozumie problem rzetelności i bezpieczeństwa systemów informatycznych	INF_K3_W04_inz	Test
W2	rozumie zagadnienie powtórnego użycia kodu i inżynierii opartej na komponentach	INF_K3_W04_inz	Test
W3	rozumie zagadnienie inżynierii oprogramowania rozproszonego i zorientowanego na usługi	INF_K3_W04_inz	Test
W4	rozumie zagadnienie inżynierii złożonych systemów	INF_K3_W04_inz	Test
W5	rozumie zagadnienie inżynierii oprogramowania czasu rzeczywistego	INF_K3_W04_inz	Test
W6	rozumie zagadnienie zarządzania projektami	INF_K3_W04_inz	Test
W7	rozumie problem zarządzania jakością i konfiguracją	INF_K3_W04_inz	Test
Umiejętności – Student/ka:			
U1	potrafi wykorzystać odpowiednie metryki do określania wymagań i mierzalnych kryteriów dostępności, niezawodności, bezpieczeństwa, zabezpieczeń oraz odporności oprogramowania	INF_K3_U04_inz, INF_K3_U06_inz	Projekt, Zadania domowe
U2	potrafi wykorzystać i projektować oprogramowanie/komponenty, pod kątem ich ponownego użycia do budowy nowego systemu informatycznego	INF_K3_U05_inz, INF_K3_U06_inz	Projekt, Zadania domowe

Kod	Efekty uczenia się dla zajęć w zakresie	Efekty uczenia się dla kierunku	Metody weryfikacji osiągnięcia efektów uczenia się dla zajęć
U3	potrafi zastosować odpowiednie wzorce architektoniczne i projektowe do budowy oprogramowania rozproszonego i zorientowanego na usługi	INF_K3_U05_inz, INF_K3_U06_inz	Projekt, Zadania domowe
U4	potrafi zaprojektować oraz zaimplementować złożone usługi sieciowe zgodne z ideą REST	INF_K3_U05_inz	Projekt, Zadania domowe
U5	potrafi opracować koncepcję realizacji złożonego systemu socjotechnicznego	INF_K3_U04_inz, INF_K3_U06_inz	Projekt, Zadania domowe
U6	potrafi zastosować odpowiednie wzorce projektowe do budowy prostego systemu czasu rzeczywistego	INF_K3_U05_inz, INF_K3_U07_inz	Projekt, Zadania domowe
U7	potrafi dokonać analizy ryzyka dla projektu informatycznego	INF_K3_U04_inz, INF_K3_U09	Projekt, Zadania domowe
U8	potrafi przygotować plan realizacji dla złożonego przedsięwzięcia informatycznego	INF_K3_U03, INF_K3_U09	Projekt, Zadania domowe
U9	potrafi wykorzystać narzędzia informatyczne w celu zapewnienia wysokiej jakości oprogramowania	INF_K3_U06_inz	Projekt, Zadania domowe

Treści programowe dla zajęć

Lp.	Treści programowe dla zajęć	Efekty uczenia się dla zajęć	Formy zajęć
1.	• dlaczego niezawodność i bezpieczeństwo są ważnymi atrybutami dla wszystkich systemów oprogramowania; • pięć wymiarów rzetelności: dostępność, niezawodność, bezpieczeństwo, zabezpieczenia i odporność; • pojęcie systemów socjotechnicznych i dlaczego konieczne jest traktowanie systemów jako całość, a nie tylko systemy oprogramowania; • redundancja i różnorodność jako podstawowe pojęcia wykorzystywane w osiąganiu niezawodnych systemów i procesów;	W1, U1	Wykład, Laboratorium

Lp.	Treści programowe dla zajęć	Efekty uczenia się dla zajęć	Formy zajęć
2.	• korzyści i problemy związane z ponownym wykorzystaniem oprogramowania podczas opracowywania nowych systemów; • koncepcja platformy programistycznej oraz sposób jej wykorzystania w rozwoju aplikacji; • pojęcie linii produktów oprogramowania, tworzonej z wykorzystaniem wspólnej architektury i komponentów wielokrotnego użytku, które są konfigurowane dla każdej wersji produktu; • tworzenie i rozwój systemów poprzez konfigurację gotowego oprogramowania • czym jest komponent oprogramowania, który może być zawarty w programie jako element wykonywalny; • kluczowe elementy modeli komponentów i ich wsparcie przez oprogramowanie middleware; • procesy oparte na komponentach (CBSE) do ponownego wykorzystania oraz z ponownym wykorzystaniem; • różne rodzaje kompozycji komponentów i ich wady oraz zalety;	W2, U2	Wykład, Laboratorium
3.	• kluczowe zagadnienia, które należy wziąć pod uwagę przy projektowaniu i wdrażaniu rozproszonych systemów oprogramowania; • model obliczeniowy klient-serwer i warstwową architekturę systemów; • powszechnie używanyme wzorce dla architektur systemów rozproszonych, typy systemów, dla których ma zastosowanie dany wzorzec architektoniczny; • pojęcie oprogramowania jako usługi (SaaS), zapewniającej internetowy dostęp do zdalnie wdrażanych systemów aplikacyjnych; • podstawowe pojęcia usługi sieciowej, standardów usług sieciowych i architektury zorientowanej na usługi; • styl architektoniczny REST l i istotne różnice między usługami typu REST i SOAP; • proces inżynierii usług, którego celem jest tworzenie usług sieciowych wielokrotnego użytku;	W3, U3, U4	Wykład, Laboratorium
4.	• system socjotechniczny i dlaczego kwestie ludzkie, społeczne i organizacyjne wpływają na wymagania i projektowanie systemów informatycznych; • projektowani koncepcyjne i dlaczego jest to niezbędny pierwszy etap w procesie inżynierii systemów; • dlaczego różne procesy zamówień systemowych są stosowane dla różnych typów systemów; • kluczowe procesy rozwoju inżynierii systemów i ich relacjach; • systemy systemów i czym różnią się on od systemu indywidualnego; • systemy klasyfikacji systemów i różnice między różnymi typami systemów systemów; • dlaczego konwencjonalne metody inżynierii oprogramowania oparte na redukcjonizmie są nieadekwatne do tworzenia systemów systemów; • procesy inżynierii systemów systemów i wzorców architektonicznych dla systemów systemów.	W4, U5	Wykład, Laboratorium

Lp.	Treści programowe dla zajęć	Efekty uczenia się dla zajęć	Formy zajęć
5.	• pojęcie oprogramowania wbudowanego, które służy do sterowania systemami reagującymi na zdarzenia zewnętrzne w ich otoczeniu; • procesy projektowania systemów czasu rzeczywistego, w których systemy oprogramowania są zorganizowane jako zbiór współpracujących ze sobą procesów; • wzorce architektoniczne, które są powszechnie stosowane w projektowaniu wbudowanych systemów czasu rzeczywistego; • organizacja systemów operacyjnych czasu rzeczywistego i rola, jaką odgrywają we wbudowanym systemie czasu rzeczywistego;	W5, U6	Wykład, Laboratorium
6.	• podstawowe zadania kierowania projektami informatycznymi; • zarządzanie ryzykami, które mogą wystąpić w projektach informatycznych; • czynniki wpływające na motywację osobistą i ich znaczenie dla kierowania projektami; • kluczowe kwestie wpływające na pracę zespołową, takie jak skład zespołu, organizacja i komunikacja. • podstawy kalkulacji kosztów oprogramowania i czynniki wpływające na cenę systemu oprogramowania; • co powinno się znaleźć w planie projektu tworzonym w ramach procesu rozwoju typu plan-driven; • na czym polega planowanie projektów oraz wykorzystanie wykresów do przedstawienia harmonogramu projektu; • zwinne metody planowania projektów;	W6, U7, U8	Wykład, Laboratorium
7.	• procesy zarządzania jakością i wiedzą, dlaczego planowanie jakości jest ważne; • normy w zarządzaniu jakością, w jaki sposób normy są stosowane w zapewnianiu jakości; • przeglądy i inspekcje wykorzystywane jako mechanizm zapewniania jakości oprogramowania; • zarządzanie jakością w metodach zwinnych opiera się na rozwoju kultury jakości zespołu; • podstawowe funkcje, które powinien zapewniać system kontroli wersji oraz sposób, w jaki jest to realizowane w systemach scentralizowanych i rozproszonych; • wyzwania związane z budowaniem systemu oraz korzyści płynące z ciągłej integracji; • dlaczego zarządzanie zmianami w oprogramowaniu jest ważne oraz podstawowe czynności w procesie zarządzania zmianami; • podstawy zarządzania wersjami oprogramowania;	W7, U9	Wykład, Laboratorium

Informacje dodatkowe

Forma zajęć	Metody i formy prowadzenia zajęć
Wykład	Wykład z prezentacją multimedialną wybranych zagadnień, Wykład problemowy
Laboratorium	Rozwiązywanie zadań (np.: obliczeniowych, artystycznych, praktycznych), Metoda laboratoryjna, Metoda projektu, Praca w grupach

Forma zajęć	Warunki zaliczenia zajęć
Wykład	Warunkiem zaliczenia jest przystąpienie do testów w trakcie zajęć. Warunkiem zaliczenia wykładu jest zaliczenia laboratoriów. Skala ocen: 1. bardzo dobry (bdb; 5,0) – od 90% punktów, 2. dobry plus (db plus; 4,5) – od 80% punktów, 3. dobry (db; 4,0) – od 70% punktów, 4. dostateczny plus (dst plus; 3,5) – od 60% punktów, 5. dostateczny (dst; 3,0) – od 50% punktów, 6. niedostateczny (ndst; 2,0) – poniżej 50% punktów.
Laboratorium	Na ocenę z laboratorium składają się punkty z projektu oraz z zadań domowych w ramach 7 modułów tematycznych według następującego podziału: 1. rzetelności i bezpieczeństwo systemów informatycznych: 10% 2. powtórne użycie kodu: 20% 3. systemy rozproszone i zorientowane na usługi: 20% 4. złożone systemy: 10% 5. oprogramowanie czasu rzeczywistego: 10% 6. zarządzanie projektami: 20% 7. zarządzanie jakością i konfiguracją: 10% Skala ocen: 1. bardzo dobry (bdb; 5,0) – od 90% punktów, 2. dobry plus (db plus; 4,5) – od 80% punktów, 3. dobry (db; 4,0) – od 70% punktów, 4. dostateczny plus (dst plus; 3,5) – od 60% punktów, 5. dostateczny (dst; 3,0) – od 50% punktów, 6. niedostateczny (ndst; 2,0) – poniżej 50% punktów.

Literatura

Obowiązkowa

1. Ian Sommerville. Software Engineering. 10th Edition. Pearson. 2015
2. Ian Sommerville. Inżynieria oprogramowania. Wydawnictwo Naukowe PWN. 2021

Dodatkowa

1. Ian Sommerville. Engineering Software Products: An Introduction to Modern Software Engineering. Pearson. 2019

Nakład pracy studenta i punkty ECTS

Rodzaje zajęć studenta	Średnia liczba godzin* przeznaczonych na zrealizowane rodzaje zajęć
Wykład	15
Laboratorium	15
Czytanie wskazanej literatury	40
Przygotowanie do zajęć	25
Przygotowanie projektu	15
Przygotowanie do zaliczenia	15

Łączny nakład pracy studenta	**Liczba godzin** 125
Liczba punktów ECTS	**ECTS** 5

* godzina (lekcyjna) oznacza 45 minut

Efekty uczenia się dla kierunku

Kod	Treść
INF_K3_U03	Absolwent/ka potrafi pracować indywidualnie i w zespole, zarządzać swoim czasem oraz podejmować zobowiązania i dotrzymywać terminów
INF_K3_U04_inz	Absolwent/ka potrafi opracować, przeanalizować, zaprojektować klasyczne algorytmy i systemy informatyczne
INF_K3_U05_inz	Absolwent/ka potrafi pisać, uruchamiać i testować programy w wybranym środowisku programistycznym
INF_K3_U06_inz	Absolwent/ka potrafi ocenić, na podstawowym poziomie, przydatność metod i narzędzi informatycznych oraz wybrać i zastosować właściwą metodę i narzędzia do typowych zadań informatycznych
INF_K3_U07_inz	Absolwent/ka potrafi zastosować wybrane metody wykorzystywane w wiodących kierunkach badań informatyki
INF_K3_U09	Absolwent/ka potrafi przygotować dokumentacje, opracowania i raporty w języku polskim i języku angielskim, w tym z wykorzystaniem podstawowych ujęć teoretycznych, a także różnych źródeł
INF_K3_W04_inz	Absolwent/ka zna i rozumie zaawansowane pojęcia, konstrukcje i procesy związane z językami programowania i inżynierią programowania