**Nazwa przedmiotu:**

Analiza i projektowanie systemów informacyjnych

**Koordynator przedmiotu:**

Rajmund Kożuszek

**Status przedmiotu:**

Fakultatywny dowolnego wyboru

**Poziom kształcenia:**

Studia II stopnia

**Program:**

Informatyka

**Grupa przedmiotów:**

Przedmioty techniczne - zaawansowane

**Kod przedmiotu:**

APSI

**Semestr nominalny:**

2 / rok ak. 2021/2022

**Liczba punktów ECTS:**

4

**Liczba godzin pracy studenta związanych z osiągnięciem efektów uczenia się:**

1. liczba godzin kontaktowych – 62 godz., w tym
obecność na wykładach 30 godz.,
obecność na spotkaniach projektowych 30 godz.,
obecność na egzaminie 2 godz..
2. praca własna studenta – 56 godz., w tym
wykonanie projektu 40 godz.,
przygotowanie do egzaminu 16 godz.
Łączny nakład pracy studenta wynosi 118 godz., co odpowiada 4 pkt. ECTS.

**Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich:**

2,01 pkt. ECTS, co odpowiada 62 godz. kontaktowym

**Język prowadzenia zajęć:**

polski

**Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym:**

2,37 pkt. ECTS, co odpowiada 40 godz. wykonania projektu plus 30 godz. spotkań projektowych

**Formy zajęć i ich wymiar w semestrze:**

|  |  |
| --- | --- |
| Wykład: | 30h |
| Ćwiczenia: | 0h |
| Laboratorium: | 0h |
| Projekt: | 30h |
| Lekcje komputerowe: | 0h |

**Wymagania wstępne:**

Student powinien:
– mieć wiedzę i umiejętności dotyczące projektowania baz danych,
– znać język UML.

**Limit liczby studentów:**

60

**Cel przedmiotu:**

Zapoznanie studentów z cyklem życia i procesem wytwarzania projektu informatycznego oraz technikami analizy i projektowania jako składowymi tego procesu. Wykształcenie podstawowych umiejętności realizacji projektu przy użyciu metod zwinnych.

**Treści kształcenia:**

Wykład:
– Elementy cyklu życia systemu informatycznego, model kaskadowy i jego problemy.
– Metody rozwiązywania problemów modelu kaskadowego, technika kontroli zmian, model przyrostowy i jego problemy.
– Modele iteracyjne, model spiralny: ich założenia, cechy i mechanizmy; technika Timeboxing. Modele hybrydowe: Rational Unified Process. Podsumowanie i porównanie modeli.
– Problemy tradycyjnego podejścia do realizacji projektu informatycznego.
– Wprowadzenie do zasad podejścia zwinnego (Agile). Manifest i pryncypia podejścia zwinnego. Korzyści i problemy wynikające z jego stosowania.
– Inżynieria wymagań – zbieranie, dokumentowanie i analiza wymagań. Przegląd norm i metod inżynierii wymagań. Wymagania pozafunkcjonalne: ich znaczenie dla projektu, sposób definiowania; przegląd rodzajów wymagań pozafunkcjonalnych. Wymagania ilościowe i jakościowe.
– Technika przypadków użycia (Use Case): sposób opisu, różnicowanie przypadków użycia (biznesowe – systemowe – współpracy, wysokiego poziomu – rozszerzone, główne – drugorzędne, istotne – rzeczywiste, black box – white box). Metody identyfikacji przypadków użycia (4 perspektywy), rola przypadków użycia w procesie realizacji systemu.
– Strukturalizacja modelu przypadków użycia: istota i techniki strukturalizacji, związki pomiędzy aktorami, związki pomiędzy przypadkami użycia, dekompozycja przypadków użycia, najczęstsze błędy. Analiza modelu.
– Technika historyjek użytkownika (User Stories): zasady i cechy techniki, rola historyjek użytkownika w procesie realizacji systemu, historyjki a wymagania, forma opisu, zasady i techniki dekompozycji. Relacja pomiędzy wymaganiami, przypadkami użycia, a historyjkami użytkownika. Stosowalność obu technik w projektach informatycznych.
– Metoda Scrum: wprowadzenie do metody, podstawy pojęcia i techniki: Sprint (Sprint Plannig, Sprint Review, Sprint Retrospective), Backlog – jego tworzenie i pielęgnacja; uczestnicy procesu: zespół, Scrum Master, Product Owner. Zasady stosowania, zalety, wady, i ograniczenia metody, problemy we wdrażaniu, 3 „typy” Scrum.
– Architektura systemu: zakres, cel i zasady definiowania architektury. Sposób opisu i stosowane techniki. Szablony architektoniczne.
– Analiza systemowa: model analityczny, techniki opisu (analiza obiektowa). Relacja model przypadków użycia – model analityczny. Zaawansowane zagadnienia modelowania obiektowego: semantyka diagramu klas, w tym 12 typów związków agregacji, semantyka i techniki opisu modelu stanów.
– Techniki prototypowania i projektowania: Spike, Tracer Bullet, Personas.
– Projektowanie: cel zadania, zawartość i sposób opisu specyfikacji. Relacja model analityczny – model projektowy. Projekt ogólny systemu: przegląd zagadnień koniecznych do ujęcia w specyfikacji.
Projekt:
Zajęcia projektowe polegają na wytworzeniu w zespole (od 4 do 6 osób) niewielkiego, funkcjonującego systemu. W ramach projektu wykonywane są 3 iteracje, w każdej z nich należy zaprezentować przyrost ilościowy funkcjonalności aplikacji względem poprzedniego punktu kontrolnego.
Po zdefiniowaniu problemu do rozwiązania i wyspecyfikowaniu tworzonego produktu na poziomie „biznesowym” zespól realizuje poszczególne iteracje, w ramach których:
– definiuje elementy backlogu oraz zadania do wykonania;
– wykonuje prace analityczne, projektowe i implementacyjne;
– weryfikuje rezultaty.
Każda iteracja podlega weryfikacji przez osobę prowadzącą zajęcia projektowe. Na zakończenie ostatniej iteracji powinien być gotowy działający system wraz z odpowiednią dokumentacją analityczną, projektową, techniczną, użytkową i administratorską.

**Metody oceny:**

Zajęcia projektowe:
– realizacja projektów w zespołach od 4 do 6 osób;
– możliwość cotygodniowych spotkań z prowadzącym na konsultacjach projektowych;
– 4 obowiązkowe spotkania związane z inicjacją projektu i odbiorem 3 iteracji;
– ocena projektu na podstawie demonstracji, wykonanych testów oprogramowania oraz przeglądu dokumentacji.

**Egzamin:**

tak

**Literatura:**

Robert Martin: Agile Software Development, Principles, Patterns, and Practices, Prentice Hall, 2002.
Jeff Sutherland: Scrum: The Art of Doing Twice the Work in Half the Time, Random House Business, 2015.
Kenneth S. Rubin: Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison Wesley, 2012.
Geoff Watts: Scrum Mastery: From Good To Great Servant-Leadership, Inspect & Adapt Ltd, 2013.
Andrew Stellman, Jennifer Greene: Learning Agile: Understanding Scrum, XP, Lean, and Kanban, O'Reilly Media, 2014.
James Coplien, Neil Harrison: Organizational Patterns of Agile Software Development, Prentice Hall, 2004.
Alistair Cockburn: Agile Software Development: The Cooperative Game, Addison-Wesley, 2006.
Alistair Cockburn: Writing Effective Use Cases, Addison-Wesley, 2001 (WNT 2004).
Jim Highsmith: Agile Project Management, Addison-Wesley, 2004.
Mike Cohn: Succeeding With Agile: Software Development Using Scrum, Addison-Wesley, 2010.
Ivar Jacobson, Grady Booch, James Rumbaugh: The Unified Software Development Process, Addison-Wesley, 1999.
Grady Booch, Robert Maksimchuk, Michael Engel, Bobbi Young, Jim Conallen, Kelli Houston: Object-Oriented Analysis And Design With Applications (3rd Edition), Addison-Wesley, 2007.
Kurt Bittner, Ian Spence: Use Case Modeling, Addison-Wesley, 2002.
Karl Wiegers: Software Requirements, Microsoft Press, 2003.
Dean Leffingwell, Don Widrig: Managing Software Requirements: A Use Case Approach, Addison-Wesley, 2003.
Mike Cohn: User Stories Applied: For Agile Software Development, Addison-Wesley, 2004.
Jeff Patton: User Story Mapping: Discover the Whole Story, Build the Right Product, O'Reilly Media, 2014.
Jeffrey Whitten, Lonnie Bentley: Systems Analysis and Design Methods (7th Edition), McGraw Hill, 2007.
Per Kroll, Philippe Krutchen: The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process, Addison-Wesley, 2003 (WNT 2006).

**Witryna www przedmiotu:**

https://usosweb.usos.pw.edu.pl/kontroler.php?\_action=katalog2/przedmioty/pokazPrzedmiot&prz\_kod=103C-INIMU-MSP-APSI

**Uwagi:**

(-)

## Charakterystyki przedmiotowe