KARTA OPISU MODUŁU KSZTAŁCENIA

Nazwa modułu/przedmiotu	Kod
Zarządzanie wymaganiami i wytwarzaniem oprogramowania	**1010515331010510646**

Kierunek studiów	Profil kształcenia (ogólnoakademicki, praktyczny)	Rok / Semestr
Informatyka	**ogólnoakademicki**	**2 / 3**

Ścieżka obieralności/specjalność	Przedmiot oferowany w języku:	Kurs (obligatoryjny/obieralny)
Technologie wytwarzania oprogramowania	**polski**	**obligatoryjny**

Stopień studiów:	**Forma studiów** (stacjonarna/niestacjonarna)
II stopień	**niestacjonarna**

Godziny	Liczba punktów
Wykłady: **16** Ćwiczenia: **-** Laboratoria: **16** Projekty/seminaria: **-**	**4**

Status przedmiotu w programie studiów (podstawowy, kierunkowy, inny)	(ogólnouczelniany, z innego kierunku)
kierunkowy	**z danego kierunku**

Obszar(y) kształcenia i dziedzina(y) nauki i sztuki	**Podział ECTS (liczba i %)**
nauki techniczne	**4 100%**
nauki techniczne	**4 100%**

Odpowiedzialny za przedmiot / wykładowca:

dr inż. Marcin Szeląg
email: marcin.szelag@cs.put.poznan.pl
tel. 61 665 3023
Instytut Informatyki
ul. Piotrowo 2, 60-965 Poznań

Wymagania wstępne w zakresie wiedzy, umiejętności, kompetencji społecznych:

1	**Wiedza:**	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
2	**Umiejętności:**	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_U2-14, weryfikowane w procesie rekrutacji na studia 2 stopnia - efekty te prezentowane są w serwisie internetowym wydziału www.fc.put.poznan.pl
3	**Kompetencje społeczne**	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 - efekty te prezentowane są w serwisie internetowym wydziału www.fc.put.poznan.pl

Cel przedmiotu:

1. Przekazanie studentom podstawowej wiedzy, w zakresie: zarządzania i inżynierii wymagań, planowania przedsięwzięcia programistycznego, szacowania kosztów oprogramowania, harmonogramowania i monitorowania przedsięwzięć programistycznych, zarządzania ryzykiem, zapewniania jakości i zarządzania przedsięwzięciami realizowanego zgodnie ze zwinnymi metodykami.
2. Rozwijanie u studentów umiejętności w zakresie: planowania, organizacji i zarządzania przedsięwzięciami dotyczącymi wytwarzania oprogramowania oraz wykorzystania do tego celu odpowiednich narzędzi
3. Kształtowanie u studentów umiejętności pracy zespołowej, szczególnie w roli kierownika projektu/zespołu albo analityka

Efekty kształcenia i odniesienie do kierunkowych efektów kształcenia

Wiedza:

http://www.put.poznan.pl/

1. ma pogłębioną wiedzę z zakresu zbierania i analizy wymagań systemów informatycznych - [K2st_W1]
2. ma zaawansowaną i pogłębioną wiedzę z zakresu narzędzi wykorzystywanych do ciągłej integracji, budowania i wersjonowania systemów informatycznych - [K2st_W1]
3. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną z zakresu inżynierii oprogramowania - [K2st_W2]
4. ma podstawową wiedzę dotyczącą zarządzania jakością oprogramowania - [K2st_W2]
5. ma zaawansowaną wiedzę szczegółową dotyczącą szacowania pracochłonności wytwarzania oprogramowania - [K2st_W3]
6. ma wiedzę o trendach rozwojowych w zakresie metodyk wytwarzania oprogramowania - [K2st_W4]
7. ma zaawansowaną i szczegółową wiedzę o procesach zachodzących w różnych modelach cyklu życia oprogramowania - [K2st_W5]
8. zna podstawowe metody, techniki i narzędzia stosowane przy analizie i projektowaniu oprogramowania - [K2st_W6]
9. zna uwarunkowania działalności firm IT - [K2st_W8]

Umiejętności:

1. potrafi pozyskiwać informacje z literatury i Internetu (w języku polskim i angielskim), integrować je, dokonywać ich interpretacji i krytycznej oceny, wyciągać wnioski oraz formułować i wyczerpująco uzasadniać opinie - [K2st_U1]
2. potrafi używać systemów wersjonowania (Git, SVN) do zarządzania konfiguracją oprogramowania - [K2st_U2]
3. potrafi używać systemów i technologii służących do zarządzania zadaniami i zbierania informacji o błędach, np. Trac, JIRA - [K2st_U2]
4. potrafi posługiwać się narzędziami do harmonogramowania projektów - [K2st_U2]
5. potrafi sformułować specyfikację funkcjonalną w formie przypadków użycia - [K2st_U5]
6. potrafi sformułować wymagania pozafunkcjonalne dla wybranych charakterystyk jakościowych - [K2st_U5]
7. potrafi ocenić przydatność i możliwość wykorzystania do wytwarzania oprogramowania nowych metod pracy zespołowej i narzędzi informatycznych - [K2st_U6]
8. potrafi poprawnie użyć do szacowania pracochłonności wytwarzania oprogramowania metodę punktów funkcyjnych lub metodę COCOMO II - [K2st_U7]
9. potrafi efektywnie uczestniczyć w inspekcji oprogramowania - [K2st_U8]
10. potrafi ocenić przydatność metod i narzędzi wykorzystywanych na różnych etapach cyklu życia oprogramowania - [K2st_U9]
11. potrafi przeprowadzić analizę ryzyka związanego z przedsięwzięciem informatycznym - [K2st_U11]
12. potrafi współdziałać w zespole projektowym, przyjmując w nim różne role - [K2st_U15]
13. potrafi określić kierunki dalszego uczenia się i zrealizować proces samokształcenia - [K2st_U16]

Kompetencje społeczne:

1. rozumie konieczność ciągłego doskonalenia procesu wytwarzania oprogramowania, zarówno w aspekcie indywidualnym, jak i zespołowym - [K2st_K1]
2. zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych lub też do utraty zdrowia, a nawet życia - [K2st_K2]
3. rozumie znaczenie działalności propagującej najnowsze techniki i narzędzia wykorzystywane w procesie wytwarzania oprogramowania - [K2st_K3]
4. prawidłowo identyfikuje i rozstrzyga dylematy związane z wytwarzaniem oprogramowania - [K2st_K4]

Sposoby sprawdzenia efektów kształcenia

http://www.put.poznan.pl/

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 egzaminie pisemnym, egzamin składa się z pytań otwartych ocenianych w skali 0-5 pkt. oraz pytań zamkniętych (wielokrotnego wyboru), obejmujących łącznie cały zakres wykładu. Zaliczenie (ocena 3,0) wymaga zdobycia minimum 13 pkt.
- omówienie wyników egzaminu,

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 (lub 2 zadań) dotyczącego przerobionych zagadnień, realizowanego częściowo na laboratoriach a częściowo w ramach pracy własnej,
- ocena wiedzy i umiejętności zdobytych na laboratoriach poprzez pisemny test.

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:

Zarządzanie i inżynieria wymagań: cele i kontekst systemu, wymagania funkcjonalne i pozafunkcjonalne, sposoby pozyskiwania wymagań, opis wymagań, analiza i negocjowanie wymagań, atestowanie wymagań, zarządzanie wymaganiami.

Planowanie przedsięwzięcia: modele cyklu życia oprogramowania, cykl życia projektu, składowe planu przedsięwzięcia, struktura organizacyjna projektu.

Szacowanie kosztów oprogramowania: metody algorytmiczne, metody eksperckie, metody dekompozycji, elementy analizy finansowej projektów.

Harmonogramowanie i monitorowanie przedsięwzięć programistycznych: definiowanie zadań, zależności kolejnościowe, definicja i przydział zasobów, szacowanie czasów trwania zadań, budowa harmonogramu, narzędzia harmonogramowania i zarządzania zadaniami.

Zarządzanie ryzykiem: ryzyko a podejmowanie decyzji, ryzyka w przedsięwzięciach programistycznych, identyfikacja i analiza ryzyka, strategie zarządzania ryzykiem, plan zarządzania ryzykiem.

Zarządzanie konfiguracjami: zasady, modele i narzędzia zarządzania konfiguracjami.

Budowanie i dystrybuowanie aplikacji: Gradle, Docker.

Zapewnianie jakości: kryteria jakości, standaryzacja, przeglądy i audyty, modele CMMI i ISO 900x.

Zarządzanie przedsięwzięciami realizowanymi zgodnie ze zwinnymi metodykami: podstawowe praktyki i cykl życia lekkich metodyk, zalety i ograniczenia.

Program laboratorium obejmuje następujące zagadnienia:

Ogólny przegląd rodzajów narzędzi oraz technologii służących do organizacji projektu informatycznego i ogólnie wspomagania wytwarzania oprogramowania, ze szczególnym uwzględnieniem aspektów praktyki ciągłej integracji (ang. continuous integration - CI). Nowoczesne systemy kontroli wersji (repozytoria kodu źródłowego) i ich rola w projektach informatycznych oraz organizacji pracy programistów. Podstawowe funkcje tych systemów, rodzaje architektury (scentralizowana i rozproszona) i różnice między nimi. Zadania praktyczne z dwóch przykładowych systemów kontroli wersji działających w różnych architekturach (np. technologie Git i Subversion). Automatyzacja cyklu budowania, testowania, raportów jakości, dokumentacji i publikacji oprogramowania - ogólna koncepcja działania, cechy i zalety w kontekście organizacji przedsięwzięć programistycznych. Cechy (wady i zalety) i zastosowanie wybranych narzędzi technologicznych do automatyzacji (np. Ant, Groovy, Gradle, Docker) - zadania praktyczne polegające m.in. na użyciu i tworzeniu skryptów dla powyższych technologii. Serwery continuous integration - ogólna idea i funkcje, przykłady technologii, zastosowanie praktyczne - konfiguracja wybranego serwera CI (np. Jenkins) łącząca różne zagadnienia z poprzednich zadań laboratoryjnych. Zagadnienia inżynierii wymagań - tworzenie i opis wymagań, szablon wymagania, cechy dobrych i złych wymagań, rodzaje wymagań, wymagania funkcjonalne i scenariusze, wymagania pozafuncjonalne. Dobre praktyki inżynierii wymagań według Sommerville'a i Sawyera. Dokument specyfikacji wymagań (SRS) - standard IEEE 830: Recommended Practice for Software Requirements Specifications i jego następca ISO/IEC/IEEE 29148:2011. Systemy i technologie służące do zarządzania zadaniami, zbierania informacji o błędach i tworzenia dokumentacji w projektach informatycznych - funkcje, cechy. Zastosowania powyższych systemów - zadania z wybranej technologii (np. Trac). Integracja powyższych systemów z innymi narzędziami wspomagającymi wytwarzanie oprogramowania - integracja z repozytorium, integracja z IDE na przykładzie wybranej technologii (np. Mylyn + Eclipse). Harmonogramowanie i śledzenie postępów w projekcie

http://www.put.poznan.pl/

informatycznym - podstawowe pojęcia: zasoby, zadania, wykres Gantta, ścieżka krytyczna. Zadania praktyczne z harmonogramowania przy użyciu wybranej technologii (np. Microsoft Project, ProjectLibre).

Cześć wymienionych wyżej treści programowych realizowana jest w ramach pracy własnej studenta.

Metody dydaktyczne:

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, dyskusja.
2. ćwiczenia laboratoryjne: słowne wprowadzenie, notatka elektroniczna, prezentacja multimedialna, zadania praktyczne typu krok po kroku (ang. tutorial), otwarte zadania praktyczne.

Literatura podstawowa:

1. Inżynieria oprogramowania, A. Jaszkiewicz, Helion, Gliwice, 1997.
2. Inżynieria oprogramowania, I. Sommerville, WNT, 2003.
3. Praktyczne podejście do inżynierii oprogramowania, R. S. Pressman, WNT, 2004.
4. Metodyki zarządzania projektami informatycznymi, Z. Szyjewski, PLACET, 2004.
5. Zarządzanie procesem tworzenia systemów informacyjnych, J. Cadle, D. Yeates, WNT, 2004.
6. Zarządzanie projektami informatycznymi, M. Flasiński, PWN, 2013.
7. Inżynieria oprogramowania, K. Sacha, PWN, 2010.
8. Zarządzanie wymaganiami, D. Leffingwell, D. Widrig, WNT, 2003.
9. Agile development. filozofia programowania zwinnego, J. Shore, S. Warden, Helion, 2008.
10. Pro Git, Scott Chacon, Ben Straub, 2nd edition, 2014 (https://git-scm.com/book/en/v2).

Literatura uzupełniająca:

1. Requirements Engineering: A Good Practice Guide, I. Sommerville, P. Sawyer, Wiley, 1997.
2. APM Agile Project Management. Jak tworzyć innowacyjne produkty, J. Highsmith, PWN, 2005.
3. 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering.
4. Continuous Integration, M. Fowler, 2006, http://www.martinfowler.com/articles/continuousIntegration.html.
5. Managing Software Development with Trac and Subversion, D. J. Murphy, Packt Publishing, 2007.
6. Building and Testing with Gradle, T. Berglund, M. McCullough, O'Reilly Media, 2011.

Bilans nakładu pracy przeciętnego studenta

Czynność	Czas (godz.)
1. udział w zajęciach laboratoryjnych / ćwiczeniach	16
2. przygotowanie do ćwiczeń laboratoryjnych	7
3. dokończenie ocenianego zadania lub zadań z ćwiczeń laboratoryjnych	20
4. udział w konsultacjach związanych z realizacją procesu kształcenia	2
5. przygotowanie do testu pisemnego na laboratorium	6
6. udział w wykładach	16
7. zapoznanie się ze wskazaną literaturą / materiałami dydaktycznymi	14
8. przygotowanie do egzaminu	16
9. udział w egzaminie	2
10. omówienie wyników egzaminu	1

Obciążenie pracą studenta

forma aktywności	godzin	ECTS
Łączny nakład pracy	100	4
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	37	2
Zajęcia o charakterze praktycznym	43	2

http://www.put.poznan.pl/