

Projektowanie oprogramowania

1. Warunki wstępne

Warunkiem uczestnictwa w zajęciach jest zaliczenie przedmiotu: Podstawy inżynierii oprogramowania (ćwiczenia).

Zajęcia składają się z 30 godzin wykładu i 30 godzin projektu.

2. Cele przedmiotu

C1 Zapoznanie studentów z zadaniami realizowanymi w ramach podstawowych cyklu życia oprogramowania wg ISO/IEC 12207, poznanie przez nich dobrych praktyk projektowych (w tym wzorców projektowych).

C2 WYROBIENIE umiejętności opracowania i przedstawiania w języku technicznym dokumentacji projektowej, praktycznego stosowania języka UML oraz narzędzi wspierających.

3. Przedmiotowe efekty uczenia się

Z zakresu wiedzy:

PEK_W01 Rozpoznaje i opisuje elementy podstawowych diagramów UML stosowanych do modelowania i specyfikacji oprogramowania

PEK_W02 Wymienia, klasyfikuje i opisuje wzorce stosowane na różnych poziomach ziarnistości (architektoniczne, projektowe, implementacyjny – dobre praktyki)

PEK_W03 Wymienia typowe zadania wykonywane w ramach cyklu wytwórczego oprogramowania oraz ich artefakty

Z zakresu umiejętności:

PEK_U01 Stosuje język UML do specyfikacji wymagań i modelowania dziedziny przedmiotowej

PEK_U02 Projektuje graficzny interfejs użytkownika

PEK_U03 Adaptuje podstawowe wzorce architektoniczne lub projektowe do rozwiązywanego problemu inżynierskiego

PEK_U04 Implementuje wybrane wymagania funkcjonalne aplikacji w języku wysokiego poziomu i specyfikuje testy funkcjonalne

Z zakresu kompetencji społecznych:

PEK_K01 Potrafi rozwiązywać napotkane problemy techniczne i dzieli się zdobytą wiedzą w zespole

4. Wykład

4.1. Egzamin

Pisemny. Pytania otwarte i zamknięte (testowe wielokrotnego wyboru, z luką, krótka odpowiedź, dopasowanie etc.). Należy zdobyć co najmniej połowę punktów, aby zdać egzamin.

4.2. Wymagania wstępne do zaliczenia wykładu

Aby podejść do egzaminu student *nie musi zaliczyć* zajęć towarzyszących.

4.3. Zasady dyskwalifikacji

Student otrzymuje ocenę niedostateczną, jeżeli w czasie egzaminu zostanie przyłapany na korzystaniu ze ściąg lub pomocy kolegów.

4. Projekt

Celem projektu jest nabycie umiejętności systematycznej specyfikacji i dokumentacji oprogramowania z wykorzystaniem UML oraz zapoznanie się z narzędziami do modelowania. W ramach projektu ma powstać prototyp oprogramowania wraz z dokumentacją.

Uwaga. Prowadzący projekt mogą ustalić dodatkowe wymagania, niesprzeczne z podanymi poniżej.

4.1. Wymagania wstępne do uzyskania zaliczenia projektu

Obecność na zajęciach projektowych jest obowiązkowa.

4.2. Realizacja projektu

Studenci pracują grupowo (zespoły 2-3 osobowe), ale mogą być oceniani – tam gdzie to możliwe – indywidualnie. Elementy, które mogą być oceniane indywidualnie są zaznaczone na niebiesko. Zajęcia mają charakter konsultacji, w trakcie których zespoły mogą zadawać pytania do częściowych rozwiązań/dokumentów.

Plan prac do wykonania w ramach kolejnych terminów projektów, terminy oddawania wyników prac, punktacja wyników prac, narzędzia realizacji etc. są przedstawione w poniżej tabeli.

4.3. Narzędzia realizacji

Dokumenty tekstowe: edytor tekstowy, arkusz kalkulacyjny (opcja)

Narzędzia do modelowania:

- UML Designer (www.uml-designer.org) – narzędzie open source na różne platformy
- Visual Paradigm Community (wersja darmowa) i/lub Visual Paradigm Online (wersja darmowa): <https://online.visual-paradigm.com/> (możliwość pracy zespołowej)
- <https://app.diagrams.net/>

4.4. Harmonogram czynności

P#	Temat	Uwagi	Rezultaty (punkty)	Narzędzia	Elementy podlegające sprawdzeniu
1	Zajęcia organizacyjne	Przedstawienie tematu wiodącego dla grupy laboratoryjnej. Omówienie sposobu organizacji zajęć.			
P2 – P4 Opracowanie koncepcji systemu (razem: 30 p.); deadline oddania wyników prac P4					
2	Wizja. Słownik.	Studenci budują wizję i słownik dla tematu wiodącego.	Wizja (max. 10p). Słownik (max. 5p).	Edytor tekstu – wizja, zgodnie z szablonem (użytkownicy, cechy, etc.) Edytor/Excel – słownik	Spójność (wewnętrzna, zewnętrzna), niesprzeczność
3	Model domenowy. Reguły biznesowe.	Studenci dla tematu wiodącego na podstawie zdefiniowanego słownika definiują diagram domenowy oraz min. 10 reguł biznesowych (różnych typów) w języku naturalnym. Diagram domenowy zawiera klasy + związki. Atrybuty umieszczamy tylko o ile występują w regułach biznesowych. Reguły biznesowe powinny być podzielone na kategorie. Zaleca się stosowanie szablonów RuleSpeak.	Model domenowy (max. 10p) Reguły biznesowe (różnych typów) (5p.)	Edytor modeli: diagram klas Edytor tekstu lub Excel - reguły biznesowe; w przypadku reguł strukturalnych zaleca się ich „dopięcie” do elementów diagramu klas, np. w postaci notatek	Poprawność składniowa i semantyczna. Poprawność reguł biznesowych. Spójność reguł z modelem domenowym.
P4 – P7 Specyfikacja wymagań (razem 20p. + LP*10p + LP*2.5p* + 5p*); deadline oddania kompletnej dokumentacji P8					
4	Specyfikacja wymagań.	Specyfikacja wymagań w postaci: Diagramu wymagań lub zbioru historyjek	Specyfikacja wymagań (5p.)	Edytor modeli: diagram wymagań lub Word, Excel (dla zbioru historyjek)	Śladowalność do wizji
5	Diagram przypadków użycia.	Na podstawie specyfikacji wymagań studenci definiują diagram przypadków użycia. Do przypadków użycia przypisują realizowane przez przypadek historie użytkownika lub śladują wymagania z diagramu wymagań.	Model PU + opisy streszczające (5p.)	Edytor modeli – diagram przypadków użycia + opisy streszczające (jeżeli zdefiniowano diagram wymagań, śladowanie do wymagań)	Śladowalność przypadków użycia do specyfikacji wymagań
6	Specyfikacja przypadków użycia. Model informacyjny.	Każdy ze studentów w zespole, dla dwóch ustalonych z prowadzącym PU piszą w narzędziu (alternatywnie): 1) specyfikację wejścia/wyjścia dla każdej z historii powiązanych z przypadkiem użycia + powiązane reguły biznesowe; wątki alternatywne na poziomie tytułów 2) scenariusze przypadków (główny i alternatywne) w wersji tekstowej	Specyfikacja PU dla dwóch PU – po 5p. za PU alternatywnie diagramy aktywności (po 7.5p*. za diagram dla PU)	Edytor modeli – specyfikacja PU w postaci scenariuszy lub specyfikacja historyjek w ramach PU (opis we/wy, powiązanych reguł biznesowych) Edytor modeli – diagramy aktywności	Spójność wewnętrzna.

		<p>*) alternatywnie studenci mogą wizualizować specyfikację przypadków użycia za pomocą diagramów aktywności</p> <p>Studenci budują model informacyjny jako uszczegółowienie modelu domenowego. Uszczegółowienie może ograniczać się do rozważanych przypadków użycia, ale model powinien pozostać całościowy.</p> <p>Uwaga: elementy rozważane (przypadki użycia, klasy) odpowiednio pokolorowane.</p> <p>Studenci mogą, opcjonalnie, przetłumaczyć reguły biznesowe do OCL.</p>	<p>Model informacyjny – 10p.</p> <p>Reguły biznesowe w OCL (max. 2.5p.*)</p>	<p>Edytor modeli – diagram klas</p> <p>Narzędzie wspierające OCL (opcja)</p>	
7	Prototyp interfejsu.	<p>Dla ustalonych na P6 przypadków użycia zespół/student buduje prototyp interfejsu (przynajmniej dla wątków głównych); mogą to być rysunki odręczne lub wykonane w dowolnym narzędziu, np. w NetBeans (prototyp ewolucyjny, podejście zalecane). Za zdefiniowanie prototypu dla wątków alternatywnych ze wskazaniem reguł przewodnika stylu – dodatkowe punkty.</p>	<p>Prototyp interfejsu dla wątków głównych po 3p. za PU; prototypy dla wątków alternatywnych dodatkowo po 2p. za PU; za wskazanie reguł przewodnika stylu dodatkowo 2.5p.*</p>	Dowolne dostępne narzędzie	Możliwość realizacji PU z wykorzystaniem zaproponowanego interfejsu.
P8 – P10 Projekt ogólny i szczegółowy (razem 20 + LP*7.5p. + 2.5* + LP*2.5*p); deadline oddania kompletnej dokumentacji P11					
8-9	Architektura systemu. Projekt bazy danych.	<p>Zespół – na podstawie wymagań funkcjonalnych i нефункциональных – proponuje logiczną architekturę oprogramowania oraz architekturę fizyczną systemu. Konieczność wykorzystania wzorca dla architektury logicznej.</p> <p>Na tym etapie powstaje model logiczny</p>	<p>Architektura logiczna - diagram pakietów + opis elementów składowych (max. 5 p.) Architektura fizyczna – diagram rozmieszczenia + opis elementów (max. 5 punktów)</p> <p>Model danych – w</p>	<p>Edytor modeli – diagram pakietów Edytor modeli – diagram rozmieszczenia</p> <p>Edytor modeli – diagram klas (baza</p>	<p>Sprawdzenie pokrycia wymagań funkcjonalnych.</p> <p>Kompletność. Zgodność z</p>

		danych (projekt bazy) poprzez uszczegółowienie modelu informacyjnego.	zależności od technologii, np. diagram klas (baza obiektowa), definicja tabel lub diagram związków encji dla baz relacyjnych (10p.); Można dostać max. 2.5p* za wygenerowanie schematu bazy z modelu, który będzie używany	obiektowa, serializacja), diagram encji lub opis tekstowy tabel (baza relacyjna)	interfejsem. Spójność.
10	Realizacja PU.	Student/zespół, dla wyspecyfikowanych PU, projektuje ich realizację, definiując diagramy klas i sekwencji. Realizacja obejmuje wątki główne i alternatywne. Klasy muszą zostać umieszczone w architekturze. Uwaga: ten etap może być wykonany i oddany po implementacji (redokumentacja)	Realizacja PU – po 10p*. za każdy PU jeżeli wykonane przed implementacją i po 7.5p. – gdy wykonane po implementacji (zalecane); Jeżeli realizacje nie obejmują wątków alternatywnych, w obu przypadkach o 2.5p mniej za PU	Edytor modeli – diagram klas, diagram sekwencji	Zgodność z interfejsem. Spójność.
P11 – P14 Implementacja + testy + ocena jakości (razem LP*15p. + 10p + 12.5p.*); deadline oddania kompletnej dokumentacji P14					
11	Implementacja interfejsu zgodnie z projektem.	Na podstawie prototypu interfejsu student/zespół buduje interfejs. W przypadku wykrycia rozbieżności pomiędzy prototypem i implementacją – student dokumentuje różnice.	Kod źródłowy, który da się skompilować i uruchomić (po 10 p. za PU) + max. 2.5p.* za wygenerowanie fragmentów kodu aplikacji, które będą	Różne, w zależności od języka programowania	Odpowiedź na pytania: - czy działa zgodnie ze specyfikacją (przypadki użycia, prototyp interfejsu) - czy kod źródłowy zgodny z projektem
12	Podłączenie logiki aplikacji do interfejsu.	Implementacja logiki aplikacji (zgodnie z projektem) – wątki główne + alternatywne. Dodatkowe punkty za wykorzystanie narzędzi dokumentacji kodu.			

		Uwaga. Implementacja może być uproszczona, tzn. nie musi działać w środowisku rozproszonym, może korzystać z mechanizmów serializacji do utrwalania danych (strata części punktów).	używane + max. 2.5p.* za wygenerowanie dokumentacji dla kodu źródłowego Za wykorzystanie wzorca projektowego dodatkowo do 2.5p.*		
13	Testy jednostkowe.	Definicja testów jednostkowych dla wybranych metod klas (minimum dla dwóch metod/studenta, testy logiki biznesowej)	Kod z testami (max. 5p.)	Różne, w zależności od języka programowania	Czy testy obejmują sytuacje typowe, brzegowe.
14	Przypadki testowe dla PU. Automatyzacja testów funkcjonalnych. Badanie jakości projektu.	Przypadki testowe (zapisane w postaci tekstowej) dla zrealizowanych PU Automatyzacja testów dla wybranych przypadków użycia Wyliczenie i ocena metryk zaimplementowanej aplikacji	Tekst z przypadkami testowymi (po 5p. za każdy PU) Demonstracja + skrypt testowy (max. 5 p.) Raport z wynikami metryk oraz komentarz (5 p.*)	Word Np. Selenium, Functional Tester, inne Np. JDepend	Odpowiedź na pytania: - czy przypadek testowy zgodny z interfejsem - czy system zachowuje się zgodnie z przypadkami testowymi - czy testy można uruchamiać w dowolnej kolejności Czy dokonano oceny jakości kodu i wyciągnięto wnioski
L15 Rezerwa + oceny					

4.5. Zasady dyskwalifikacji

Zespół (Student) otrzyma ocenę niedostateczną z projektu, jeżeli przedłoży prowadzącemu rozwiązanie, którego nie jest autorem lub będące plagiatem lub kopią całości lub części innego rozwiązania.

4.6. Sposób oceniania

Studenci za wykonane prace otrzymują punkty. Maksymalna liczba punktów za dany artefakt jest podana w tabeli (patrz punkt 4.4). Prowadzący powinien przedstawić ocenę prac najpóźniej dwa tygodnie po oddaniu prac.

Gwiazdką oznaczono punkty za zadania dodatkowe (student/zespół decyduje o tym, czy je wykonać).

Ostateczne terminy oddawania prac są określone w tabeli.

Za oddanie prac po terminie zespół otrzymuje -5p. za każdy tydzień spóźnienia.

Uwaga 1: Aby zaliczyć przedmiot każdy student MUSI zaimplementować przynajmniej 1 przypadek użycia.

Uwaga 2: Oddanej i ocenionej pracy nie można poprawiać w celu podniesienia otrzymanej już punktacji. Terminy oddawania prac są ustalone tak, aby zespół przed oddaniem mógł skonsultować z prowadzącym jakość oddawanego artefaktu.

Uwaga 3: Nie przewiduje się zwolnień z egzaminu za ocenę celującą z projektu.

Jeżeli już oddana i oceniona praca, ze względu na popełnione błędy, może wpłynąć negatywnie na ocenę oddawanego dokumentu, zespół ma możliwość dokonania poprawek w oddanej pracy i przedłożenia obu prac (oddanej-poprawionej i bieżącej).

Za wykonanie projektu student może zdobyć maksymalnie (przy założeniu, że student pracuje samodzielnie, LP = 2): 145p. + 30p.* – razem 175p.

Etap I: 30

Etap II: $20 + 2*5 + 2*5 = 40$ $2*2.5* + 2.5* + 2.5* = 10*$ Razem: 50

Etap III: $20 + 2*7.5 = 35$ $2*2.5* + 2.5* = 7.5*$ Razem: 42.5

Etap IV: $2*15 + 10 = 40$ 12.5* Razem: 52.5

Proponowana skala ocen:

	Dla 1 osoby	
< 50% → ndst	< 72.5p	→ ndst
[50%, 60%) → dst	73 – 87	→ dst
[60%, 70%) → dst+	87.5 – 101.5	→ dst+
[70%, 80) → db	102 – 116	→ db
[80%, 90) → db+	116.5 – 130.5	→ db+
[90%, 97) → bdb	131 – 140	→ bdb
>= 97% → cel	> 140	→ cel

5. Tematy wiodące

Zostaną zaproponowane przez prowadzących projekty.