KATEDRA INFORMATYKI STOSOWANEJ PŁ

ANALIZA I PROJEKTOWANIE SYSTEMÓW INFORMATYCZNYCH

Analiza Przypadków Użycia - analiza zachowania

Przygotował: mgr inż. Radosław Adamus

Wprowadzenie:

W procesie definiowania wymagań dla systemu tworzyliśmy *Model Przypadków Użycia*. Model ten zawiera opis wymagań funkcjonalnych dla projektowanego systemu. Kolejnym zadaniem, jakie musi zostać podjęte jest analiza przypadków użycia. W tym procesie tworzony jest opis sposobu, w jaki konkretny przypadek użycia jest realizowany w *modelu projektowym*. Można powiedzieć, że tworzy się dzięki temu pomost pomiędzy modelem przypadków użycia a modelem projektowym.

Każdy przypadek użycia zdefiniowany w perspektywie przypadków użycia (ang. *use case view*) ma swój odpowiednik w modelu projektowym znajdującym się w perspektywie logicznej (ang. *Logical View*) w postaci *realizacji przypadku użycia*.

Analiza przypadków użycia

Określanie kluczowych abstrakcji dla systemu

Proces pozyskiwania wymagań i modelowanie biznesowe powoduje odkrycie podstawowych pojęć występujących w modelowanej rzeczywistości, które muszą zostać odzwierciedlone w projektowanym systemie. Pojęcie te nazywane SA *kluczowymi abstrakcjami*. Są one związane z systemem jako całością i dzięki wykryciu ich na samym początku procesu analizy przypadków użycia, unikamy powtórzeń w procesie analizy pojedynczych przypadków użycia. Dodatkowo możemy spodziewać się, że model, który powstanie będzie bardziej spójny (zmniejszamy prawdopodobieństwo sytuacji, że dla różnych przypadków użycia zostaną wybrane różne nazwy dla tych samych pojęć).

Źródłem kluczowych abstrakcji są:

- Wiedza dziedzinowa,
- Wymagania,
- Słownik pojęć,

Znajdowanie klas analizy

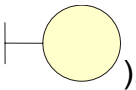
Dla stworzenia wczesnego, koncepcyjnego modelu elementów systemu, posiadających zachowanie i odpowiedzialności wykorzystuje się tzw. *klasy analizy*. Stanowią one pierwszy szkic modelu obiektowego naszego systemu.

Zadaniem *klas analizy* jest opis najważniejszych wymagań funkcjonalnych dla systemu. Klasy analizy modelują obiekty z dziedziny problemowej. Umożliwiają nam tworzenie modelu na wysokim poziomie abstrakcji bez podejmowania decyzji, które z elementów systemu będą miały realizację programową, a które sprzętową.

Można wyróżnić trzy podstawowe aspekty systemu, które mogą podlegać częstym zmianom:

- Granice pomiędzy systemem a aktorem,
- Informacje, których używa system,
- Logika sterowania systemem.

Aby odizolować te, podatne na zmiany, części systemu wyróżniamy trzy typy klas analizy:

- Klasy ***boundary*** (stereotyp: <<boundary>>, ikona: ),
- Klasy ***entity*** (stereotyp: <<entity>>, ikona: ),
- Klasy ***control*** (stereotyp: <<control>>, ikona: ).

Dzięki temu podziałowi, możemy zdefiniować role, jakie grają obiekty w wykonywaniu określonego przypadku użycia.

Klasa boundary:

Klasa boundary stanowi pośrednią warstwę pomiędzy interfejsem a czymś na zewnątrz systemu. Izoluje system od zmian w otoczeniu (np. zmiany w interfejsie do innego systemu, zmianach w wymaganiach użytkownika dotyczących interfejsu z systemem).

System może posiadać kilka typów klas *boundary*:

- Klasy interfejsu użytkownika (komunikacja z użytkownikiem systemu)
- Klasy interfejsu z innym systemem (komunikacji z innym systemem)

- Klasy interfejsu sprzętowego (komunikacja z urządzeniami sprzętowymi, czujnikami, itp.)

Wskazówki dotyczące klas *boundary*:

- Jedna klasa *boundary* dla pary aktor/przypadek użycia (w dalszym procesie analizy może zostać to zmienione poprzez uszczegółowienie np. dla systemu z interfejsem graficznym jedna klasa *boundary* może odpowiadać jednemu oknu czy formularzowi w postaci dialogu).
- Aktor komunikuje się tylko z klasą *boundary*.
- Zidentyfikuj klasy *boundary* tylko w oparciu o zjawiska zachodzące w ciągu zdarzeń dla przypadku użycia.
- Rozważ zewnętrzne źródła zdarzeń i upewnij się, że system może wykryć zajście takiego zdarzenia.
- Dla klas interfejsu użytkownika skoncentruj się na tym, jakie informacje są prezentowane użytkownikowi, nie koncentruj się na szczegółach interfejsu użytkownika (to nie jest szczegółowy projekt interfejsu użytkownika).
- Dla klas interfejsu systemowego i sprzętowego skoncentruj się na tym jaki protokół komunikacji musi zostać zdefiniowany, nie koncentruj się na tym jak protokół zostanie zaimplementowany.

Klasa entity:

Klasy *entity* reprezentują kluczowe pojęcia związane z systemem. Stanowią one punkt widzenia na system pokazujący logiczną strukturę danych. Klasa *entity* reprezentuje i zarządza informacjami (danymi) w systemie. Zazwyczaj obiekty klas *entity* są obiektami trwałymi.

Klasy *entity* najczęściej:

- nie są specyficzne dla jednego przypadku użycia,
- są niezależne od środowiska zewnętrznego (aktorów)
- mogą posiadać zachowanie jednak najczęściej jest ono silnie związane z samym zjawiskiem reprezentowanym przez obiekt klasy *entity*.

Źródłami dla wyszukiwania klas *entity* są: słownik pojęć, ciągi zdarzeń dla przypadku użycia, kluczowe abstrakcje.

Wskazówki dla wyszukiwania klas *entity*:

- wyszukiwanie rzeczowników i fraz rzeczownikowych w opisie ciągów zdarzeń w *specyfikacji przypadku użycia*.
  - Stworzenie listy znalezionych wyrażeń
  - Usuwanie wyrażeń nadmiarowych i zbyt ogólnikowych
  - Usuwanie aktorów
  - Usuwanie atrybutów klas (powinny być zachowane dla dalszego procesu)
  - Usuwanie operacji

Klasa control

Klasa *control* stanowi abstrakcję koordynatora przypadku użycia, która hermetyzuje zachowanie specyficzne dla konkretnego przypadku użycia. Obiekty klasy *control* kontrolują inne obiekty, koordynując działania wykonywane w realizacji przypadku użycia.

Obiekt klasy *control* jest tworzony w momencie rozpoczęcia realizacji przez system przypadku użycia i zazwyczaj kończy swoje życie wraz z zakończenie działania odpowiedniego przypadku użycia.

Zazwyczaj klasa *control*, ze względu na swoja funkcję jest specyficzna dla przypadku użycia (każdy przypadek użycia ma swoją klasę *control*), jednak niektóre obiekty klas *control* mogą współdziałać przy realizacji więcej niż jednego przypadku użycia (jeżeli przypadki te są ściśle ze sobą powiązane).

Wskazówki dla wyszukiwania klas *control*:

- Jedna klasa *control* na przypadek użycia.

Nie wszystkie przypadki użycia potrzebują klasy *control*. Jeżeli przypadek użycia posiada tylko jedną klasę *entity*, to przypadek użycia może być realizowany w postaci kooperacji obiektu klasy *entity* i obiektu klasy *boundary*.

Podsumowanie klas analizy:

Dla każdego przypadku użycia istnieje przynajmniej jeden diagram klas o nazwie VOPC (ang. *View of Participating Classes*) przedstawiający klasy (analizy) współdziałające przy realizacji tego przypadku użycia oraz związki pomiędzy nimi. Dlatego po wstępnym określeniu klas analizy należy stworzyć taki diagram dla każdego przypadku użycia. Diagramy zostaną uzupełnione w kolejnych krokach procesu analizy przypadków użycia. Uzupełnienie będzie dotyczyło doprecyzowania

samych klas analizy, określenia operacji i atrybutów tych klas oraz zdefiniowania asocjacji pomiędzy klasami.

Należy pamiętać o tym, że podział na klasy analizy zniknie w procesie konkretyzacji projektu systemu. Można powiedzieć, że są swego rodzaju prototypami klas, część z nich stanie się podsystemami, komponentami, klasami, inne zostaną odrzucone w dalszym procesie projektowym.

Opis przypadku użycia w terminach interakcji obiektów (przypisanie zachowania do klas (analizy)) – diagramy interakcji

Celem tej aktywności jest rozdzielenie odpowiedzialności za działania wykonywane w realizacji przypadku użycia pomiędzy klasy analizy (oraz ewentualne ich uzupełnienie). Zadanie to jest wykonywane poprzez opisanie działania wykonywanego podczas realizacji przypadku użycia w terminach współpracujących obiektów klas analizy.

Do modelowania współdziałania pomiędzy obiektami służą w języku UML *diagramy interakcji*. Wyróżniamy dwa diagramy interakcji:

- Diagramy współpracy (ang. *Collaboration Diagrams*)
- Diagramy sekwencji (ang. *Sequence Diagrams*)

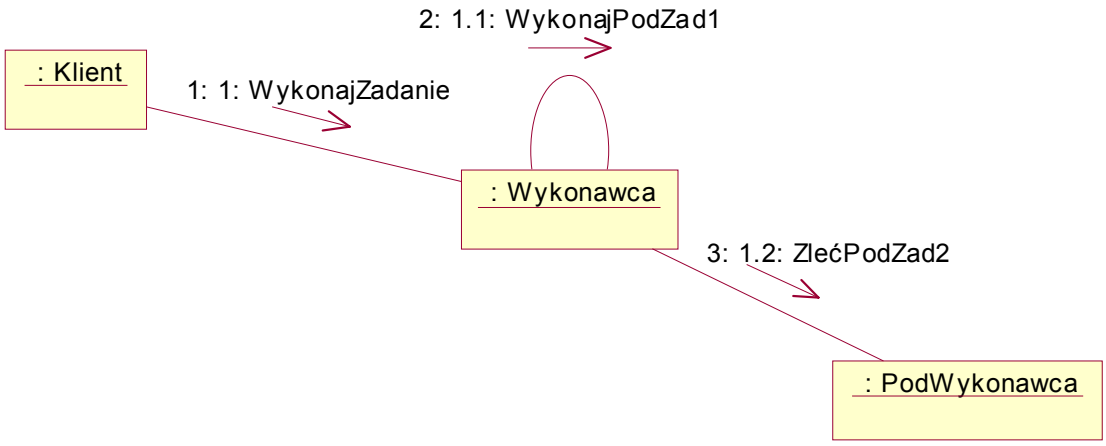
Proces rozdzielania odpowiedzialności pomiędzy klasy analizy wymaga przypisania odpowiedzialności do klas a następnie zamodelowania procesu realizacji przypadku użycia na diagramach interakcji.

Diagramy interakcji

Diagramy interakcji stanowią jeden z rodzajów diagramów dynamicznych. Pozwalają na utworzenie opisu interakcji obiektów systemu podczas realizacji danego zadania: przypadku użycia (czy jednego konkretnego ciągu zdarzeń dla danego przypadku użycia).

Diagramy współpracy pokazują w jaki sposób system realizuje dany przypadek użycia. Współpracujące obiekty, połączone “linkami”, stanowią rodzaj “kolektywu”, zwanego tu kolaboracją. Linki odpowiadają powiązaniom, czyli wystąpieniom asocjacji z diagramu klas, a to oznacza, że odpowiednia asocjacja musi istnieć na diagramie klas (ściślej mówiąc „link” na diagramie współpracy stanowi podstawę do zdefiniowania asocjacji na diagramie klas).

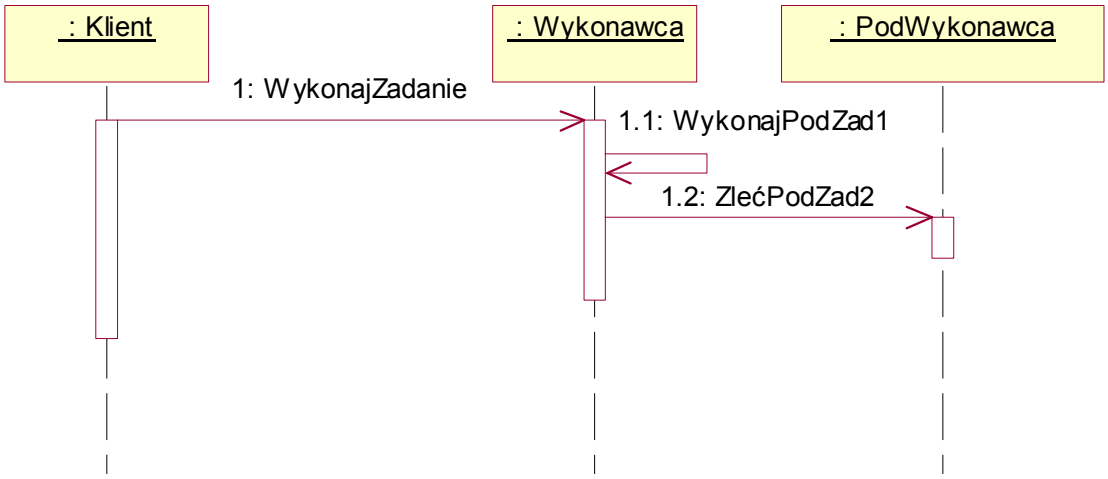
Diagramy współpracy mogą dodatkowo pokazywać interakcje zachodzące między obiektami zaangażowanymi w realizację danego przypadku użycia. Sekwencja interakcji oznacza tu sekwencję komunikatów przesyłanych między współpracującymi obiektami.



Rys. 1 Diagram współpracy

Na diagramach współpracy nie pokazuje się odpowiedzi na wysyłane komunikaty. Komunikaty mogą być numerowane, albo kolejnymi liczbami naturalnymi, albo stosując tzw. numerację zagnieżdżoną (patrz Rys. 1).

Diagramy sekwencji pokazują informację semantycznie identyczną jak diagramy współpracy, natomiast kładą przede wszystkim nacisk na kolejność komunikatów (w przeciwieństwie do diagramów współpracy, które większą wagę przywiązują do powiązań pomiędzy obiektami).



Rys 2. Diagram sekwencji

Wybór diagramu interakcji

Wybór rodzaju diagramu interakcji używanego do modelowania zachowania systemu przy realizacji przypadku użycia jest przede wszystkim uzależnione od osobistych preferencji. Wynika to z tego, że diagramy te (sekwencji i współpracy) pokazują semantycznie równoważną informację. Jednak istnieją pewne przypadki, dla których jeden z diagramów jest bardziej adekwatny niż drugi.

- Dla sesji burzy mózgów lepszym diagramem może być diagram współpracy uwypuklający zależności pomiędzy obiektami (a co za tym idzie klasami).
- Diagram współpracy ułatwia wyszukanie wzorca komunikacji pomiędzy klasami, który może potem stać się podstawą warunkującą użycie wzorca projektowego modelującego takie zachowanie.
- W przypadku modelowania systemów czasu rzeczywistego, gdzie niezwykle istotne jest uwypuklenie kolejności wykonywanych operacji lepszym diagramem może okazać się diagram sekwencji.

W przypadku narzędzia, jakim jest Rational Rose można automatycznie generować jeden diagram interakcji na podstawie definicji drugiego (skrót klawiaturowy F5).

Zadania

1. Na podstawie specyfikacji systemu płacowego, słownika pojęć i ogólnej znajomości problematyki spróbuj znaleźć podstawowe pojęcia, które staną się kluczowymi abstrakcjami naszego systemu.

2. Dla każdego przypadku użycia znajdź wstępny zbiór klas analizy (źródło specyfikacja przypadku użycia, słownik pojęć). Utwórz diagram sekwencji obrazujący współdziałanie klas analizy przy realizacji przypadku użycia.