

# Inżynieria oprogramowania

## Instrukcja do laboratorium

rok akad. 2011/2012

### Informacje podstawowe:

Celem laboratorium jest nabycie przez studentów praktycznej umiejętności wykonywania modeli analitycznych i projektowych systemu oraz posługiwania się narzędziami CASE (Computer Aided Software Engineering). Modele tworzone są w języku UML (Unified Modeling Language) za pomocą narzędzia Enterprise Architect 7.5.

### Prowadzący zajęcia laboratoryjne (Informatyka):

- dr inż. Anna Bobkowska, [annab@eti.pg.gda.pl](mailto:annab@eti.pg.gda.pl), pok. 649, tel. 29-89
- dr inż. Aleksander Jarzębowicz, [olek@eti.pg.gda.pl](mailto:olek@eti.pg.gda.pl), pok. 648, tel. 14-64
- mgr inż. Katarzyna Łukasiewicz, [katarzyna.bulska@eti.pg.gda.pl](mailto:katarzyna.bulska@eti.pg.gda.pl), pok. 648, tel. 14-64
- mgr inż. Alan Turower, [alan.turower@eti.pg.gda.pl](mailto:alan.turower@eti.pg.gda.pl), pok. 318, tel. 13-64

### Prowadzący zajęcia laboratoryjne (Inżynieria Biomedyczna):

- dr inż. Agnieszka Janczulewicz, [agus@biomed.eti.pg.gda.pl](mailto:agus@biomed.eti.pg.gda.pl), pok. 316, tel. 20-09

### Materiały pomocnicze:

Podstawowym źródłem informacji jest treść wykładów przedmiotu oraz udostępniony przykładowy model UML. Dodatkowa literatura z zakresu laboratorium, pomocna w opanowaniu UML i modelowania, obejmuje następujące źródła:

1. Fowler M., Scott K., UML w kropelce (ang. *UML distilled*), LTP Oficyna Wydawnicza, 2005.
2. Booch G., Rumbaugh J., and Jacobson I., UML Przewodnik użytkownika., WNT, 2002.
3. OMG Unified Modeling Language Specification, Version 2.3.
4. Maciaszek L., Requirements Analysis and System Design, Addison Wesley, 2007.
5. McLaughlin B., Pollice G., West D., Head First Object-Oriented Analysis and Design. Edycja polska (Rusz głową!), Helion, 2008.
6. Górski J. (red.), Inżynieria oprogramowania, Wyd. MIKOM, W-wa 1999.

**Harmonogram zajęć w semestrze zimowym 2011/2012:**

	Wstęp	Wizja systemu (5 pkt)		Przypadki użycia (5 pkt + 5 pkt)		Modelowanie obiektowe (5 pkt + 5 pkt)			Modele dynamiki (5 pkt + 5 pkt)		Weryfikacja (5 pkt)		Elementy projektu (5 pkt + 5 pkt)			Rezerwa
			Z	S	Z	S		Z	S	Z		Z	S		Z	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Poniedziałek	26.09	3.10	10.10	17.10	24.10	31.10	7.11	14.11	21.11	28.11	5.12	12.12	19.12	2.01	9.01	16.01
Wtorek	27.09	4.10	11.10	18.10	25.10	1.11	8.11	15.11	22.11	29.11	6.12	13.12	20.12	3.01	10.01	17.01
Środa	28.09	5.10	12.10	19.10	26.10	2.11	9.11	16.11	23.11	30.11	7.12	14.12	21.12	4.01	11.01	
Czwartek	29.09	6.10	13.10	20.10	27.10	3.11	10.11	17.11	24.11	1.12	8.12	15.12	22.12	5.01	12.01	
Piątek	30.09	7.10	14.10	21.10	28.10	4.11	11.11	18.11	25.11	2.12	9.12	16.12	23.12	6.01	13.01	

S - sprawdzian

Z - zaliczenie ćwiczenia

Na czerwono oznaczone są dni wolne.

8 grudnia (czwartek) - zajęcia wg planu piątkowego

17 stycznia (wtorek) - zajęcia wg planu piątkowego

## Zasady zaliczenia:

1. Z przedmiotu „Inżynieria Oprogramowania” wystawiana jest jedna łączna ocena. Do uzyskania jest 50 pkt z egzaminu i 50 pkt z laboratorium, suma wyników studenta z tych dwóch części decyduje o ocenie końcowej.
2. W ramach laboratorium punkty przyznawane są za zadania projektowe oraz sprawdziany.
3. Do zaliczenia laboratorium konieczne jest zaliczenie wszystkich zadań i zdobycie przynajmniej połowy spośród dostępnej puli punktów (25 pkt).
4. Zadania projektowe realizowane są w 3-osobowych zespołach (ewentualnie 2-os.). Treść poszczególnych ćwiczeń opisana jest poniżej. Dokładne zasady realizacji i oddawania zadań są ustalane indywidualnie z prowadzącymi poszczególne terminy laboratorium.
5. Zadania powinny być oddawane terminowo wg zamieszczonego wyżej harmonogramu. Opóźnienie skutkuje punktami ujemnymi (-1 pkt za każdy rozpoczęty tydzień opóźnienia).
6. Zadanie uznaje się za zaliczone, jeżeli zespół uzyskał za nie minimum 50% dostępnej liczby punktów nie licząc opóźnienia (tzn. zadanie oddane 3 tyg. po terminie i ocenione na 4 pkt na 5 możliwych jest uznane za zaliczone mimo wyniku  $4-3=1$ , ale zadanie oddane terminowo i ocenione na 1 pkt wymaga poprawy).
7. Terminy i zakres sprawdzianów są ustalone (patrz harmonogram) i będą dodatkowo omawiane przez prowadzących. Sprawdziany odbywają się tylko raz, nie przewiduje się poprawek. W przypadku nieusprawiedliwionej nieobecności na sprawdzianie punkty te przepadają.
8. Obecność na zajęciach laboratoryjnych jest obowiązkowa. Dopuszczalne są bez konsekwencji 2 nieusprawiedliwione nieobecności. Przy 3 nieobecnościach student otrzymuje -3 pkt do swojego wyniku z laboratorium, przy 4 nieobecnościach -7 pkt. Przy 5 nieobecnościach następuje wykluczenie z laboratorium z oceną niedostateczną.
9. Usprawiedliwienia lekarskie powinny być przedstawiane prowadzącemu na najbliższych zajęciach po powrocie na uczelnię. W przypadku nieobecności na sprawdzianie, usprawiedliwienie uprawnia do pisania sprawdzianu w innym terminie uzgodnionym z prowadzącym.
10. Oceny punktowe za zadania przyznawane są wszystkim członkom zespołu. Jednakże w przypadku zauważenia znaczącej różnicy w pracy poszczególnych osób, prowadzący może te oceny zróżnicować.
11. Próby oddawania cudzych prac i inne przypadki „academic dishonesty” skutkują natychmiastową oceną niedostateczną z laboratorium.
12. Laboratorium jest zamykane wraz z końcem semestru. Po tym czasie żadne zaległe zadania nie będą przyjmowane.

## **Zadania**

W tym punkcie przedstawiono poszczególne zadania projektowe

### Z1 - Wizja systemu

Cele: Wybór tematu, zdefiniowanie zakresu dalszych prac, opracowanie celów, wymagań i ograniczeń rozważanego systemu

Zadania:

1. Ustalenie tematu z prowadzącym zajęcia.
2. Wyznaczenie granicy projektowanego systemu.
3. Przedstawienie prowadzącemu wstępnych założeń dot. wizji systemu.
4. Przygotowanie wzbogaconego wizerunku (rich picture) obszaru problemowego.
4. Wykonanie opisu zadanego systemu według szablonu (plik ws-2011.doc).
5. Zweryfikowanie z prowadzącym przyjętych założeń dotyczących wizji systemu (po sprawdzeniu wyników tego etapu)

Produkty: Dokument wizji systemu (zawierający wzbogacony wizerunek)

### Z2 - Przypadki użycia systemu

Cele: Identyfikacja wymagań funkcjonalnych systemu i przedstawienie ich za pomocą przypadków użycia

Zadania:

1. Identyfikacja i opis aktorów.
2. Identyfikacja przypadków użycia.
3. Opisy przypadków użycia
4. Identyfikacja powiązań pomiędzy aktorami a przypadkami użycia ("communicate") oraz pomiędzy przypadkami użycia ("include", "extend") i przedstawienie tego na diagramie przypadków użycia.

Produkty: Diagram przypadków użycia w Enterprise Architect

### Z3 - Diagram klas

Cele: Stworzenie modelu klas pokrywającego wszystkie istotne z punktu widzenia projektu klasy i powiązania pomiędzy nimi. Po wstępnej identyfikacji klas w systemie poprzez kolejne iteracje należy dążyć do stworzenia modelu pokrywającego pełny zakres funkcjonalny systemu

Zadania:

1. Identyfikacja klas potrzebnych do funkcjonowania systemu.

2. Identyfikacja związków między klasami i określenie ich semantyki (nazwa, liczność).
3. Identyfikacja atrybutów klas oraz w możliwym zakresie usług klas (operacje).
4. Optymalizacja modelu obiektowego z wykorzystaniem dziedziczenia.
5. Opis klas oraz ich atrybutów i operacji (w zakresie niezbędnym do zrozumienia).

Produkty: Diagram klas w Enterprise Architect

#### Z4 - Modelowanie dynamiki systemu

Cele: Identyfikacja i przedstawienie tego, w jaki sposób obiekty klas składowych systemu poprzez współpracę i interakcje pozwalają na zrealizowanie wymaganej funkcjonalności systemu. Identyfikacja i przedstawienie złożonego zachowania obiektów.

Zadania:

1. Identyfikacja scenariuszy.
2. Przygotowanie scenariuszy obejmujących zwykłą działalność.
3. Rozszerzenie scenariuszy o sytuacje 'specjalne', błędne, iteracje, miejsca, gdzie można przerwać działania, wycofać, itp.
4. Zidentyfikowanie zdarzeń (komunikatów) w systemie, a następnie odpowiadających im aktorów, identyfikacja parametrów zdarzeń.
5. Przygotowanie diagramów sekwencji, komunikacji i czynności.
6. Przygotowanie diagramu stanów dla wybranej klasy o skomplikowanym zachowaniu.

Produkty: Diagramy sekwencji (trzy, dla wybranych przypadków użycia), diagram komunikacji (jeden, dla wybranego przypadku użycia), diagram czynności (jeden, do wyboru - dla procesu biznesowego albo dla wybranego przypadku użycia), diagram stanów (dla klasy o najbardziej skomplikowanym zachowaniu)

#### Z5 - Weryfikacja

Cele: Poprawa błędów z poprzednich etapów. Uspójnienie modelu.

Zadania:

1. Poprawa błędów wskazywanych wcześniej przez prowadzącego.
2. Weryfikacja zgodności modeli.
  - a) Zgodność modelu przypadków użycia z wizją systemu.
  - b) Zgodność funkcjonalności zdefiniowanej w modelu przypadków użycia z usługami klas.
  - c) Zgodność pomiędzy diagramami sekwencji, komunikacji i czynności a diagramami klas.

### 3. Weryfikacja czytelności diagramów oraz dokumentacji projektowej.

Produkty: Poprawiony model oraz poprawiona wizja systemu.

#### Z6 - Elementy projektowania

Cele: Przejście od ogólniejszego modelu analitycznego do modelu projektowego ukierunkowanego na implementację w wybranych technologiach

Zadania:

1. Podział systemu (model klas) na podsystemy.
2. Uszczegółowienie klas (typy atrybutów, sygnatury operacji, widzialność).
3. Utworzenie schematu BD dla systemu.
4. Automatyczne wygenerowanie szkieletu kodu.
5. Projekt interfejsu użytkownika.

Produkty: Diagram pakietów obrazujący architekturę, uszczegółowiony diagram klas, pliki ze schematem BD, pliki źródłowe kodu, projekt interfejsu użytkownika (w dowolnej formie)