KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Architektura i weryfikacja oprogramowania [S2Inf1E-IO>SAV]

Przedmiot

Kierunek studiów
Informatyka/Computing

Rok/Semestr
1/2

Studia w zakresie (specjalność)
Inżynieria oprogramowania

Profil studiów
ogólnoakademicki

Poziom studiów
drugiego stopnia

Język oferowanego przedmiotu
angielski

Forma studiów
stacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
30

Laboratorium
30

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

5,00

Koordynatorzy

dr hab. inż. Bartosz Walter prof. PP
bartosz.walter@put.poznan.pl

mgr inż. Michał Maćkowiak
michal.mackowiak@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien mieć podstawową wiedzę dotyczącą podstawowych algorytmów i ich złożoności obliczeniowej, programowania obiektowego, wzorców projektowych, baz danych, testowania oprogramowania i aplikacji interetowych. Student powinien potrafić rozwiązywać podstawowe problemy dotyczęce analizy wymagań, tworzenia specyfikacji wymagań, projekowania systemu i umiejętności niezbędnych do zdobywania nowej wiedzy z podanych źródeł informacji. Student powinien rozumieć potrzebę rozszerzenia jej/jego wiedzy i kompetencji oraz mieć chęć pracy w zespole.

Cel przedmiotu

1. Zapoznanie studentów z wiedzą dotyczącą architektury oprogramowania, z uwzględnieniem zrozumienia czym jest architektura, jak powinna być dokumentowana i oceniana 2. Wprowadzenie studentów w temat architektury opartej na komponentach i usługach 3. Rozwinięcie u studentów umiejętności pracy w zespole w kontekście projektowania systemów informatycznych

Przedmiotowe efekty uczenia się

Wiedza:
- ma uporządkowaną i podbudowaną teoretycznie wiedzę dotyczącą architektury oprogamowania, testowania i modelowania oprogramowania
- ma zaawansowaną i szczegółową wiedzę dotyczącą wybranych obszarów informatyki, tworzenia aplikacji interenotwych, tworzenia interfejsów użytkownika, skryptów
- ma zaawansowaną i szczegółową wiedzę dotyczącą cyklu życia oprogramwania, która uwzlędnia rozwój systemów i ich testowanie

Umiejętności:
- potrafi pozyskać, połączyć, zinterpretować i ocenić informacje z literatury, baz danych i innych źródeł informacji w języku ojczystym oraz w języku angielskim; wyciągać wnioski i formułować na ich podstawie konkluzje
- potrafi, przy formułowaniu i rozwiązywaniu zadań inżynierskich, integrować wiedzę z różnych obszarów informatyki (a w razie potrzeby także wiedzę z innych dyscyplin naukowych) oraz zastosować podejście systemowe, uwzględniające także aspekty pozatechniczne
- potrafi ocenić przydatność i możliwość wykorzystania nowych osiągnięć (metod i narzędzi) oraz nowych produktów informatycznych
- potrafi zaprojektować i ocenić architekturę oprogramowania złożonych systemów informatycznych, korzystając z odpowiednich metod
- potrafi, zgodnie z zadaną specyfikacją, uwzględniającą aspekty pozatechniczne, zaprojektować złożony system informatyczny oraz zrealizować ten projekt — co najmniej w części — używając właściwych metod, technik i narzędzi, w tym przystosowując do tego celu istniejące lub opracowując nowe narzędzia
- potrafi współdziałać w zespole, przyjmując w nim różne role, jak architekt czy tester

Kompetencje społeczne:
- rozumie, że wiedza i umiejętności związane z informatyką szybko się zmieniają
- zna przykłady oraz rozumie powody błędów w systemach it prowadzących do porażek i katastrof, które spowodowały straty finansowe, społecznościowe lub doprowadziły do uszczerbku na zdrowiu czy nawet śmierci ludzi

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

1. Ocena formująca:
a) wykłady: na podstawie odpowiedzi na pytania podczas testu obejmującego treści prezetnowane na wykładach;
b) laboratoria: na podstawie oceny z wykonanych zadań podaczas zajęć oraz zadań domowych.
2. Ocena podsumowująca:
a) w ramach wykładów ocena wyznaczona jest na podstawie:
- oceny efektów uczenia przez test pisemny, w ramach którego student może zdobyć maksymalnie 100 punktów. Do zdania niezbędne jest minimum 50 pkt;
- ocena końcowa wyliczana jest z wykorzystaniem następującej skali: (90%, 100%] -> 5.0, (80%, 90%] -> 4.5, (70%, 80%] -> 4.0, (60%, 70%] -> 3.5, (50%, 60%] -> 3.0, (0%, 50%] -> 2.0;
- dyskusji wyników egzaminu;
b) w ramach laboratoriów ocena wyznaczona jest na podstawie:
- oceny przygotowania studenta do poszczególnych laboratoriów oraz jego umiejętności do zrealizowania zadań w ramach laboratoriów;
- oceny pracy wykonanej przez studenta w trakcie laboratoriów - oceniającej umiejętność wykorzystania poznanych zasad i metod;
- oceny wykonania projektu, w tym umiejętności pracy w zespole.

Treści programowe

Definicja architektury oprogramowania. Rola architekta. Proces tworzenia architektury oporgramowania. Style architektoniczne. Dokumentowanie architektury oprogramowania. Ocena architektury oprogramowania i jej powody. Opis metody ATAM (Architecture Tradeoff Analysis Method). Reguły tworzenia dobry diagramów. Architektura oparta na komponentach. Właściwości komponentów. Odwrócenie starowania. Metody wstrzykiwania zależności. Rola kontenera. Przegląd technologii komponentowych. Architektura oparta na usługach. Web services i RESTful jako

implementacje koncepcji architektury SOA. Modelowanie ograniczeń dla modeli UML z wykorzystaniem OCL. Definiowanie warunków wstępnych i końcowych. Walidacja wyrażeń OCL. Design by contract jako metoda quasi-formalna do definiowania funkcjonalności. Przegląd metod testowania na różnych poziomach. Rola i struktura testów w projekcie informatycznym.

Tematyka zajęć

Definicja architektury oprogramowania. Rola architekta. Proces tworzenia architektury oporgramowania. Style architektoniczne. Dokumentowanie architektury oprogramowania. Ocena architektury oprogramowania i jej powody. Opis metody ATAM (Architecture Tradeoff Analysis Method). Reguły tworzenia dobry diagramów. Architektura oparta na komponentach. Właściwości komponentów. Odwrócenie starowania. Metody wstrzykiwania zależności. Rola kontenera. Przegląd technologii komponentowych. Architektura oparta na usługach. Web services i RESTful jako implementacje koncepcji architektury SOA. Modelowanie ograniczeń dla modeli UML z wykorzystaniem OCL. Definiowanie warunków wstępnych i końcowych. Walidacja wyrażeń OCL. Design by contract jako metoda quasi-formalna do definiowania funkcjonalności. Przegląd metod testowania na różnych poziomach. Rola i struktura testów w projekcie informatycznym.

Metody dydaktyczne

Wykład: prezentacja multimedialna, ilustrowana przykładami podawanymi na tablicy.
Ćwiczenia laboratoryjne: prezentacja multimedialna prezentacja ilustrowana przykładami podawanymi na tablicy oraz wykonanie zadań podanych przez prowadzącego.

Literatura

Podstawowa
1. L. Bass, P. Clements, R. Kazman, "Software architecture in practice", WNT
2. P. Kruchten, "The Rational Unified Process-An Introduction", Addison-Wesley
3. R. V. Binder: "Testing Object-Oriented Systems: Models, Patterns and Tools", Addison-Wesley
Uzupełniająca
1. D. Spinellis and G. Gousios, "Beautiful Architecture", O"Reilly Media

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	125	5,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	60	2,50
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	65	2,50