PROGRAM ROZWOJOWY
POLITECHNIKI WARSZAWSKIEJ

Zakład Elektrotechniki Teoretycznej i Informatyki Stosowanej
Wydział Elektryczny, Politechnika Warszawska

Laboratorium modelowania oprogramowania w języku UML

Ćwiczenie 8

Modelowanie interakcji, zgodność modelu z kodem

Materiały dla studentów

Projekt współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Spis treści

1. Informacje wstępne

1.1. Cel ćwiczenia

Celem ćwiczenia jest rozwinięcie praktycznych umiejętności w zakresie tworzenia modeli interakcji na poziomie projektu. Nabyte umiejętności powinny umożliwić studentom na samodzielne wykonywanie modeli dynamiki systemu, jako realizacji wymagań funkcjonalnych (przypadków użycia). Powinny również pokazać zależność między tworzonymi modelami a dynamicznym kodem (treścią metod). Tworzone w trakcie ćwiczenia modele interakcji (diagramy sekwencji) mają za zadanie zilustrować zasady realizacji scenariuszy interakcji aktor-system w systemie o architekturze warstwowej. Po zakończeniu ćwiczenia studenci powinni potrafić wykonać podstawowy model dynamiki systemu, opisujący wymianę komunikatów między obiektami. Powinni również potrafić napisać fragmenty kodu zgodne z wytworzonym modelem.

Ćwiczenie wykonywane jest na trzech kolejnych zajęciach, które ilustrują proces realizacji wymagań funkcjonalnych. Zajęcia nr 1 obejmują konstrukcję prostych modeli interakcji bez alternatyw na podstawie podstawowych scenariuszy przypadków użycia utworzonych podczas ćwiczenia 6. Podczas zajęć nr 2 tworzone są modele interakcji zawierające ramki z warunkami (pętla, alternatywa, opcja) na bazie alternatywnych scenariuszy. Zajęcia nr 3 kończą cykl wytworzeniem kodu w oparciu o stworzone wcześniej modele.

Wynikiem ćwiczenia powinien być plik „.EAP” zawierający model interakcji zgodny z wybranymi (najważniejszymi) przypadkami użycia systemu. Dodatkowym wynikiem jest kod, zawierający wybrane treści metod utworzonych na podstawie diagramów interakcji.

Za poprawne wykonanie ćwiczenia można otrzymać 13 punktów. Za wykonanie pierwszych dwóch zadań podczas zajęć można otrzymać po 3 pkt. Za wykonanie ostatniego zadania oraz za poprawiony i rozbudowany model końcowy można otrzymać 7 punktów.

1.2. Tematyka i przygotowanie się do ćwiczenia

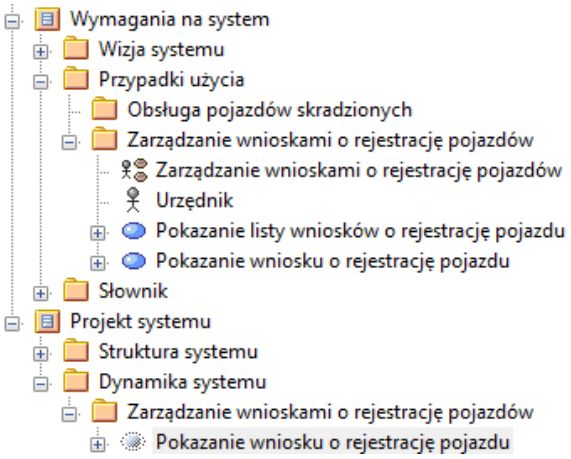
Tematem ćwiczenia jest wykonanie modeli interakcji (diagramy sekwencji). W celu przygotowania się do ćwiczenia należy zapoznać się z problematyką tworzenia diagramów sekwencji na etapach projektowania systemu. Tematyka ta jest opisana w sekcjach 7.3 i 8.3 podręcznika [1]. Bardzo istotne jest zachowanie spójności modelu interakcji z modelami wymagań (słownik, model przypadków użycia), co zostało przedstawione w sekcjach 7.3 i 7.4. Jednocześnie, należy zwrócić uwagę na zgodność tworzonych modeli interakcji z modelem klas na poziomie projektu (ćwiczenie 7, zajęcia 2 i 3).

Oprócz zapoznania się z podstawami modelowania, należy zapoznać się z zasadami działania narzędzia Enterprise Architect (podręcznik użytkownika [2]) w zakresie określonym w omówieniu zadań w instrukcji poniżej.

2. Zajęcia nr 1 – scenariusz pracy

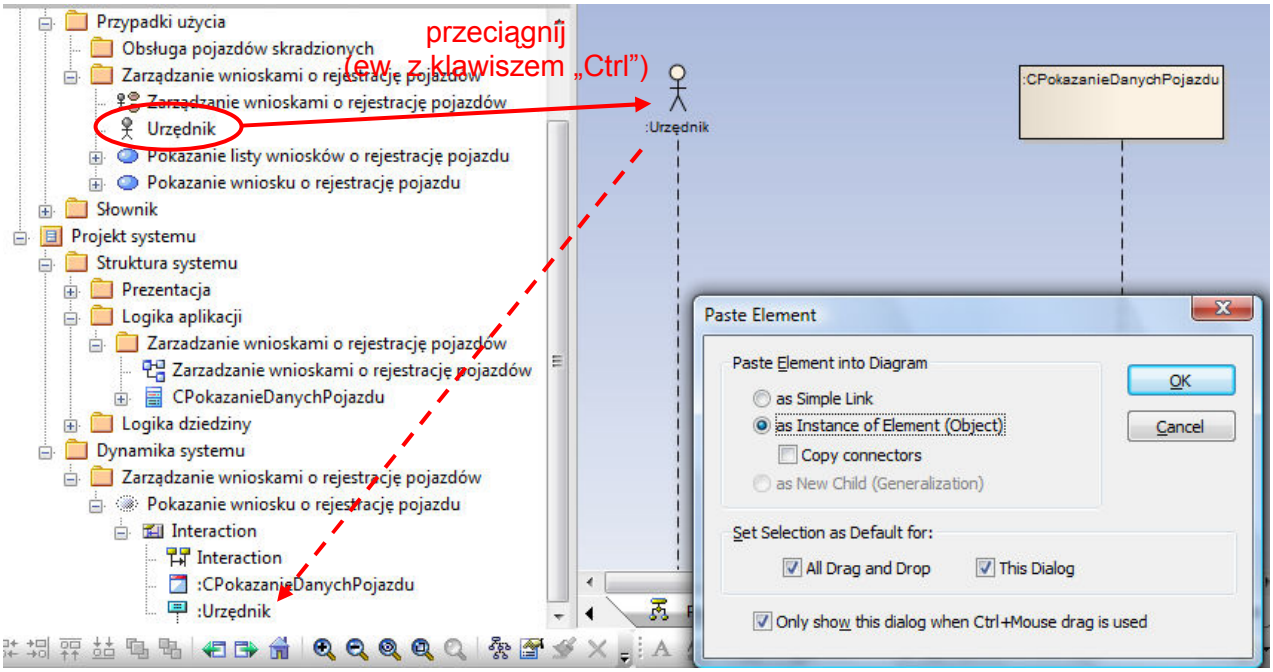
2.1. Omówienie ćwiczenia przez prowadzącego

Zanim utworzymy diagramy sekwencji należy przygotować strukturę modelu dynamiki systemu. W szczególności, należy utworzyć elementy odpowiadające przypadkom użycia, których realizację będziemy modelować. W pakiecie „Dynamika systemu” należy utworzyć pod-pakiety odpowiadające tym zawartym w pakiecie „Przypadki użycia” wymagań na system. Dla wybranych (najważniejszych) przypadków użycia należy następnie w odpowiednim pakiecie utworzyć tzw. kolaboracje (realizacje przypadków użycia). Dokonujemy tego w menu kontekstowym pakietu wybierając opcję „Add→Add element” i dodając elementy typu („Type:”) Collaboration. Ikona tego elementu modelu przypomina ikonę przypadku użycia. Przykładowy efekt naszych działań widać na poniższym rysunku.



Wewnątrz kolaboracji można utworzyć diagram sekwencji wybierając w menu kontekstowym opcję „Add→Interaction→with Sequence Diagram”. Zanim przystąpimy do tworzenia diagramu sekwencji, należy zapoznać się ze scenariuszami odpowiedniego przypadku użycia. Można otworzyć odpowiednie okno dialogowe, lub wygenerowany diagram czynności (patrz ćwiczenie 6, zajęcia 3). Scenariusze przypadku użycia powinny także być łatwo dostępne podczas tworzenia diagramu sekwencji. Diagram sekwencji powinien bowiem stanowić dokładną realizację scenariuszy.

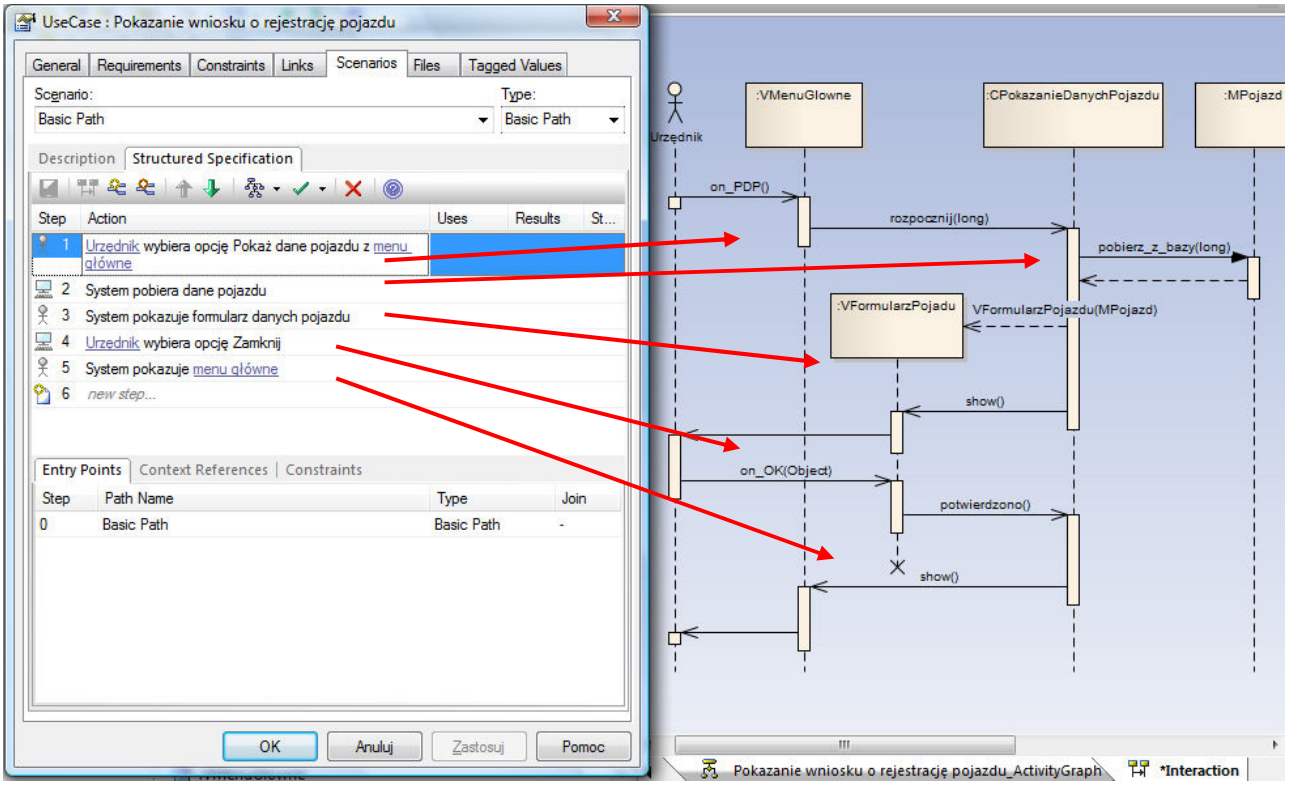
Na diagramie sekwencji ustawiamy linie życia. Najlepszym sposobem jest tutaj przeciągnięcie aktora lub klasy z przeglądarki projektu na diagram. WAŻNE: Element należy wstawić jako instancję („as Instance of Element (Object)”). Łatwo stwierdzić prawidłowość wstawienia, gdyż przed nazwą aktora/klasy na diagramie sekwencji powinien się znaleźć dwukropek „:”. Ponadto, odpowiedni nowy obiekt powinien się pojawić w przeglądarce projektu obok ikony diagramu. Zostało to zilustrowane na rysunku poniżej. UWAGA: tryb wstawiania elementów można zmienić przeciągając element trzymając klawisz „Ctrl” – pojawi się odpowiednie okienko (patrz rysunek).



Klasy umieszczone na diagramie należy przeciągać z pakietów zawartych w pakiecie „Struktura systemu”. Ważne jest przestrzeganie układu diagramu sekwencji. Po lewej stronie umieszczany aktora głównego. Następnie (od lewej): obiekty klas warstwy prezentacji; obiekty klas logiki aplikacji oraz obiekty klas logiki dziedzinowej. Komunikaty między liniami życia tworzymy zgodnie z zasadami poznanymi podczas ćwiczenia 3.

2.2. Zadanie

W trakcie zadania należy zamodelować realizację kilku najważniejszych przypadków użycia. Powinny powstać elementarne diagramy sekwencji, realizujące jedynie podstawowe scenariusze. Należy pamiętać, że podstawą oceny będzie zgodność diagramów sekwencji ze scenariuszami, tak jak to zilustrowano na rysunkach poniżej.



Uwaga: można również sprawdzać zgodność diagramu sekwencji ze scenariuszem opisanym wygenerowanym diagramem czynności (patrz zakładka „Pokazanie wniosku o rejestrację pojazdu_ActivityGraph” na rysunku powyżej).

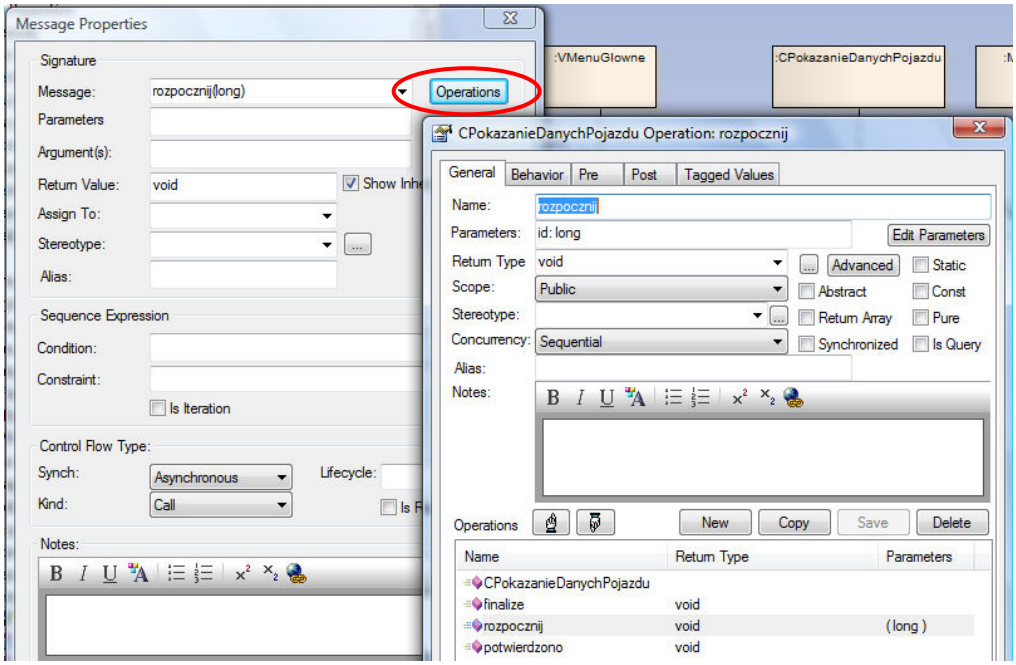
Proponowany plan wykonania zadania:

1. Utworzenie pakietów w pakiecie „Dynamika systemu”.

2. Utworzenie kolaboracji odpowiadających wybranym przypadkom użycia.

3. Utworzenie diagramów sekwencji uwzględniających tylko scenariusze główne przypadków użycia.

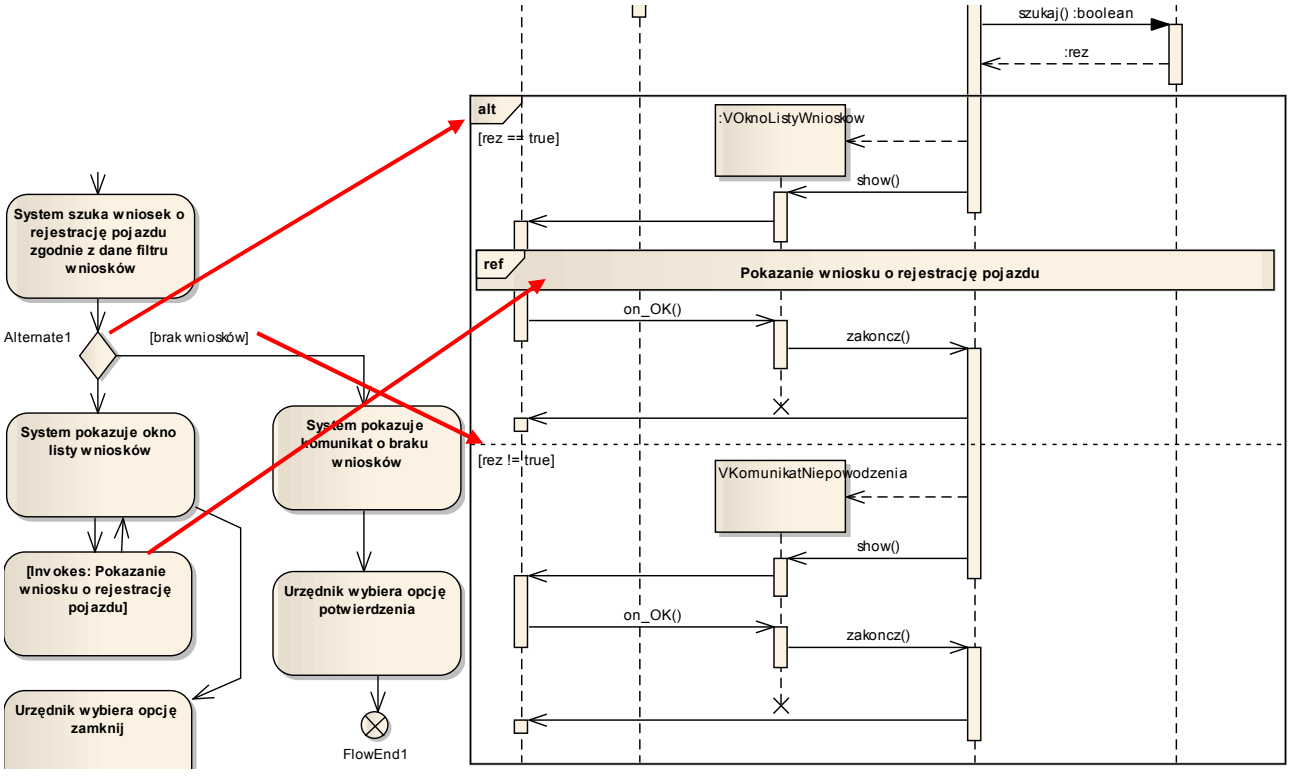
4. Ewentualna modyfikacja (uzupełnienie) klas poprzez zmianę lub uzupełnienie sygnatur operacji, w zależności od potrzeb. UWAGA: nazwy komunikatów **muszą** odpowiadać operacjom klas obiektów, do których kierowane są komunikaty. Jeśli klasa nie posiada odpowiedniej operacji lub jej sygnatura (parametry) jest niepełna, należy dokonać modyfikacji przyciskając przycisk „Operations” w okienku komunikatu (patrz rysunek poniżej).



3. Zajęcia nr 2 – scenariusz pracy

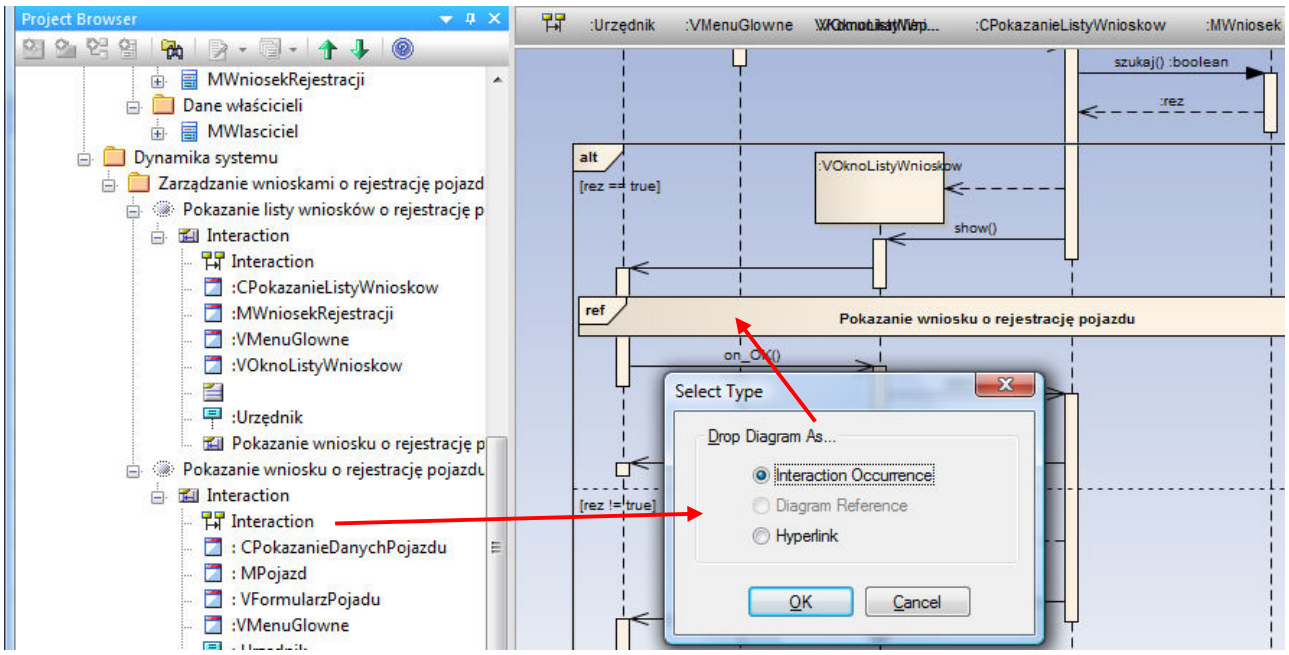
3.1. Omówienie zadania

Zadanie wykonywane podczas drugich zajęć jest kontynuacją zadania z zajęć pierwszych. Dodatkowym, istotnym elementem są ramki umożliwiające modelowanie alternatywnych przebiegów interakcji. Będziemy stosować trzy rodzaje ramek: „opt", „alt" oraz „ref". Pierwsze dwa rodzaje stosujemy dla scenariuszy alternatywnych. Drugi rodzaj stosujemy dla węzłów «invoke», które zamieniamy w ramki odpowiadające diagramowi sekwencji dla realizacji wstawianego przypadku użycia. Odpowiednie zależności między zdaniami scenariusza przypadku użycia, a elementami diagramu sekwencji pokazano na rysunku poniżej.



Podczas tworzenia diagramów warto pamiętać, że ramki „alt” i „opt” tworzymy wybierając odpowiedni element z przybornika diagramu. W ramach ramki „alt” (we właściwościach) należy utworzyć tyle sekcji, ile mamy scenariuszy alternatywnych (np. sekcje „rez == true” i „rez != true” na rysunku powyżej).

Ramki „ref” tworzymy przeciągając odpowiedni diagram sekwencji z przeglądarki projektu do wnętrza diagramu. Po przeciągnięciu należy wybrać wariant „Interaction Occurence”. Powyższą zasadę ilustruje poniższy rysunek.



3.2. Zadanie

W ramach wykonania zadania należy rozbudować realizacje przypadków użycia. W szczególności, należy uzupełnić diagramy sekwencji z poprzednich zajęć o ramki „alt”, „opt” oraz „ref”. Należy utworzyć ramki każdego z powyższych rodzajów. Oznacza to, że do realizacji należy wybrać przykładowe przypadki użycia połączone relacją «invoke». Należy również utworzyć kolejne realizacje przypadków użycia (kolaboracje), dla najważniejszych przypadków użycia nie uwzględnionych podczas poprzednich zajęć.

Proponowany plan wykonania zadania:

1. Uzupełnienie istniejących realizacji przypadków użycia o ramki uwzględniające scenariusze alternatywne.

2. Utworzenie kolaboracji (z diagramami sekwencji) odpowiadających wybranym przypadkom użycia nie uwzględnionym podczas poprzednich zajęć.

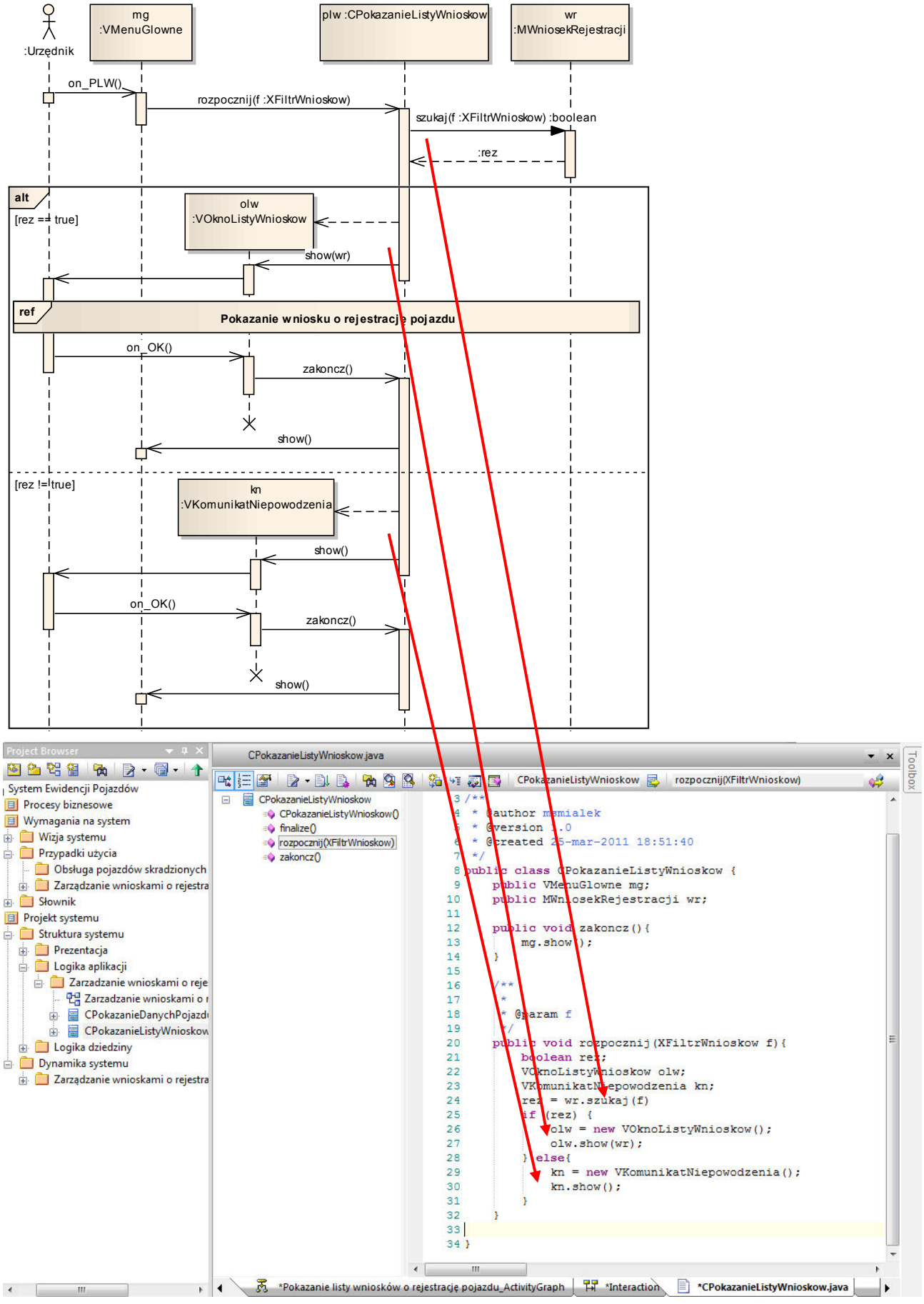
4. Ewentualna modyfikacja (uzupełnienie) klas poprzez zmianę lub uzupełnienie sygnatur operacji, w zależności od potrzeb.

4. Zajęcia nr 3 – scenariusz pracy

4.1. Omówienie zadania

Zadanie jest kontynuacją poprzednich zajęć polegającą na stworzeniu kodu odpowiadającego diagramom sekwencji. Odpowiednie zależności zilustrowane są na rysunku na następnej stronie. Zaznaczono niektóre zależności między elementami diagramu i kodem klasy warstwy logiki aplikacji. Zachowana jest również zgodność nazw komunikatów i metod, obiektów i zmiennych/atrybutów

oraz nazw parametrów. Przed przystąpieniem do wykonania zadania należy uważnie przestudiować pokazane zależności.



4.2. Zadanie

W ramach zadania należy wykonać kod metod klas warstwy logiki aplikacji (klas o nazwach z przedrostkiem „C”. Należy uzupełnić ciała metod, tak, aby zachować zgodność z realizacjami przypadków użycia. Należy zwrócić szczególną uwagę na zgodność elementów napisanych „ręcznie” z elementami wygenerowanymi automatycznie. W szczególności, dotyczy to nazw i parametrów metod oraz ich wywołań, a także nazw atrybutów (pól) klas, używanych w treści metod. W razie stwierdzenia niezgodności, należy poprawić model i/lub treść metod, a następnie wygenerować kod ponownie.

Proponowany plan wykonania zadania:

1. Uzupełnienie treści metod wybranych klas warstwy logiki aplikacji.

2. Ewentualne poprawnie i/lub uzupełnienie modelu (klasy, diagramy sekwencji) w razie stwierdzenia nieprawidłowości.

3. Ewentualna ponowna generacja kodu poprawionych klas.

5. Literatura

1. Michał Smiałek: *Zrozumieć UML 2.0. Metody modelowania obiektowego*, Wydawnictwo Helion, 2005
2. Enterprise Architect User Guide (http://www.sparxsystems.com/bin/EAUserGuide.pdf)