Modelowanie Systemów informacyjnych (MSI)

Informatyka Społeczna

wersja z 2021-10-03

Wykładowca: **dr inż. Mariusz Trzaska**
(*mtrzaska@pjwstk.edu.pl*, *http://www.mtrzaska.com*)

1. Wprowadzenie

   Przedmiot poświęcony jest wyposażeniu studentów w umiejętność opracowania modelu pojęciowego dziedziny problemowej oraz osadzenia go w konkretnym środowisku implementacyjnym (zarówno obiektowym jak i relacyjnym). Studenci poznają sposoby realizacji konstrukcji, niezbędnych do osadzenia modelu, a nie istniejących w wybranym języku. Dyskutowane są również elementy związane z użytecznością (w tym jej testowaniem) graficznych interfejsów użytkownika. Wiedza teoretyczna poparta jest praktyczną umiejętnością konstruowania diagramów przypadków użycia oraz klas jak również implementacją struktury danych i logiki biznesowej. W trakcie prac, wykorzystywana jest jedna z fundamentalnych zasad współczesnej inżynierii oprogramowania, polegająca na korzystaniu z istniejących komponentów (ang. Reuse). Każdy ze studentów jest zobowiązany do przeprowadzenia fazy analizy oraz wykonania prac projektowych i implementacyjnych, w oparciu o indywidualne wymagania użytkownika (projekt wielkości 8-12 klas).

2. Plan zajęć

Tabela rekomendowanego sposobu realizacji wykładów, ćwiczeń i laboratoriów w rozbiciu na 15 tygodni

Nr tyg.	Wykład	Ćwiczenia
1	• Obiektowość • Model przypadków użycia	• Krótkie przedstawienie założeń przedmiotu oraz projektu • Omówienie zakresu dokumentacji projektowej. • Ćwiczenie różnych pojęć z zakresu obiektowości • Ćwiczenie diagramów *Use Case*
2	• Model obiektowy – cz. 1 • Model obiektowy – cz. 2	Ćwiczenia z diagramu klas.
3	• Model obiektowy – cz. 3 • Diagramy aktywności	Ćwiczenia z diagramu klas i/lub aktywności.

4	Wykorzystanie klas w obiektowych językach programowania:	• **Kolokwium z diagramu klas.** • Ćwiczenia z realizacji klas oraz pojęć pokrewnych w obiektowych językach programowania.
5	• Realizacja asocjacji w obiektowych językach programowania • Realizacja asocjacji w obiektowych językach programowania (2)	• Ćwiczenia z realizacji asocjacji w obiektowych językach programowania. • **Odbiór mini-projektu MP1**
6	• Realizacja różnych modeli dziedziczenia w obiektowych językach programowania • Realizacja pozostałych, wybranych konstrukcji UML w obiektowych językach programowania	• Ćwiczenia z realizacji dziedziczenia w obiektowych językach programowania. • **Odbiór mini-projektu MP2** • Ćwiczenia z realizacji pozostałych, wybranych konstrukcji UML w obiektowych językach programowania.
7	• Wykorzystanie modelu relacyjnego w obiektowych językach programowania • Wykorzystanie modelu relacyjnego w obiektowych językach programowania (2)	• **Odbiór mini-projektu MP3** • Kończenie projektów końcowych. • **Ostateczny termin wysłania dokumentacji projektowej.**
8	• Użyteczność (*usability*) graficznych interfejsów użytkownika • Powtórzenie materiału	• **Odbiór projektów końcowych**. • Wszystkie sprawy związane z zaliczeniem ćwiczeń (w tym kolokwia poprawkowe).

3. Mini-projekty

   Ich celem jest praktyczne sprawdzenie zrozumienia sposobów implementacji poszczególnych konstrukcji z modelu pojęciowego (klasy, asocjacje, ekstensja, itp.) jak również stworzenie fundamentu do projektu końcowego. Każdy z mini-projektów zawiera realizację różnych konstrukcji zgodnie z poniższą tabelą. Powinien posiadać też metodę `main` ilustrującą ich działanie. Terminy oddawania: patrz załączony plan zajęć. Istnieje możliwość oddawania mini projektów na późniejszych zajęciach, ale wiąże się to z 50% redukcją oceny. Elementy MP, które podlegają ocenie[1]:

MP 1 Klasy, atrybuty	**MP 2** asocjacje	**MP 3** dziedziczenie

[1] Należy zaproponować własne przypadki biznesowe, tzn. nie wolno wykorzystywać przykładów z wykładów, książki, itp.

• Ekstensja • Atr. Złożony • Atr. Opcjonalny • Atr. Powt. • Atr. Klasowy • Atr. Pochodny • Met. Klas	• Binarna • Z atrybut • Kwalifikowana • Kompozycja *(w każdym przypadku: liczności 1-* lub *-* oraz automatyczne tworzenie poł. zwrotnego)*	• Klasa abstr. i polimorfizm • Overlapping • Wielodziedziczenie • Wieloaspektowe • Dynamiczne

Mini-projekty są ważną częścią składową oceny końcowej (patrz punkt 5.1.1). W związku z tym warto je dobrze wykonać.

4. Projekt

Projekt składa się z dwóch części: dokumentacyjnej oraz implementacyjnej. Jego tematyka powinna być tak dobrana aby dało się stworzyć odpowiednią liczbę (8 - 12) sensownych klas biznesowych. W związku z tym raczej odpadają wszelkie aplikacje narzędziowe, systemowe, odtwarzacze multimediów, itp. W razie wątpliwości należy skonsultować się z prowadzącym ćwiczenia.

4.1. Dokumentacja

4.1.1. Część analityczną:

4.1.1.1. Cel

4.1.1.2. Zakres

4.1.1.3. Kontekst

4.1.1.4. Wymagania funkcjonalne

4.1.1.5. Wymagania niefunkcjonalne

4.1.1.6. Ewolucja systemu

4.1.1.7. Diagram przypadków użycia

4.1.1.8. Analityczny diagram klas.

4.1.2. Część projektowa

4.1.2.1. Implementacyjny diagram klas. Wszystkie konstrukcje nie istniejące w danym języku programowania są zamienione (zgodnie z podjętymi decyzjami projektowymi) na równoważne.

4.1.2.2. Opis/uzasadnienie podjętych decyzji projektowych (zmian dokonanych pomiędzy diagramami analitycznymi oraz projektowymi).

4.1.3. Dokumentację projektową oddajemy w postaci **jednego** pliku PDF (*MSI_Grupa_Nazwisko_Imię_NrIndeksu.PDF*) wysłanego na adres mailowy prowadzącego ćwiczenia. Ostateczny termin – patrz załączony plan zajęć.

4.2. Implementacja

Szczegóły oceny – patrz pkt. 5.2.1.

4.2.1. Fragment struktury (w wariancie minimalnym, przynajmniej 4 klasy z odpowiednimi powiązaniami; minimum 2 asocjacje oraz 1 dziedziczenie).

4.2.2. Metody potrzebne do realizacji wybranego przypadku (lub przypadków) użycia (np. sprzedaż towaru, wypożyczenie samochodu itp.).

4.2.3. Projekt powinien zawierać przykładowe dane (np. generowane w klasie `main`).

4.2.4. Część implementacyjna projektu będzie indywidualnie odbierana w czasie zajęć (patrz dalej). W związku z tym nie trzeba jej przekazywać w żadnej trwałej formie.

4.2.5. Językiem implementacji może być Java, C# lub C++. Inne języki wymagają zgody prowadzącego ćwiczenia.

4.3. **Każdy** projekt będzie indywidualnie odbierany. W trakcie odbioru można spodziewać się szczegółowych pytań dotyczących sposobu implementacji, np. „co by było gdyby...", „dlaczego jest to zrobione w ten sposób...", „proszę dokonać następującej modyfikacji...". Osoby, które samodzielnie wykonały projekt nie powinny mieć problemów z udzieleniem odpowiedzi na powyższe pytania. Brak umiejętności odpowiedzi na powyższe pytania oznacza **brak zaliczenia ćwiczeń**.

4.4. Ocenie będzie podlegać:

4.4.1. Trudność zadania,

4.4.2. Zakres zrealizowanej funkcjonalności,

4.4.3. Elegancja zaimplementowanych rozwiązań.

5. Zaliczenie ćwiczeń

Ocena końcowa z ćwiczeń składa się z następujących składowych:

5.1. Punkty (maks 100 pkt.; liczy się suma - nie trzeba indywidualnie zaliczać każdego elementu)

5.1.1. Punkty za kolokwium (maks. 50 pkt.)

5.1.2. Punkty za mini-projekty (maks. 15 + 20 + 15 = 50 pkt.),

5.2. Ocena z projektu,

5.2.1. Ocena za implementację.

5.2.2. Ocena za dokumentację

Z każdej części (5.1, 5.2.1, 5.2.2) trzeba otrzymać ocenę pozytywną. W związku z tym osoby, które np. zaliczą MP, a nie zaliczą projektu **nie zaliczą ćwiczeń**.

Dodatkowo można zdobyć punkty za rozwiązania zadań podanych w czasie wykładów. Te bonusowe punkty doliczane są do ogólnej liczby punktów (patrz pkt. 5.1) pod warunkiem posiadania co najmniej 50% pkt.

6. Terminy

Terminy realizacji poszczególnych zadań (mini-projekty, dokumentacja projektu końcowego, implementacja projektu końcowego) są podane w tabeli (pkt. 2). Są

one ostateczne co oznacza, że po ich upłynięciu zadania **nie będą przyjmowane**.

Nie ma również możliwości zaliczania przedmiotu po zakończeniu semestru, warunkowo na egzaminie, w sesji poprawkowej, itp.

7. Egzamin

   Aktualnie egzamin z MSI składa się tylko z części zadaniowej.

   7.1. Należy stworzyć prosty analityczny diagram klas (pojęciowy) na podstawie tekstowych wymagań.

   7.2. Oprócz tego krótko opisać sposób implementacji poszczególnych konstrukcji, uzasadnić kryteria wyboru, podać zalety i wady oraz fragmenty pseudo-kodu (np. w języku Java).

   Istnieje możliwość pisania egzaminu bez części „implementacyjnej" (pkt. 7.2). W takiej sytuacji można otrzymać co najwyżej ocenę *dostateczną plus* (3,5).

   Osoby mające ocenę 5,0 z ćwiczeń są zwolnione z egzaminu.

8. Materiały

   8.1. Książka: M. Trzaska: „Modelowanie i implementacja systemów informatycznych". Rok 2008. Wydawnictwo PJWSTK. Stron 299. ISBN 978-83-89244-71-3.

   - Wersja elektroniczna (eBook działający na Windows, iOS, urządzeniach mobilnych oraz czytnikach): księgarnia internetowa Virtualo.pl.
   - Wersja drukowana: sklep PJWSTK.
   - Fragmenty w PDF: http://www.mtrzaska.com/mas-ksiazka

   8.2. Książki:

   - J. Płodzień, E. Stemposz: Analiza i projektowanie systemów informatycznych, wydawnictwo PJWSTK, 2003 i wydanie II-gie rozszerzone 2005.
   - Ewa Stemposz, Andrzej Jodłowski, Alina Stasiecka: Zarys metodyki wspierającej naukę projektowania systemów informacyjnych. Wydawnictwo PJWSTK, 2013. ISBN 978-83-63103-39-2

   8.3. Informacje ogólne (mogą pojawiać się nowsze wersje).

   http://www.mtrzaska.com/msi-informacje

   8.4. Wersja elektroniczna wykładów:

   https://www.mtrzaska.com/tags/msi/

   Ze względu na skomplikowanie omawianych zagadnień, zaleca się uczęszczanie na wykład (niezależnie od tego, że powyższe materiały są dostępne *on-line*).

   8.5. Bezpłatne książki on-line:

   8.5.1. Bruce Eckel - Thinking in Java: http://www.mindview.net/Books/TIJ/

   8.5.2. Allen B. Downey - How to Think Like a Computer Scientist: Java Version: http://www.greenteapress.com/thinkapjava/

8.5.3. Robert Sedgewick and Kevin Wayne - Introduction to Programming in Java: An Interdisciplinary Approach: http://introcs.cs.princeton.edu/home/

8.6. Przykładowe zadania programistyczne

Niniejsze zadania programistyczne mają na celu jedynie przypomnienie materiału dotyczącego programowania (przyswojonego w czasie wcześniejszych kursów) i ich rozwiązania nie będą oceniane w ramach przedmiotu MSI. Mogą być wykorzystane jako weryfikacja własnych umiejętności. Studenci przystępujący do kursu MSI, powinni umieć rozwiązać część z nich.

https://www.mtrzaska.com/mas-zadania-programistyczne/

9. Narzędzia implementacyjne

Ze względu na fakt iż istnieje dość duża dowolność wyboru technologii realizacji projektu, nie ma listy narzędzi obowiązkowych. Niemniej poniżej umieszczono listę aplikacji, które mogą być przydatne:

- Narzędzia CASE (edytory diagramów):
  - UMLet (*open source*, wieloplatformowy): https://www.umlet.com/
  - Modelio (*open source*, wieloplatformowy): https://www.modelio.org/,
  - Visual Paradigm Community Edition (również jako **on-line**): http://www.visual-paradigm.com,
  - ArgoUML: http://argouml.tigris.org/,
  - MagicDraw Community Edition: http://www.magicdraw.com,
  - StarUML: http://staruml.sourceforge.net/en/,
  - MS Visio (dostępny na licencji ELMS dla studentów PJWSTK),
  - Obszerna lista narzędzi: http://en.wikipedia.org/wiki/List_of_UML_tools.
- IDE
  - IntelliJ IDEA (darmowy *Community*): https://www.jetbrains.com/idea/
  - Eclipse for Java: http://www.eclipse.org/,
  - NetBeans for Java: http://www.netbeans.org/,
  - MS Visual Studio (dostępny na licencji ELMS dla studentów PJWSTK).
- Edytory GUI
  - wbudowany w IntelliJ IDEA lub NetBeans;
  - dla Eclipse: Jigloo SWT/Swing GUI Builder (http://www.cloudgarden.com/jigloo/);
  - dla Eclipse: WindowBuilder Pro - po przejęciu przez Google darmowy (http://code.google.com/intl/pl/webtoolkit/tools/wbpro);
  - wbudowany w MS Visual Studio.

10. Uwagi

W razie wątpliwości proszę o kontakt: *mtrzaska@pjwstk.edu.pl*