Wrocław, 17.09.2021

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. Wykład – katalog kursów

2.1. Egzamin

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

2.2. Wymagania wstępne do zaliczenia wykładu

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

2.3. Zasady dyskwalifikacji

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

3. 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.

3.1. Wymagania wstępne do uzyskania zaliczenia projektu

Obecność na zajęciach projektowych jest obligatoryjna.

3.2. 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.

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: 25 p.); deadline oddania wyników prac P4					
2	Wizja. Słownik.	Studenci budują wizję i słownik dla tematu wiodącego.	Wizja (max. 5p.). Słownik (max. 5p).	Edytor tekstu – wizja, zgodnie z szablonem (użytkownicy, cechy, etc.) Visual Paradigm – 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.)	Visual Paradigm – diagram klas Edytor tekstu lub Excel - reguły biznesowe; w przypadku reguł strukturalnych zaleca się ich „dopięcie” do elementów diagramu klas (constrains)	Poprawność składniowa i semantyczna. Poprawność reguł biznesowych. Spójność reguł z modelem domenowym.
P4 – P7 Specyfikacja wymagań (razem 20p. + LP*10p + LP*2.5p* + 7,5p*); deadline oddania kompletnej dokumentacji P8					
4	Specyfikacja wymagań.	Specyfikacja wymagań w postaci: Diagramu wymagań lub zbioru historyjek	Specyfikacja wymagań (5p.)	Visual Paradigm – diagram wymagań lub, alternatywnie Word, Excel (dla zbioru historyjek)	Śladowalność do wizji
	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. Każdy ze studentów ustala z prowadzącym 2 przypadki użycia, którymi będzie się zajmować w ramach dalszych prac.	Model PU + opisy streszczające (5p.)	Visual Paradigm – diagram przypadków użycia + opisy streszczające (jeżeli definiowano diagram wymagań, śladowanie do wymagań)	Śladowalność przypadków użycia do specyfikacji wymagań
5	Model informacyjny.	Studenci budują model informacyjny jako uszczegółowienie modelu domenowego. Uszczegółowienie może ograniczać się do rozważanych przypadków użycia, ale model powinien pozostać całościowy.	Model informacyjny – 10p.	Visual Paradigm – diagram klas	

		Uwaga: elementy rozważane (przypadki użycia, klasy) odpowiednio pokolorowane. Model cyklu życia wybranych obiektów (jeżeli ma zastosowanie) Studenci mogą, opcjonalnie, przetłumaczyć reguły biznesowe do OCL.	Model cyklu życia (max. 2.5*) Reguły biznesowe w OCL (max. 2.5p.*)	Visual Paradigm – diagram stanów Narzędzie wspierające OCL (opcja)	
6	Prototyp interfejsu.	Dla ustalonych na P4 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.	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.*	Dowolne dostępne narzędzie	Możliwość realizacji PU z wykorzystaniem zaproponowanego interfejsu.
7	Specyfikacja przypadków użycia.	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 *) alternatywnie studenci mogą wizualizować specyfikację przypadków użycia za pomocą diagramów aktywności	Specyfikacja PU dla dwóch PU – po 5p. za PU alternatywnie diagramy aktywności (po 7.5p*. za diagram dla PU)	Visual Paradigm – specyfikacja PU w postaci scenariuszy lub specyfikacja historyjek w ramach PU (opis we/wy, powiązanych reguł biznesowych) Visual Paradigm – diagramy aktywności	Spójność wewnętrzna.
		Uwaga. Kolejność wykonania zadań 6 i 7 może być zamieniona.			

P8 – P10 Projekt ogólny i szczegółowy (razem 15 + LP*7.5p. + 2.5* + LP*2.5*p); deadline oddania kompletnej dokumentacji P11					
8-9	Architektura systemu. Projekt bazy danych.	Zespół – na podstawie wymagań funkcjonalnych i niefunkcjonalnych – proponuje logiczną architekturę oprogramowania oraz architekturę fizyczną systemu. Konieczność wykorzystania wzorca dla architektury logicznej. Na tym etapie powstaje model logiczny danych (projekt bazy) poprzez uszczegółowienie modelu informacyjnego.	Architektura logiczna - diagram pakietów + opis elementów składowych (max. 5 p.) Architektura fizyczna – diagram rozmieszczenia + opis elementów (max. 5 punktów) Model danych – w zależności od technologii, np. diagram klas (baza obiektowa), definicja tabel lub diagram związków encji dla baz relacyjnych (5p.); Można dostać max. 2.5p* za wygenerowanie schematu bazy z modelu, który będzie używany	Visual Paradigm – diagram pakietów Visual Paradigm – diagram rozmieszczenia Visual Paradigm – diagram klas (baza obiektowa, serializacja), diagram encji lub opis tekstowy tabel (baza relacyjna)	Sprawdzenie pokrycia wymagań funkcjonalnych. Kompletność. Zgodność z 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	Visual Paradigm – diagram klas, diagram sekwencji	Zgodność z interfejsem. Spójność.

			alternatywnych, w obu przypadkach o 2.5p mniej za PU		
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 używane + max. 2.5p.* za wygenerowanie dokumentacji dla kodu źródłowego Za wykorzystanie wzorca projektowego dodatkowo do 2.5p.*	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).			
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					

3.3. 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.

3.4. Sposób oceniania
Studenci za wykonane prace otrzymują punkty. Maksymalna liczba punktów za dany artefakt jest podana w tabeli (patrz punkt 3.2). 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 student 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): 135p. + 30p.*– razem 165p.

Etap I: 25		
Etap II: 20 + 2*5 + 2*5 = 40	2*2.5* + 2.5*+ 2.5* = 10*	Razem: 50
Etap III: 15 + 2*7.5 = 30	2*2.5* + 2.5* = 7.5*	Razem: 37.5
Etap IV: 2*15 + 10 = 40	12.5*	Razem: 52.5

Proponowana skala ocen:	Dla 1 osoby	
< 50% → ndst	< 67.5p	→ ndst
[50%, 60%) → dst	68 – 81	→ dst
[60%, 70%) → dst+	81.5 – 94.5	→ dst+
[70%, 80) → db	95 – 108	→ db
[80%, 90) → db+	108,5 – 121.5	→ db+
[90%, 98) → bdb	122 – 132	→ bdb
>= 98% → cel	>=132	→ cel

4. Tematy wiodące
Zostaną zaproponowane przez prowadzących projekty.