UML IT

Modelowanie systemów IT.

Spojrzenie z zewnątrz. 4.1.1

Użytkownika „Nie obchodzi, jak to działa, dopóki jak to działa." Jeśli dziś ktoś korzysta z nowoczesnego urządzenia, na przykład, bankomatu, czy telefonu komórkowego, rzadko interesuje go, jak wygląda ten sprzęt od środka.

Przeciętny użytkownik nie dba o to, z jakich części elektronicznych składa się urządzenie lub jakie oprogramowanie zawiera. Z drugiej strony, do czego może służyć maszyna, lub jakie funkcje zapewnia, jest dla niego ważne.

Zwykle potencjalny nabywca telefonu komórkowego jest zainteresowany tym, jak urządzenie może być używane; nie jest zainteresowany tym, jak urządzenie jest zbudowane wewnętrznie, o ile ma pożądane funkcje. Ten typ widoku systemu nazywany jest widokiem czarnej skrzynki, co oznacza system lub urządzenie jest przedstawiony jako czarna skrzynka – nie możesz zajrzeć do środka. Nie wiesz, jak to działa; tylko wiesz, że to działa.

Pokazuje to obraz, jaki powinien mieć użytkownik systemu informatycznego. Wykorzystuje system informatyczny do wykonywania swojej pracy, tak jak ekspres do kawy lub fotokopiarka. Wie, co można zrobić z systemem i zazwyczaj wie też jak to zrobić. Widok zewnętrzny jest istotną częścią modelu systemu informatycznego. Tutaj ustala się, co przyszli użytkownicy oczekują od systemu informatycznego.

Funkcjonalność zdefiniowana w tym widoku powinien docelowo służyć do weryfikacji, czy system informatyczny spełnia wymagania. Widok zewnętrzny składa się z elementów diagramów przypadków użycia, sekwencji przypadków użycia diagramy i prototypy interfejsów.

Diagramy zdefiniowane w UML, które opisujemy pokazują jak uniknąć nieporozumień i błędnych interpretacji. Diagramy są narzędziami opisu wymagań dla systemu informatycznego.

4.1.1

Istotnym elementem systemu jest jego interfejs użytkownika.

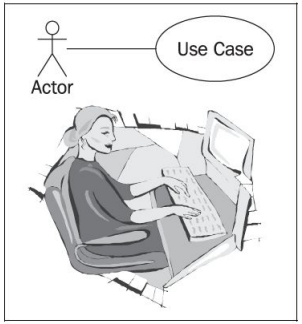
Interfejs użytkownika to jedyny punkt dostępu, jaki użytkownik ma do systemu.

Jeśli na przykład brakuje przycisku nagrywania w dyktafonie, nie można nic nagrać, nawet jeśli urządzenie jest wewnętrznie wyposażony w tę funkcję. Interfejs użytkownika reprezentuje rodzaj widoku funkcjonalności urządzenia.

Rzeczy których w interfejsie nie ma są niedostępne dla użytkownika. Interfejs daje statyczny obraz systemu, obraz ten nie pokazuje w jaki sposób system ma być używany i jakie elementy operacyjne należy zastosować lub w jakiej kolejności wykonać określone czynności. Z tego powodu interfejsy użytkownika wymagają instrukcji obsługi. Oznacza to, że potrzebujemy opis, który identyfikuje możliwe działania i kolejność wykonywanie czynności, aby system był używany w sensowny sposób.

W UML te “instrukcje” (courses) nazywane są przypadkami użycia. Przypadki użycia to instrukcje obsługi interfejsu użytkownika. Tylko to, co jest zdefiniowane w instrukcji obsługi, jest sensownym sposobem działania.

4.1.1



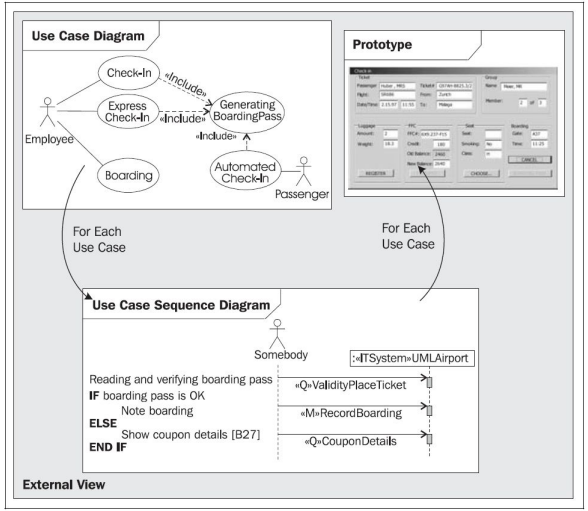
Najlepszym podejściem do modelowania przypadków użycia systemu informatycznego jest wyobrażenie sobie użytkownika, który siedzi przed klawiaturą i pracuje z systemem informatycznym. Użytkownik staje się aktorem, a przypadek użycia jest niczym innym jak abstrakcyjnym opisem aktywności użytkownika.

Aktor:

- Bezpośrednia interakcja z systemem
- Zawsze znajduje się poza systemem, z którym wchodzi w interakcję

W przypadku systemu informatycznego oznacza to, że aktorem jest zawsze ten, który bezpośrednio obsługuje System informatyczny.

Zewnętrzne elementy systemu 4.1.2



Widok zewnętrzny systemu IT składa się z trzech następujących elementów:

• Diagramy przypadków które pokazują wszystkich użytkowników systemu informatycznego (aktorów) oraz wszystkie zadania, które użytkownicy mogą wykonać za pomocą systemu informatycznego (przypadki użycia). Często przedstawianie wszystkich przypadków użycia systemu informatycznego na jednym diagramie nie ma sensu, ponieważ diagram byłby przepełniony.

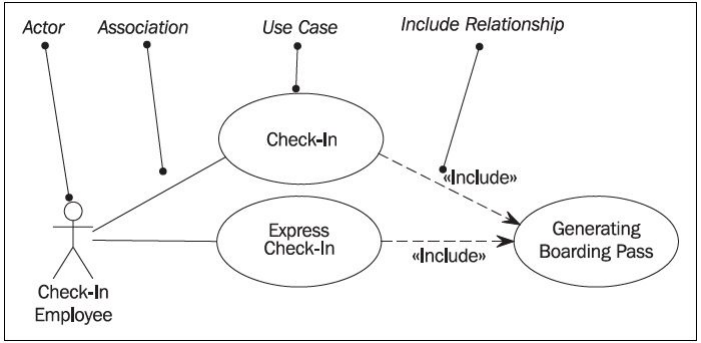
4.1.2

• Diagramy sekwencji przypadków, pokazują procesy podczas interakcji między użytkownikiem i system informatycznym dla celów indywidualnych.

• Prototypy interfejsów pokazują, jak mogą wyglądać interfejsy użytkownika w przypadku użytkowania.

Wszystkie trzy elementy łącznie obrze opisują system IT z perspektywy jego użytkownika.

Przypadek użycia diagramu. 4.1.3

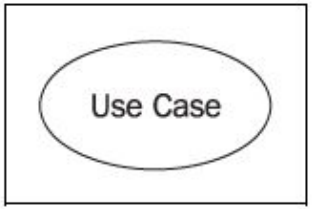


Korzystając z diagramów pracujemy na następujących elementach:

- Aktor - użytkownik systemu IT [indywidualność jest nieistotna]

Aktorzy reprezentują role jakie przyjmują użytkownicy systemu (np. pracownik). Jedna osoba może działać w większej liczbie roli niż jedna w systemie. Ważne dla systemu jest to jaką rolę pełni dany użytkownik. Dlatego do systemów IT konieczne jest logowanie się w określonej roli, np. użytkownik, administrator. Każdemu użytkownikowi odpowiadają określone funkcjonalności oraz dostępy.

4.1.3

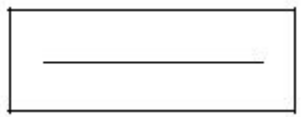


- Przypadek użycia

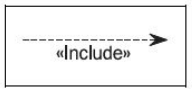
Przypadki użycia opisują interakcje zachodzące między aktorami a systemami IT podczas realizacja procesów biznesowych. Stanowi część funkcjonalności systemu i umożliwia użytkownikowi dostęp do danej funkcji.

Wszystko co mogliby zrobić użytkownicy musi być im nadane (przydział dostępu). Funkcje istniejące w systemach ale nie będące możliwe do użycia nie są dostępne dla użytkowników.

4.1.3

- Asocjacja 

Asocjacja to połączenie między aktorem a przypadkiem użycia. Skojarzenie wskazuje że aktor może przeprowadzić przypadek użycia. Kilku aktorów w jednym przypadku użycia oznacza, że każdy aktor może przeprowadzić przypadek użycia samodzielnie, aktorzy nie wykonują czynności jednocześnie.

- Tworzenie relacji 

Relacja wprowadzona to relacja między dwoma przypadkami użycia.

Oznacza, że przypadek użycia, na który wskazuje strzałka, jest zawarty w przypadku użycia po drugiej stronie strzałki. Umożliwia to ponowne użycie przypadku użycia w innym przypadku użycia.

4.1.4 Zdarzenia

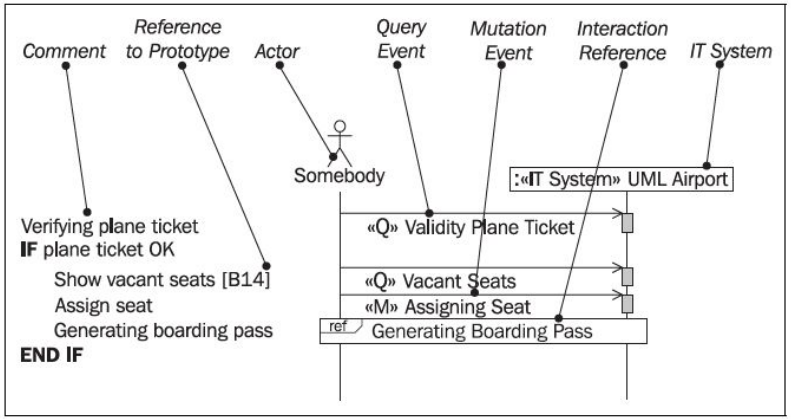
Zdarzenie (event) w UML jest opisem tego co użytkownik robi z systemem. Zdarzenia dzielimy na dwa rodzaje:

- zapytania (query events) - ich celem jest wyświetlenie informacji, nie zmieniają nic w systemie
- mutacje (mutation event) - ich celem jest przechowywanie, przetwarzanie lub usuwanie informacji z systemu

4.1.5 Diagram przypadków użycia

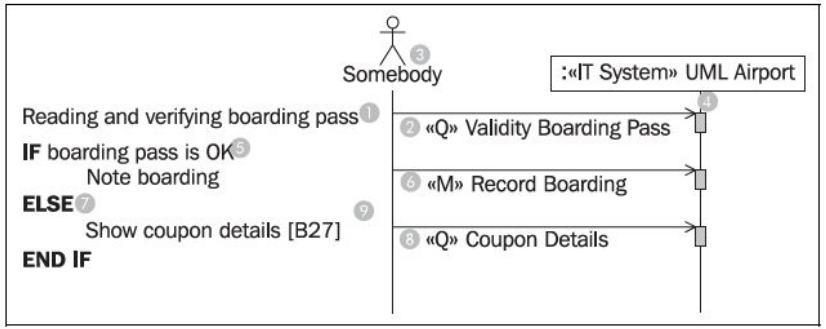
Elementy diagramu:

- komentarz
- odnośnik do prototypu
- użytkownik
- zapytanie
- mutacja
- odnośnik do interakcji
- system IT



4.1.4 Czytanie diagramu

Diagram “boarding”. Karta pokładowa jest czytana i weryfikowana (1) przez wysłanie zapytania (2) od użytkownika (3) do systemu(4). System jest czarną skrzynką więc nie widać z diagramu co dzieje się w środku. Jeśli karta jest OK (5) jest to zapisywane w systemie (6), jeśli nie (7) wyświetlane są szczegóły (8)



4.1.6 Konstrukcja zewnętrznego widoku

Aby skonstruować zewnętrzny widok powinny zostać wykonane następujące kroki:

- zebranie źródeł informacji
- identyfikacja potencjalnych użytkowników
- identyfikacją potencjalnych przypadków użycia
- połączenie przypadków użycia z użytkownikami
- opis użytkowników
- poszukiwanie innych przypadków użycia
- edycja przypadków użycia
- dokumentacja przypadków użycia
- modelowanie relacji pomiędzy przypadkami użycia
- weryfikacja

4.2.1 Obiekty i klasy

Podstawą podejścia obiektowego jest możliwie jak najlepsze odwzorowanie czegoś, co istnieje w świecie rzeczywistym najpierw w modelu, a później w systemie informatycznym.

Reprezentacja ta jednak nigdy nie będzie w pełni odpowiadać rzeczywistości. Musimy się skupić tylko na ważnych dla nas aspektach. Tylko gdy znamy, przynajmniej w przybliżeniu, przeznaczenie systemu informatycznego, możemy budować funkcjonalne obiekty.

W modelach zawsze abstrahujemy od rzeczywistości w sposób ukierunkowany na cel. Ograniczamy nasze rozważania do aspektów ważnych dla bieżącego celu i pomijamy wszystko inne.

4.2.1 Obiekty i klasy

Przedstawiając świat rzeczywisty w modelach abstrakcyjnych, rozróżniamy dwa etapy. W pierwszym kroku, abstrahujemy od pojedynczych osób lub rzeczy do obiektów. W drugim kroku, łączymy podobne obiekty w klasy.

Obiekt reprezentuje dokładnie jeden konkretny przykład ze świata rzeczywistego. Na przykład w bazie danych obiekt odpowiada wpisowi w arkuszu kalkulacyjnym.

Definicja obiektów jest już pierwszym krokiem abstrakcji, ponieważ tylko istotne cechy są modelowane w obiekcie.

4.2.1 Obiekty i klasy

W drugim kroku abstrakcji, łączymy podobne obiekty w klasy. Podobne oznacza, że:

- Cel abstrakcji jest podobny
- Jesteśmy zainteresowani podobnymi cechami
- Obiekty mają podobne zachowanie

W większości przypadków oba etapy abstrakcji są połączone, co oznacza, że klasy są tworzone bezpośrednio. Krok tworzenia obiektów nie jest realizowany wprost.

4.2.1 Obiekty i klasy

Praca z klasami staje się łatwiejsza, gdy weźmiemy pod uwagę, że termin klasa ma dwa różne znaczenia:

- klasa jest wzorcem, według którego tworzone są obiekty
- klasa jest zbiorem obiektów, które zostały utworzone zgodnie z tą klasą.

Klasa jako zbiór zawiera i zna wszystkie swoje obiekty. Można ją sobie wyobrazić jako tabelę w bazie danych, która zna wszystkie swoje wpisy. Zazwyczaj klasy, oprócz atrybutów, zawierają metody, które określają zachowanie obiektów. Jednak w tym podejściu do modelowania systemów informatycznych zasadniczo powstrzymujemy się od korzystania z tej możliwości. Zachowanie obiektów zależy w dużej mierze od ich stanów.

4.2.2 Generalizacja, specjalizacja i dziedziczenie

Terminy takie jak nadklasa, podklasa czy dziedziczenie przychodzą na myśl, gdy myślimy o podejściu zorientowanym obiektowo. Pojęcia te są bardzo ważne, gdy mamy do czynienia z językami programowania zorientowanymi obiektowo, takimi jak Java, Smalltalk czy C++.

W modelowaniu klas mają dużo mniejszą rolę. Powodem tego jest fakt, że modelowanie odpowiednich obiektów lub idei z rzeczywistego świata daje niewielkie możliwości wykorzystania dziedziczenia.

4.2.2 Generalizacja, specjalizacja i dziedziczenie

Generalizacja jest procesem wyodrębniania wspólnych cech z dwóch lub więcej klas i łączenie ich w uogólnioną nadklasę. Współdzielonymi cechami mogą być atrybuty, asocjacje lub metody.

Specjalizacja oznacza tworzenie nowych podklas na podstawie istniejącej klasy. Jeśli okaże się, że pewne atrybuty, asocjacje lub metody mają zastosowanie tylko do niektórych obiektów danej klasy, można stworzyć podklasę. Najbardziej ogólną klasą w generalizacji/specjalizacji nazywana jest nadklasą i zazwyczaj znajduje się na górze diagramu. Bardziej szczegółowe klasy są nazywane podklasami i są zazwyczaj umieszczane poniżej klasy nadrzędnej.

4.2.2 Generalizacja, specjalizacja i dziedziczenie

Relacje pomiędzy nadklasą a podklasą są bardzo ważne.

Do tej relacji odnoszą się poniższe reguły:

- Wszystkie stwierdzenia dotyczące nadklasy odnoszą się również do wszystkich podklas. Mówimy, że podklasy "dziedziczą" atrybuty, asocjacje i operacje z nadklasy.
- Wszystko, co może być zrobione z obiektem nadklasy, może być również zrobione z obiektem podklasy.
- Podklasa musi być specjalną formą nadklasy.

4.2.3 Statyczne i dynamiczne reguły biznesowe

Reguły biznesowe są regułami domenowymi, które są przedstawione w modelu systemu informatycznego. Reguły domenowe mogą pochodzić ze strategii biznesowych, wymagań, wytycznych technicznych i ograniczeń. Reguły biznesowe nie są związane z technologią informacyjną, wynikają one wyłącznie z branży.

Wielu wymagań nie da się zamodelować jako reguł biznesowych. Katalog wymagań jest częścią specyfikacji systemu informatycznego, uzupełniając model systemu IT.

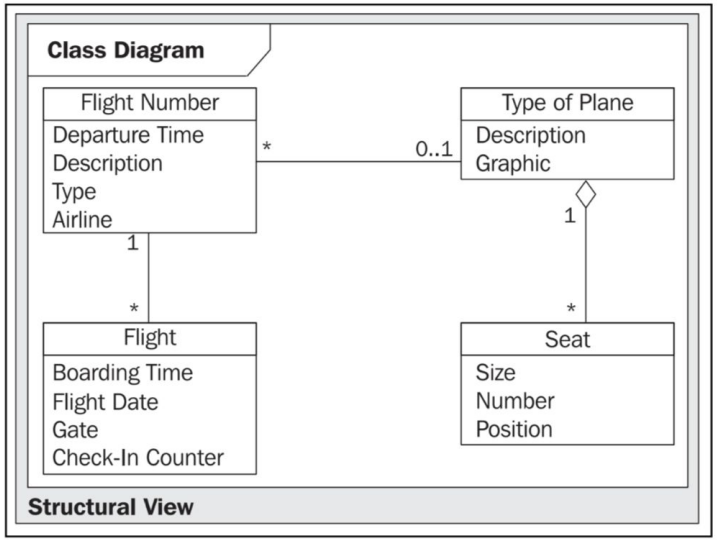
4.2.3 Statyczne i dynamiczne reguły biznesowe

Reguły biznesowe można podzielić na dwie kategorie:

- **Statyczne reguły biznesowe**: Reguły biznesowe, które mogą być weryfikowane w dowolnym momencie. Te reguły biznesowe dotyczą statycznych struktur klas. Te reguły biznesowe mogą być udokumentowane w diagramach klas w widoku strukturalnym.
- **Dynamiczne reguły biznesowe**: Reguły biznesowe, które mogą być weryfikowane tylko w pewnym, gdy coś się wydarzy. Te reguły biznesowe dotyczą dynamicznego zachowania obiektów danej klasy. Te reguły biznesowe mogą być udokumentowane w diagramie diagramu stanów widoku behawioralnego.

4.2.4 Elementy widoku

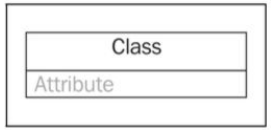
Strukturalny widok systemu informatycznego, jak widać na obrazku - składa się z jednego lub wielu diagramów klas. Diagram klasowy pokazuje wszystkie dostępne klasy w obrębie systemu informatycznego, ich realacje pomiędzy innymi klasami, charakterystyke (atrybuty) oraz zachowanie (metody). Diagramy klas przedstawiaja wewnętrzne struktury statyczne systemu informatycznego.



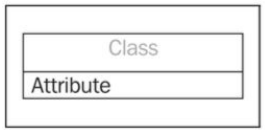
4.2.5 Diagram klas

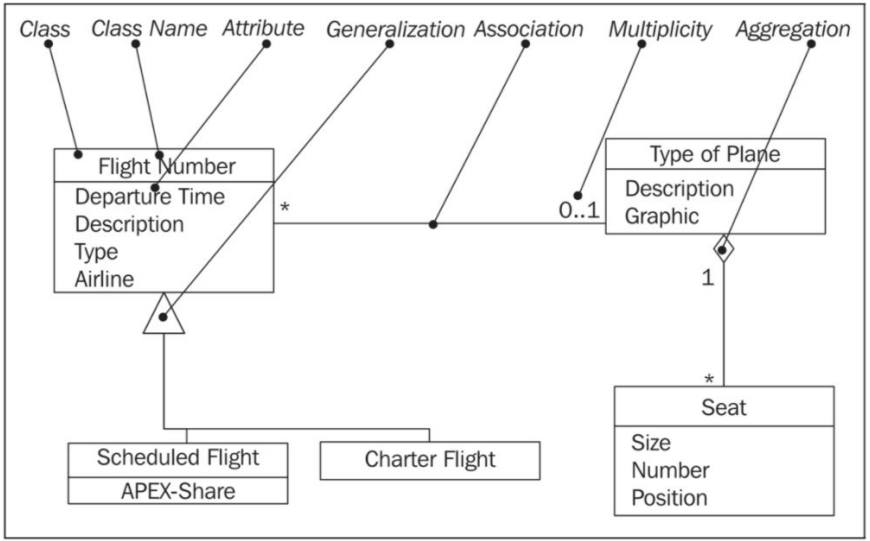
W diagramach klasowych pracujemy z następującymi elementami:

Klasa - reprezentuje odpowiednie pojęcie z domeny, zbiór osób, obiektów lub idei, które są przedstawione w systemie informatycznym (np.: pasaże- rowie, bilety)



Atrybut - reprezentuje charakterystykę klasy, która jest interesująca dla (np.: imie lub wiek pasażera)

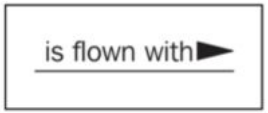


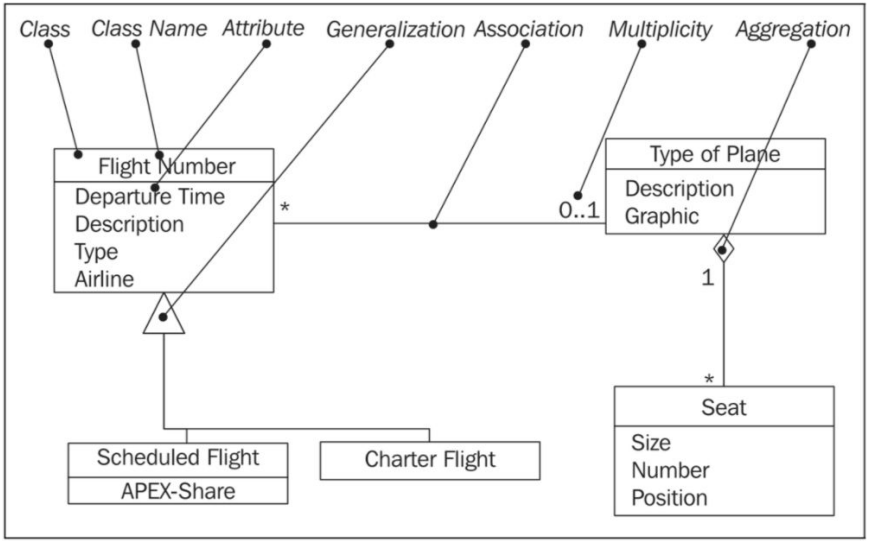


4.2.5

Generalizacja - związek opisujący klasy i podklasy (klasy ogólne i szcze- gółowe)

Asocjacja - wskazuje na silniejsze powiązanie pomiędzy obiektami danych klas (np firma zatrudnia pracowników)

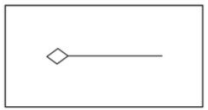


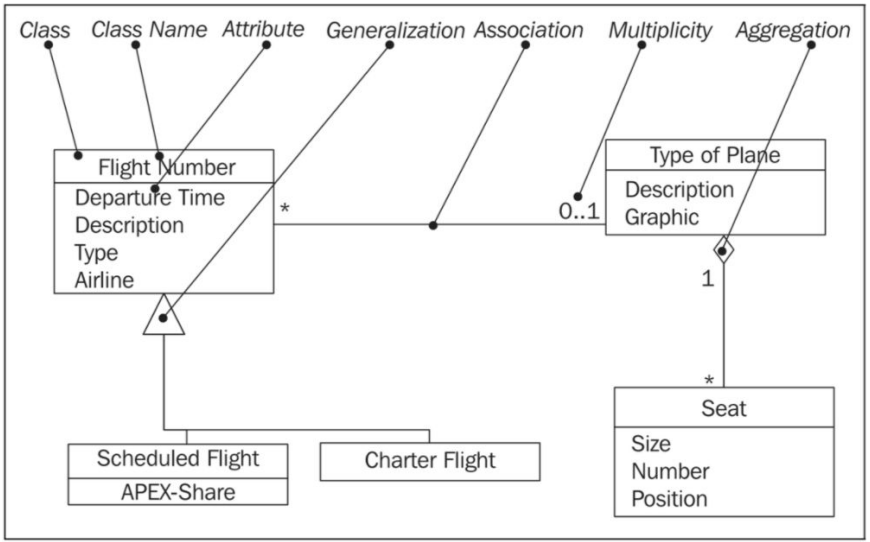


4.2.5

Mnogość - pozwala na stwierdzenie liczby obiektów, które są zaangażowane w asocjację

Agregacja - reprezentuje związek typu całość-część, czyli jakaś większa całość jest rozbita na elementy. Oznacza to, że elementy częściowe mogą należeć do większej całości, jednak mogą istnieć bez niej (np koła i samo- chód)





4.2.6 Czytanie diagramu klas

Patrząc na diagram obok możemy przeczytać powiązania pomiędzy klasą Klient i Bilet w następjący sposób

-Jeden obiekt z pierwszej klasy jest powiązany z wieloma obiektami drugiej klasy

Nazwa pierwszej klasy to Custormer (1), nazwa drugiej to Ticket (4). Nazwa asocjacji jest owns (2):

-Customer (1) owns (2) * (3) ticket (4).

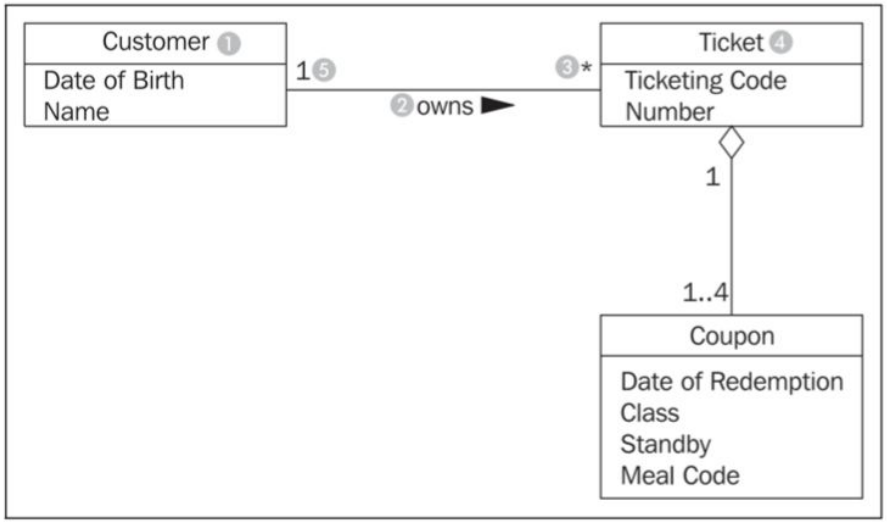
Jeżeli gwiazdkę zamienimy na słowa z języka polskiego (i przetlumaczymy nazwy klas), będzie brzmiało to następująco:

-Klient (1) posiada (2) zero, jeden lub kilka (3) bilet(ów) (4).

Ponieważ asocjacje idą zwykle w oba kierunki, nasze skojarzenie może rownież przyjąć następujące znaczenie:

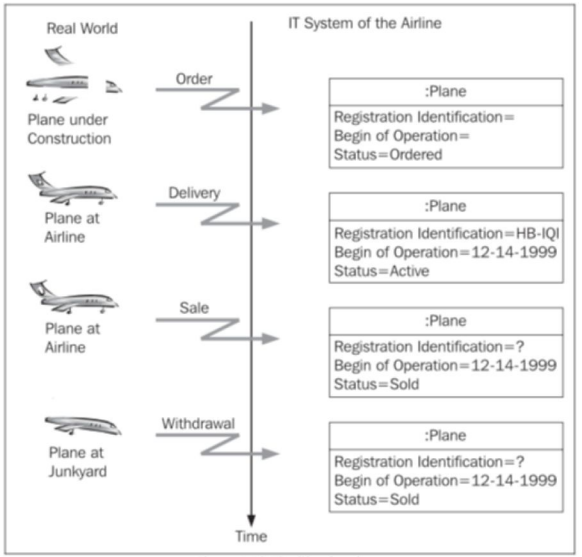
-Bilet (4) jest w posiadaniu (2) przez dokładnie jednego (5) klienta(1)

Mały trójkąt obok nazwy asocjaci (2) wskazuje, w którym kierunku nazwa aso- cjacji jest prawdziwa.



Widok zachowań- 4.3.1 Życie obiektu

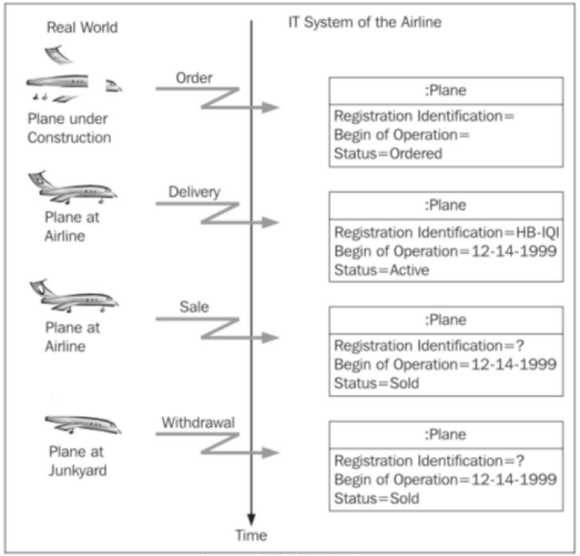
Osoby, obiekty czy koncepty z świata rzeczywistego, które modelujemy jako obiekty w systemach informatycznych posiadają „życia”. Właściwie to nawet dwa życia, to z prawdziwego świata i życie stworzonego obiektu. Chociaż te dwa życia są powiązane niekoniecznie muszą podążać tą samą ścieżką. Zazwyczaj życie rozpoczyna się w chwili narodzin, powstania czy wygenerowania i kończy śmiercią, usunięciem, czy destrukcją. Pomiędzy tymi skrajnymi zdarzeniami życie toczy się mniej lub bardziej uporządkowanym kursem.



4.3.1

By zilustrować to co zostało wspomniane wcześniej skupmy się na życiu samolotu. Samolot, który będzie brany pod uwagę to Airbus A330-223 należący do Swiss International Airlines z identyfikatorem rejestracji HB-IQI.

- Narodziny samolotu Airbus A330-223 (rzeczywistego) wydarzyły się w zależności od perspektywy, od rozpoczęcia jego budowy lub od jego pierwszego lotu.
- Narodziny obiektu jakim jest Airbus A330-223 w systemach informatycznych mają miejsce, gdy informacje na temat samolotu zostały pierwsze raz zebrane i wprowadzone. Może to mieć miejsce w momencie kupna, ponieważ samolot może zostać wprowadzony do systemu w trakcie planowania lub gdy samolot zostanie dostarczony. Inicjator utworzenia obiektu w systemie informatycznym jest zawsze zdarzeniem mutacyjnym.



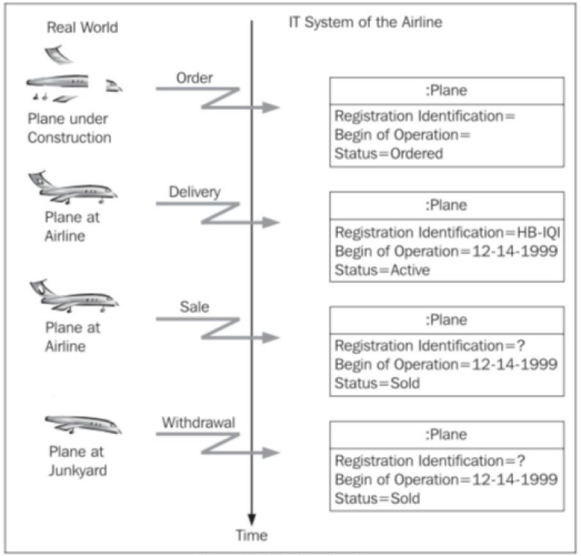
4.3.1

Ponieważ komercyjne samoloty są często sprzedawane przed tym, gdy rozpocznie się ich konstruowanie to daje to możliwość powstania obiektu w systemie IT przed jego rzeczywistym utworzeniem.

- Śmierć rzeczywistego obiektu jest związana z fizycznym zniszczeniem. W przypadku Airbusa śmierć następuje przy wycofaniu lub w wypadku zderzenia samolotu.
- Śmierć obiektu w systemie IT wiąże się z usunięciem jego z systemu Swiss Airlines. Inicjatorem śmierci obiektu zawsze zdarzenie mutacyjne.

Ponieważ komercyjne samoloty często są sprzedawane po pewnym czasie może to skutkować śmiercią obiektu w systemie przed śmiercią rzeczywistą.

Pomiędzy narodzinami i śmiercią obiekt istnieje w systemie IT, będzie wielokrotnie „czytany” i zmieniany. Czytany będzie na skutek zapytań do bazy danych, a zmieniany w wyniku zdarzeń mutacyjnych.



4.3.1

Tak długo jak czytanie i modyfikowanie obiektu nie wiąże się z żadnymi ograniczeniami, to nie jest to zbyt interesujące. Można to przedstawić przy pomocy prostego diagramu stanu. Jeżeli jednak zostaną wprowadzone jakieś zasady modyfikowania takiego obiektu, to bardzo istotną kwestią staje się udokumentowanie tych zasad. Mówimy tutaj o dynamicznych zasadach biznesowych.

Do dynamicznych zasad biznesowych należą takie zasady, które można zastosować w pewnym punkcie w czasie. Głównie, gdy występuje zapytanie do obiektu lub zdarzenie mutacyjne. Zachowanie obiektów jest określone przez takie dynamiczne zasady biznesowe.

Przykłady dynamicznych zasad biznesowych:
-samolot nie może być przypisany do lotu, jeżeli jest w trakcie serwisu.
-samolot nie może zostać wycofany, jeżeli póki ma zaplanowane loty.

4.3.1

Przyglądając się tym zasadom można zauważyć, że odwołują się one do konkretnych wydarzeń w świecie rzeczywistym i stanu w jakim znajduje się obiekt.

- zdarzenie mutacyjne przypisywania samolotowi lotu nie jest dozwolone w trakcie, gdy obiekt jest w stanie „serwisowany”.
- zdarzenie mutacyjne wycofania samoloty nie jest dozwolone, gdy obiekt jest w stanie „zaplanowano lot”.
- zdarzenie mutacyjne uruchamianie samolotu nie jest dozwolone, gdy obiekt jest w stanie „tranzyt”.

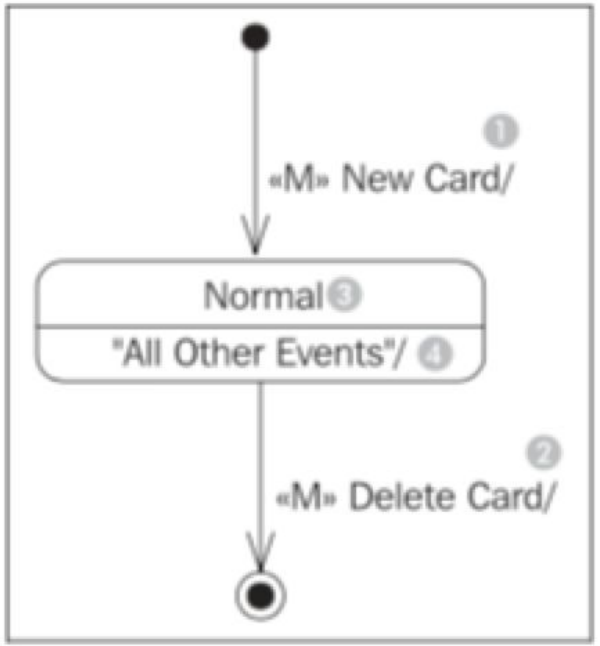
Innymi słowy: dla każdego zdarzenia powinno się być w stanie określić czy owe jest dozwolone w aktualnym stanie, w którym znajduje się obiekt i jak obiekt na nie zareaguje.

W widoku zachowań stosuje się jeden diagram stanu na daną klasę w celu dokumentowania jakich zasad należy przestrzegać.

4.3.1

Przedstawiono obok prosty diagram dla klasy „karta stałego klienta” Nowy obiekt powstaje w wyniku zdarzenia <<M>> New Card. Obiekt może zostać usunięty w wyniku zdarzenia <<M>> Delete Card. Pomiędzy nimi obiekt jest w stanie „normalny”, w którym „wszystkie inne wydarzenia” są dostępne.

.

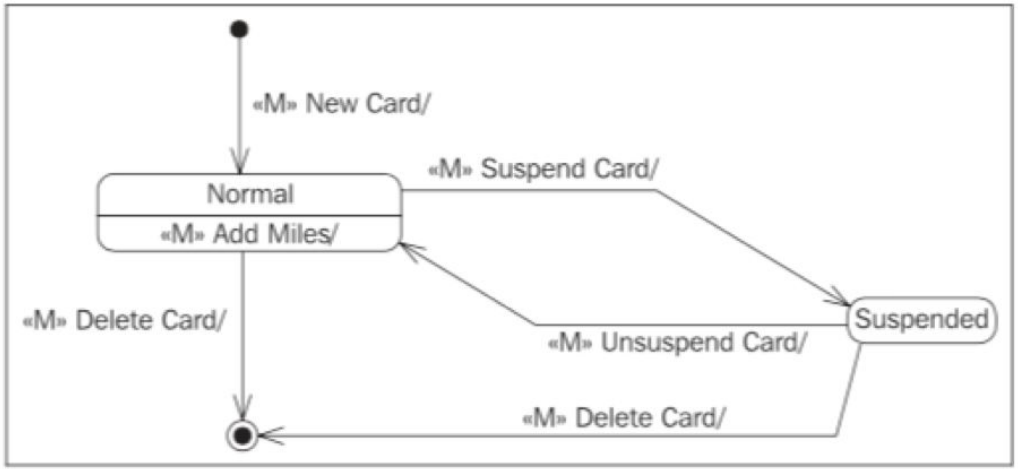


4.3.1

Jeżeli jednak dodamy więcej zasad biznesowych diagram zacznie się znacznie komplikować.

Chcielibyśmy zmodyfikować diagram dodając poniższe zasady:
-musi być możliwość zawieszenia lub przywrócenia karty stałego klienta,
-nie ma możliwości zwiększenia dostępnych kilometrów (mil) do zawieszonej karty stałego klienta.

Po modyfikacji przy wykorzystaniu powyższych zasad diagram przybrał formę:

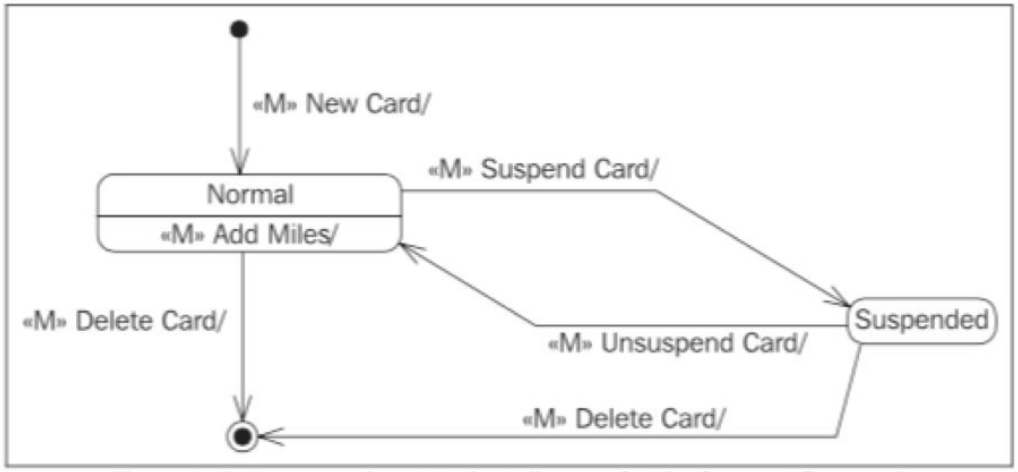


4.3.1

Diagramy stanu takie jak dla karty stałego klienta obrazują jakie ścieżki lub pomiędzy jakimi granicami toczy się życie obiektu karty stałego klienta. Przy pomocy diagramu możemy rozpoznać dozwolone i niedozwolone łańcuchy zdarzeń.

Przykład dozwolonego: new card, add miles, add miles, add miles, suspend card, delete card.

Przykład niedozwolonego: new card, add miles , suspend card, add miles, delete card. Przedostatnie zdarzenie nie jest dozwolone.



4.3.1

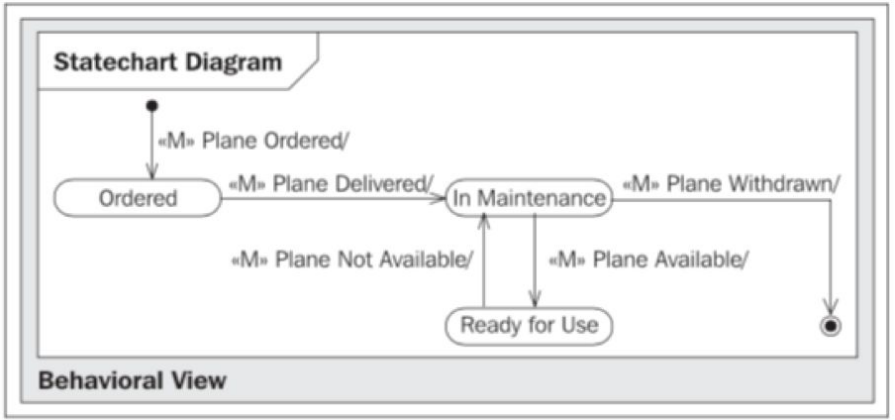
Generalizując życie obiektu musi toczyć się po dozwolonych ścieżkach, czyli obiekt musi przestrzegać pewnych zasad. Tym samym widok zdarzeń jest szczególnie ważny, ponieważ zadaniem systemu informatycznego jest zapewnienie, że zadane zasady będą przestrzegane. Ważne jest więc by te zasady udokumentować w sposób poprawny i całkowity, ponieważ w przeciwnym razie może to prowadzić do nieporozumień po stronie użytkownika i dewelopera. W dokończonym systemie IT nie powinno być możliwości dla użytkownika aby dokonać niedozwolonych zmian.

Na przykład:
W biurze handlowym obiekt odpowiada księdze np. księdze zamówień i kanceliście odpowiedzialnym za tą księgę. Diagram stanu takiego obiektu musi zawierać wszystkie zasady, które kancelista musi zastosować, gdy obsługuje tą księgę, jest to dla niego swoista instrukcja, która przykładowo mówi, że zamówienie zostało już dostarczone, ale nie zostało jeszcze opłacone, więc nie może zostać anulowane.

4.3.2 Elementy rzutu

Widok zachowań przedstawiony jest poprzez wiele diagramów stanu, każdy z nich pokazuje zachowanie konkretnego obiektu, więc wszystkie diagramy stanu składają się na obraz życia obiektu w systemie informatycznym. W praktyce jednak często nie wszystkie diagramy się tworzy, a jedynie, te które:

-Zawierają sporo zasad lub ważne zasady biznesowe lub
- Opisują istotne obiekty



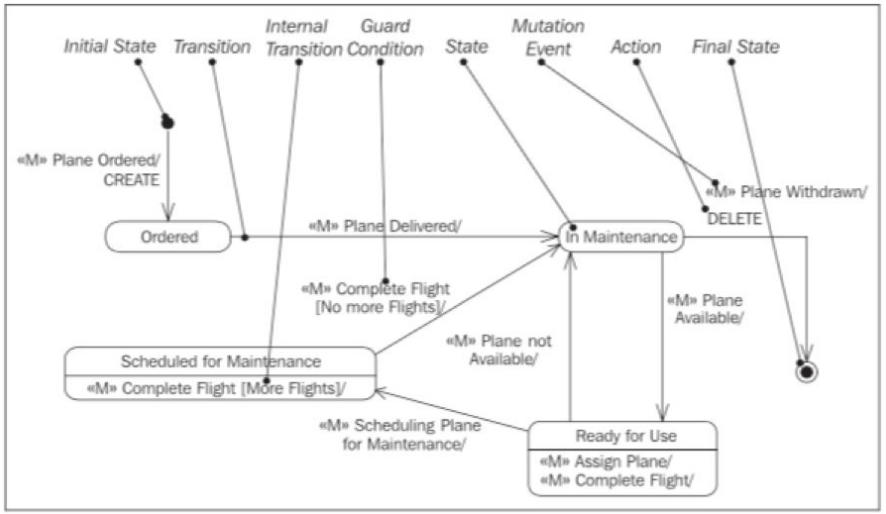
4.3.3 Diagramy stanu

Diagramy stanu składają się na następujące elementy:
- stan początkowy
Stan początkowy stanowi źródło wszystkich obiektów, przedstawia się go symbolem:

Nie jest to stan normalny, ponieważ obiekt jeszcze nie istnieje

-stan
Stan obiektu jest zawsze określony przez jego atrybuty i powiązania. Stany w diagramach stanu reprezentują zbiór tych kombinacji wartości, dla których obiekt zachowuje się tak samo w odpowiedzi na dane zdarzenie:

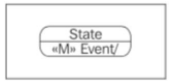
W związku z tym nie każda modyfikacja atrybutu skutkuje nowym stanem



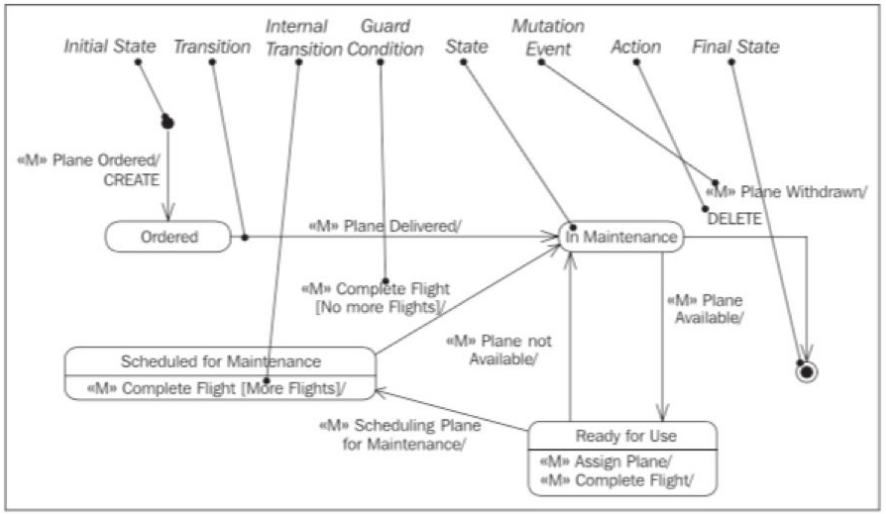
4.3.3

-Przejście
Przejście reprezentuje zmianę pomiędzy poszczególnymi stanami:

-przejście wewnętrzne
Przejście wewnętrzne to przejście z danego stanu do niego samego, to oznacza, że ten obiekt obsługuje zdarzenie bez zmiany stanu:

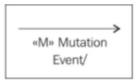


Zdarzenia, które inicjują przejście wewnętrzne są zapisywane w dolnej części symbolu stanu, np. dla karty stałego klienta linii lotniczej obiekt w stanie normalnym pozostaje w tym stanie, kiedy następuje zdarzenie <<M>> add miles.



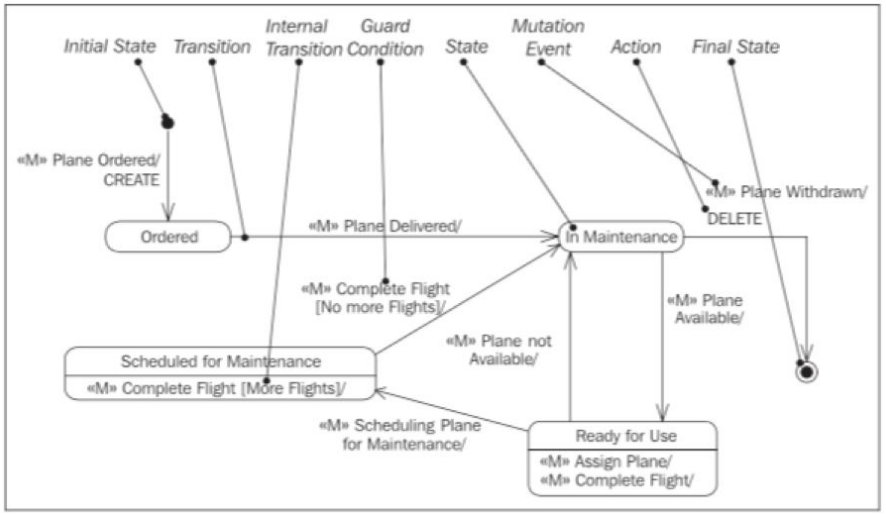
4.3.3

-Zdarzenie mutacyjne
Zdarzenie mutacyjne to inicjator przejścia z jednego stanu do drugiego lub przejścia wewnętrznego:



-akcja
Akcja to czynność jaką wykonuje obiekt w odpowiedzi na zdarzenie:

-



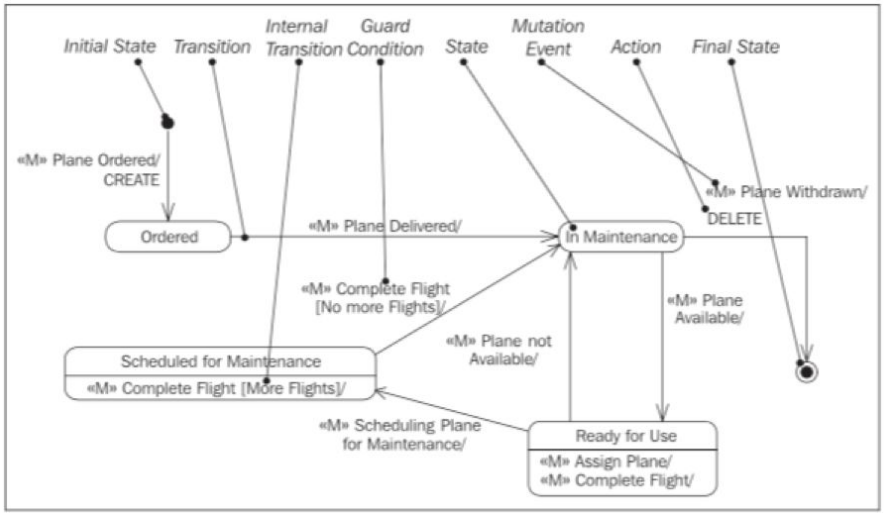
4.3.3

Warunek ochronny
Warunek ochronny to warunek, który musi zostać spełniony by umożliwić przejście, do którego jest przypisany.

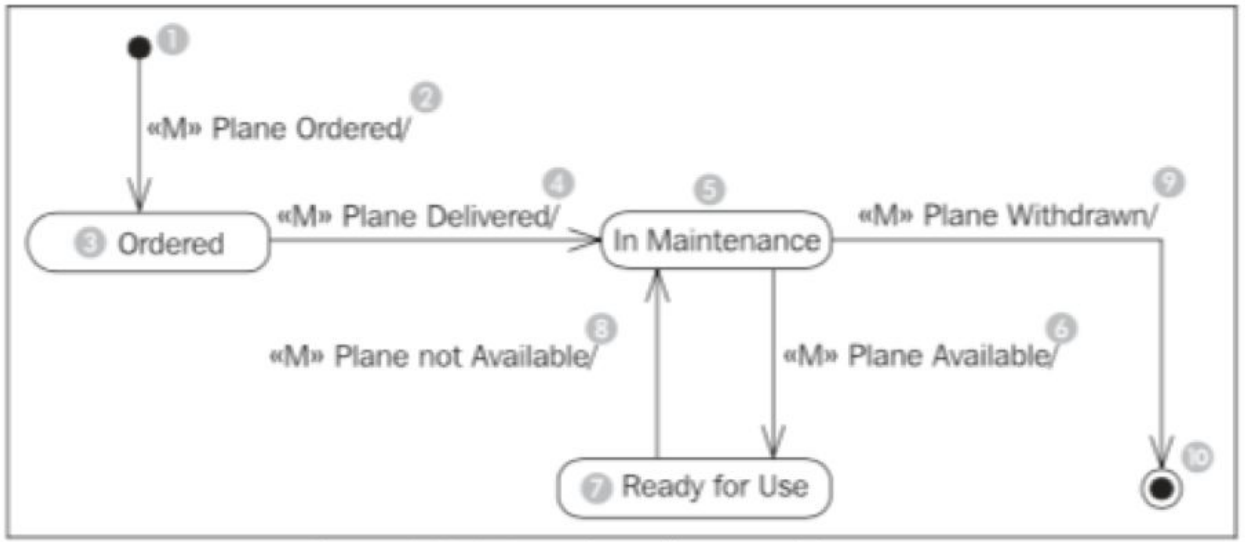
Warunek ochronny może zostać użyty do udokumentowania jakiegoś zdarzenia. W zależności od warunku może ono doprowadzić do innego przejścia.

-Stan końcowy
Stan końcowy reprezentuje koniec istnienia obiektu:

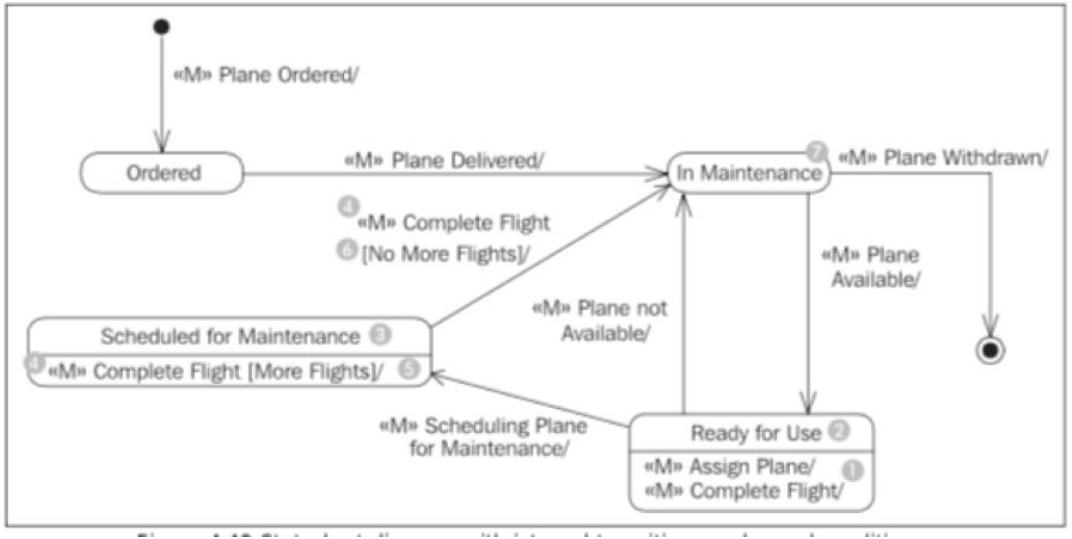
Stan końcowy nie stanowi stanu rzeczywistego, ponieważ obiekty w tym stanie już nie istnieją.



4.3.3 - Czytanie diagramów stanu



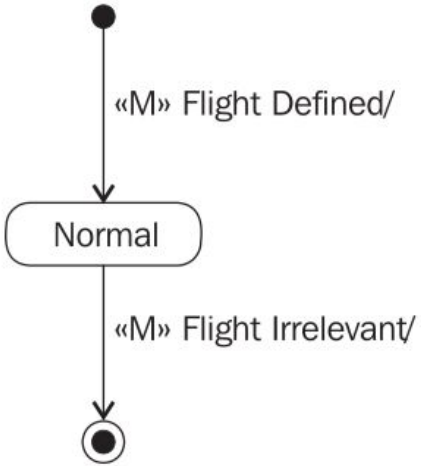
4.3.3 - Czytanie diagramów stanu



4.3.4 Konstruowanie diagramów

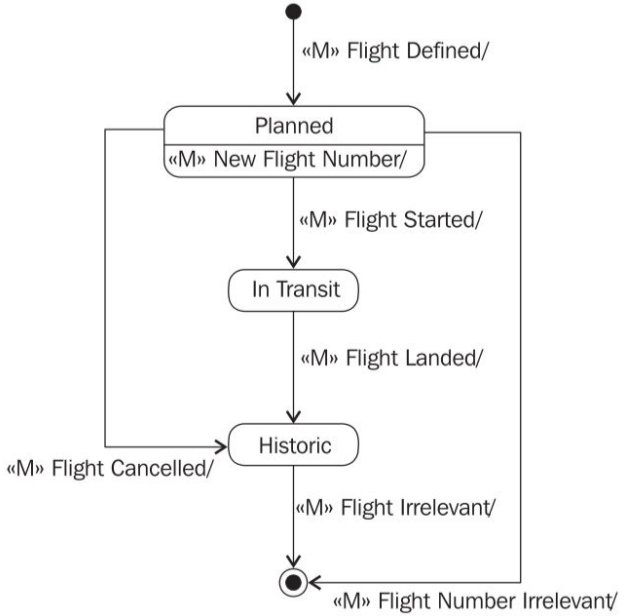
Jako podstawowy szkic możemy skonstruować prosty diagram, który składa się ze stanu podstawowego, normalnego oraz początkowego. Teraz możemy dodawać każde nowe uzyskane zdarzenie. Trzeba pamiętać, że trzeba zadać sobie parę pytań dodając każde zdarzenie:

- Czy zdarzenia mutacji dozwolone są we wszystkich przypadkach?
- W jakim stanie znajduje się obiekt po wystąpieniu zdarzenia?
- Czy przejście do nowego stanu zależy od określonych warunków?



4.3.4 Konstruowanie diagramów

Na przykład, zdarzenie **<M>** jest dozwolone tylko wtedy, gdy lot nie jest już w stanie tranzytu. Po udzieleniu odpowiedzi na wszystkie pytania dotyczące zdarzeń, tworzony jest diagram.



4.4.1 Co się dzieje wewnątrz systemu IT

W większości przypadków zwykłe podkreślenie struktur nie jest wystarczające do zrozumienia systemu. Nawet jeśli posiadamy dokładny plan rafinerii oleju to trudnym jest zrozumienie jak olej jest zamieniany w benzynę. Tylko wtedy, kiedy proces rafinerii jest dokładnie wyjaśniony staje się on zrozumiały. Jest to również prawdziwe dla modelowania systemu IT. Statyczny widok samych klas nie jest wystarczający by zrozumieć system IT. Dla przykładu, co stanie się w systemie IT, jeśli pasażer się zamelduje? Jakie procesy zachodzą? Których obiektów dotyczą? Na te pytania można odpowiedzieć przez widok interakcji.

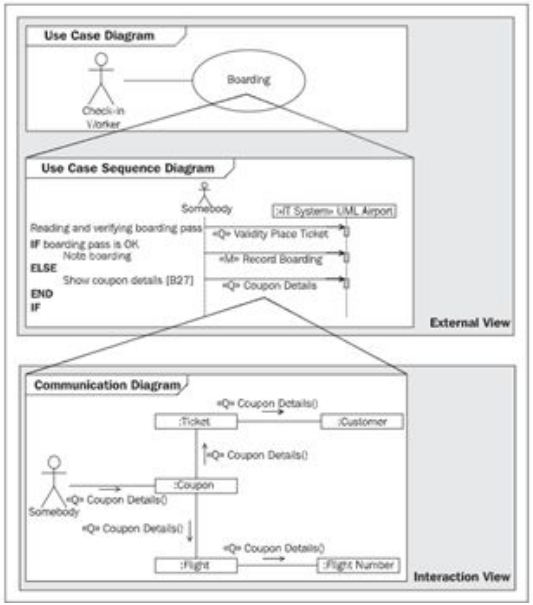
4.4.1 Co się dzieje wewnątrz systemu IT

Ważnym jest fakt, że modelowanie widoku interakcji dokłada wiele do weryfikacji oraz kompletności widoku statycznego. Mając do czynienia z diagramem klas zaczynając od modelowania zapytań aż do poziomu indywidualnych atrybutów pewnym jest, że diagram klas spełnia nasze wymagania. Aspekt ten jest ważny w zależności od tego jak kompletny jest diagram klas w chwili modelowania widoku interakcji.

- Uczciwie kompletny diagram klas jest weryfikowany przez modelowanie widoku interakcji
- Niekompletny diagram klas jest ulepszony i dopełniony przez modelowanie widoku interakcji

4.4.1 Co się dzieje wewnątrz systemu IT

Istnieje bliska relacja między widokiem interakcji a przypadkami użycia. Przypadki użycia pokazują zewnętrzny obraz. System It jest wyświetlany jako czarne pudło. W widoku interakcji to czarne pudełko jest otwierane i to co zachodzi wewnątrz systemu IT jest ujawniane. Widok interakcji ilustruje które obiekty są konieczne do przetworzenia danego zadania oraz jak obiekty komunikują się wzajemnie. UML używa dwóch typów diagramów by modelować widok interakcji: diagram komunikacji oraz diagram sekwencji.

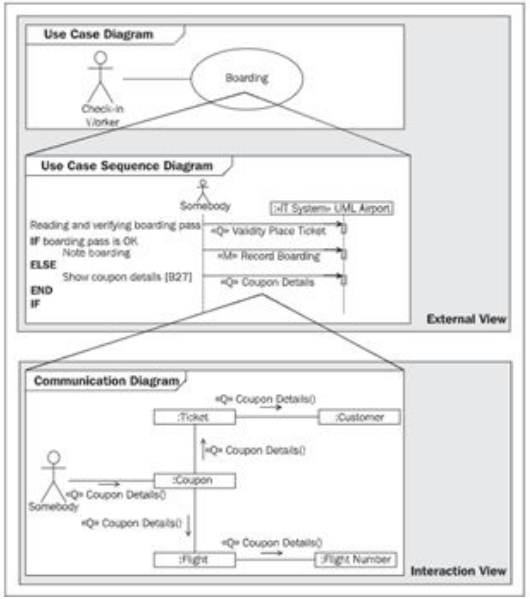


4.4.1 Co się dzieje wewnątrz systemu IT

Rysunek przedstawia nam relacje pomiędzy przypadkami użycia a diagramem komunikacji dla widoku interaktywnego:

- Na samej górze widać diagram przypadków użycia z przypadkiem użycia o nazwie boarding
- Przypadek użycia boarding jest opisany w diagramie sekwencyjnym przypadków użycia. Ten opis zawiera łańcuch wydarzeń zapytania oraz zdarzeń mutacyjnych.
- Przepływ każdego zdarzenia zapytania jest opisany w diagramie komunikacji

Wraz z zdarzeniem mutacyjnym proces jest analogiczny: ich przepływ jest opisywany w diagramie sekwencji.



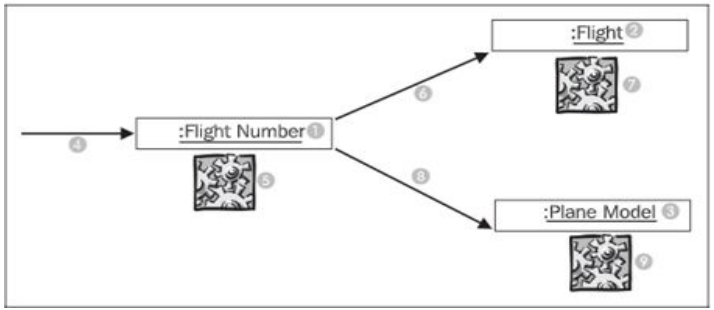
4.4.1 Co się dzieje wewnątrz systemu IT

Aby dokładnie zrozumieć diagramy widoku interakcji należy pokazać jak obiektowo zorientowany system działa od wewnątrz. System funkcjonuje przez obiekty, które samodzielnie wykonują pracę lub zlecają ją innym obiektom. To dokładnie przedstawia jak system IT jest modelowany przy pomocy UML. Nie jest tu ważne czy system IT jest zaimplementowany przy pomocy obiektowo zorientowanej technologii. System IT jest modelowany na wysokim poziomie abstrakcji, która jest odłączona od jej designu oraz programowania.

4.4.1 Co się dzieje wewnątrz systemu IT

Powyższy obrazek pokazuje współpracę trzech obiektów w systemie IT: **Numer lotu (1), Lot (2)** oraz **Model samolotu (3).** Naszym zadaniem będzie usunięcie lotu (przykładowo o numerze SR9011) z systemu IT.

W naszym modelu z przypadku użycia **anulowanie numeru lotu (4)** wysłano **zdarzenie mutacyjne** do **Numeru lotu.** Teraz obiekt **Numer lotu** może stać się aktywnym (5). Dla przykładu może on zweryfikować, czy prośba o usunięcie jest możliwa. Może on także wysłać zdarzenie (6) (8) do pozostałych obiektów (2) (3) które muszą stać się aktywne (7) (9).



4.4.1 Co się dzieje wewnątrz systemu IT

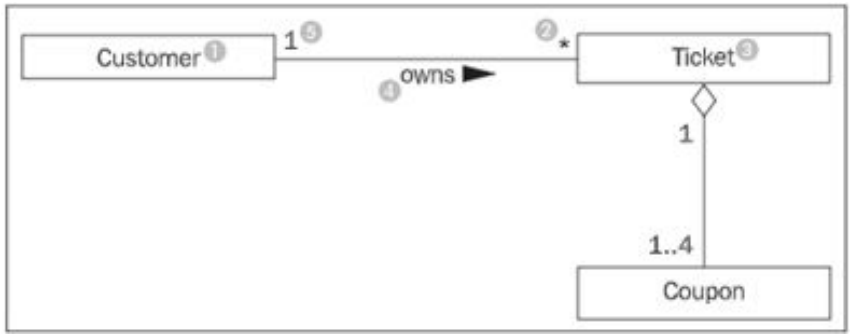
Dokładnie ten typ współpracy pomiędzy obiektami jest dokumentowany w diagramie komunikacji oraz diagramie sekwencji dla widoku interakcji. Tutaj nacisk nałożony na obiekty połączone oraz zdarzenia przesłania. To co zachodzi wewnątrz obiektów, czyli zachowanie się indywidualnych obiektów (zmodyfikowanie wartości atrybutów, przeliczenia, usunięcia obiektów itd.) nie mogą być widziane w tym diagramie. Zachowanie się obiektów jest zdefiniowane w widoku zachowań. Widok zachowań przedstawia dla każdego obiektu co dokładnie zachodzi, kiedy określone zdarzenie dociera do obiektu.

4.4.1 Co się dzieje wewnątrz systemu IT

W biurze handlowym widok interakcji mógłby być przedstawiony następująco: Idziemy do sekretarki i obserwujemy, jak wykonuje zadanie, dla przykładu przeliczanie obligacji. Tutaj obserwujemy w jakiej kolejności udaje się ona do indywidualnych sprzedawców z ich książkami, aby poprosić ich o coś (zapytanie) lub zlecić im aby zmienili coś w swoich książkach (mutacja).

4.4.1 Co się dzieje wewnątrz systemu IT

Współpraca obiektów przedstawiona jest na poniższym rysunku bazującym na mechanizmie przesyłania zdarzeń, oznacza to, że obiekty mogą przesyłać między sobą zdarzenie do poszczególnego obiektu, obiekt musi być w tym przypadku znany. Dla przykładu, jeśli zdarzenie ma być przesłane z obiektu (3) do odpowiedniemu obiektowi z (1) jest ot możliwe wtedy, kiedy obiekt (3) wie czym jest obiekt (1). Dokładne taka informacja jest dokumentowana w diagramie klas statycznego widoku:



Na powyższym diagramie przedstawione jest, że klient (1) posiada zero, jeden lub kilka biletów (3) a bilet jest posiadany przez dokładnie jednego klienta.

4.4.1 Co się dzieje wewnątrz systemu IT

Obiekty, które są połączone znają się nawzajem. Jest to wstępnym wymaganiem, aby możliwe było wysyłanie zdarzeń. Normalnie w widoku interakcji zdarzenia są przesyłane po powiązaniach w diagramie klas. Jako alternatywę można dołączyć identyfikację obiektu, do którego zdarzenie powinno być wysłane jako parametr zdarzenia.

Diagram UML którego używamy do zilustrowania widoku interakcji, diagramów sekwencji oraz diagramów komunikacji, są połączone pod ogólną nazwą diagramów interakcji.

4.4.1 Co się dzieje wewnątrz systemu IT

Oba diagramy pozwalają na podobny widok systemu IT. Oba diagramy pokazują przepływ współpracy pomiędzy obiektami. Jednakże te dwa diagramy różnią się w następujących punktach:

- W diagramie sekwencji sekwencja chronologiczna natychmiastowo staje się widoczna. Pionowa oś czasu wskazuje wyraźną sekwencję od góry do dołu. Diagramy komunikacji nie posiadają tej osi. Możliwe sekwencje muszą być udokumentowane przez numerowanie zdarzeń.

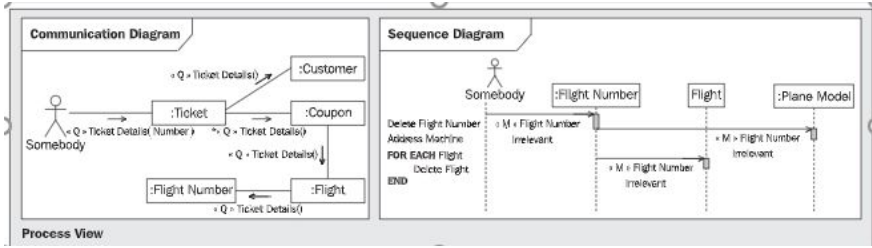
- Diagramy komunikacji są bardziej podobne do statycznych diagramów klas. Nazwy klas oraz atrybuty mogą być przedstawione. To czyni możliwym prostsze udokumentowanie poszczególnych przepływów takich jak odczyt informacji. W diagramie sekwencji nie możemy zilustrować które atrybuty mogą być odczytane przez obiekt.

- Diagramy komunikacji działają dobrze, jeśli chodzi o pokazanie równoległych ścieżek interakcji. Jednakże jest to również możliwe używając diagramu sekwencji, te diagramy łatwo mogą stać się zbyt złożone.

4.4.2 Elementy widoku

Powyższa ilustracja przedstawia widok interakcji systemów IT i składa się ona z dwóch elementów:

- Diagramów komunikacji dokumentujących przepływ zapytań wewnątrz systemu IT. Każde zapytanie jest częścią przypadku użycia.
- Diagramów sekwencji dokumentujących przepływ zdarzeń mutacyjnych wewnątrz systemu IT. Zdarzenia mutacyjne także są częścią przypadku użycia.



Oba diagramy pokazują jak obiekty systemu IT współpracują w przetwarzaniu zdarzeń. W ten sposób każde zdarzenie zapytania z przypadków użycia staje się własnym diagramem komunikacji oraz każde zdarzenie mutacyjne staje się własnym diagramem sekwencji.

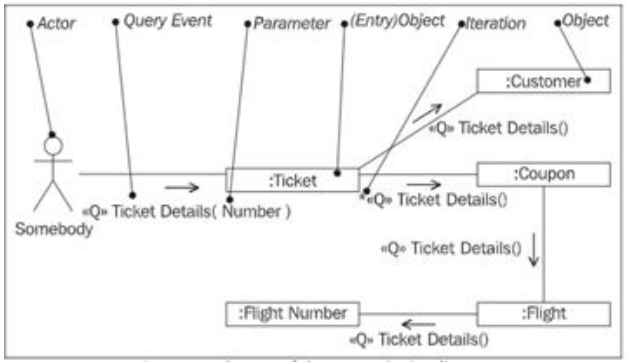
4.4.2 Elementy widoku

W rzeczywistości nie dokumentujemy każdego przepływu każdego zapytania oraz każdej mutacji. Wysiłek w to włożony byłby za duży. Jedynie co to dokumentujemy te przepływy, które są szczególnie ważne albo złożone. Następujące założenia powinny dokonać właściwego wyboru:

- Dla każdego zdarzenia zapytania musimy zweryfikować, czy wszystkie konieczne klasy, atrybuty oraz powiązania są obecne w diagramie klas. Dla prostych zapytań może być wystarczające, by sprawdzić konieczne elementy danych na wydruku. Dla takich zapytań diagram komunikacji nie jest potrzebny.

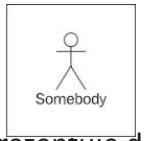
- Zapytania, które są bardzo ważne dla użytkownika, lub bardzo złożone powinny być udokumentowane w diagramie komunikacji.

4.4.3 Diagram komunikacji



W diagramach komunikacji podobnych do tych przedstawionych na powyższej ilustracji pracujemy z następującymi elementami:

Aktor „Ktoś”



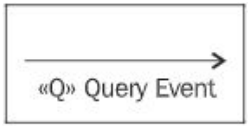
Aktor **„ktoś”** reprezentuje dowolnego aktora z diagramu przypadków użycia. Ponieważ zdarzenie zapytania jest udokumentowane w diagramie komunikacji może być ono zawarte w kilku przypadkach użycia i ponieważ te przypadki użycia posiadają różnych aktorów używamy aktora jako „ktoś”:

W ten sposób nie musimy powierzać nas samych poszczególnemu aktorowi. (W diagramach komunikacji aktorzy mogą również być powierzeni samym sobie.)

4.4.3 Diagram komunikacji

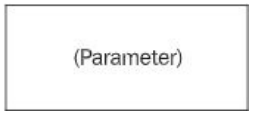
Zdarzenie zapytania

Zdarzenie zapytania reprezentuje zapytanie o informację:



Normalnie zdarzenie zapytania z przypadku użycia jest wysłane do systemu IT, dla przykładu zapytanie o dokładną informacje dotyczącą biletu.

Parametr



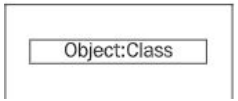
Parametry pozwalają na dołączenie informacji do zdarzenia, przykładowo numer biletu, aby prawidłowy bilet mógł zostać odczytany.

4.4.3 Diagram komunikacji

Iteracja

Iteracja wskazuje wszystkie obiekty, dla których istnieje powiązanie odbioru zdarzenia, dla przykładu wszystkie kupony biletu.

Obiekt oraz obiekt wejściowy



Obiekt reprezentuje obiekt klasy w widoku statycznym, dla przykładu „Jan Kowalski”, który jest obiektem klasy **pasażer**.

Obiekt wejściowy jest pierwszym obiektem, który otrzymuje zdarzenie zapytania od aktora. W nim rozpoczyna się ścieżka iteracji.

4.4.3 Diagram komunikacji

Odczytywanie diagramu komunikacji

Poniższa ilustracja przedstawia diagram komunikacji wraz z aktorem **„kimś”** oraz obiektami **bilet, klient, kupon, lot** oraz **numer lotu**. Diagram dokumentuje przepływ zapytania **<<Q>> szczegółów kuponu.**

Zaczynając od lewej diagram komunikacji jest odczytywany następująco: Aktor **„ktoś”** (1) wysyła zdarzenie zapytania **<<Q>> szczegółów kuponu** (2) do obiektu z klasy **kupon** (3).

4.4.3 Diagram komunikacji

Aktor „ktoś”

W naszym modelu systemu IT, przypadki użycia są źródłem zdarzeń. To co jest udokumentowane w diagramie komunikacji zachodzi w kontekście przypadków użycia. Jest dowiedzione, że wartościowe jest użycie niezdefiniowanego aktora **„kogoś”** (1) zamiast aktora z przypadków użycia. Przepływ zdarzeń, które są opisane w diagramie komunikacji może zajść w różnych przypadkach użycia z różnymi aktorami. Aktor **„ktoś”** zastępują aktora w przypadku użycia z których zdarzenie zapytania **<<Q>> szczegółów kuponu (2)** zaczyna się rozgałęziać.

4.4.3 Diagram komunikacji

Obiekt **kupon (3)** dostarcza atrybutów – **data wykupu, klasa** oraz **gotowość** (4) i przekazuje dalej zdarzenie zapytania **<<Q>> szczegółów kuponu (5)** do dwóch innych obiektów: **lot (6),** który należy do **kuponu** oraz do obiektu **bilet (7),** który również należy do **kuponu**.

Te dwa obiekty w kolejności dostarczają pewne atrybuty (8) oraz następnie przesyła dalej zdarzenie zapytania **<<Q>> szczegółów kuponu (9, 10)**. W ten sposób diagram komunikacji może być wykorzystany do udokumentowania „zbioru” atrybutów jako reakcji na zdarzenie zapytania.

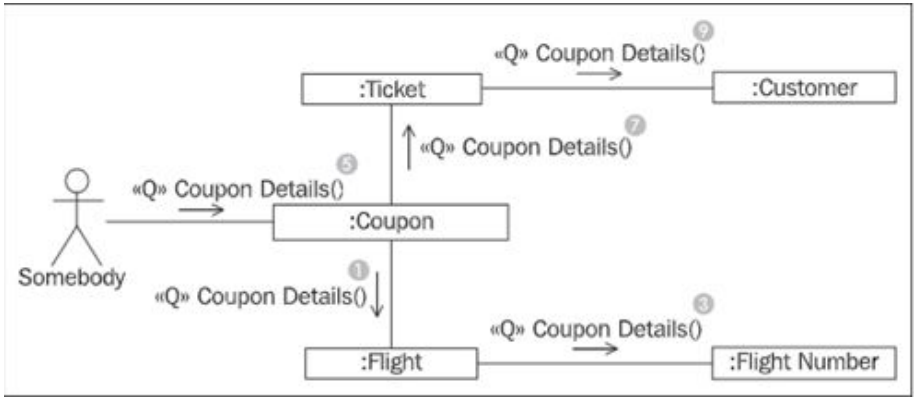
4.4.3 Diagram komunikacji

W przeciwieństwie do diagramów sekwencyjnych, diagramy komunikacyjne nie posiadają wymiaru czasowego. Obiekty mogą się rozprzestrzeniać po diagramie w dowolny sposób. Porządek, przy którym zdarzenia są przetwarzane mogą być dla nich tylko częściowo widoczne.

4.4.3 Diagram komunikacji

Można sformułować następujące stwierdzenia na podstawie diagramu na powyższej ilustracji:

- Pierwsze zdarzenie jest wysłane z aktora do **kuponu (5)**
- Następnie sekwencja nie jest zdefiniowana:
  - Z jednej strony zdarzenie przechodzi do **lotu** (1) i następnie do **numeru lotu** (3)
  - Z drugiej strony zdarzenie przechodzi do **biletu** (7) oraz następnie do **klienta** (9)



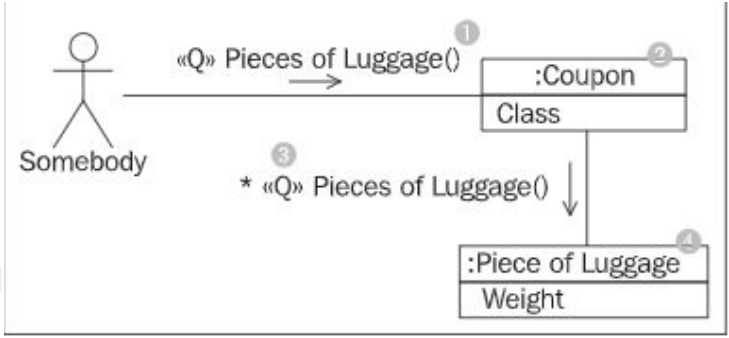
4.4.3 Diagram komunikacji

Przepływ zdarzenia rozgałęzia się przy kuponie bez jakiejkolwiek kolejności. W większości przypadków kolejność nie jest ważna. Jednakże, jeśli kolejność jest ważna, UML pozwala na numerowanie sekwencji zdarzeń w diagramie komunikacji.

4.4.3 Diagram komunikacji

Iteracja wskazuje, że wszystkie obiekty (nie jeden szczególny) są zaadresowane.

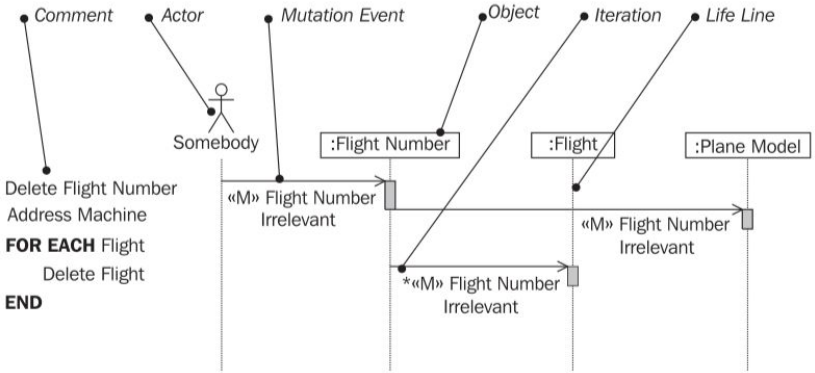
Możemy odczytać z powyższej ilustracji, że zdarzenie zapytania **<<Q>> szczegółów kuponu** (1) jest pierwsze wysłane do **kuponu (2)** a następnie jest wysłany do wszystkich (3) (iteracja) połączonych **części bagażu** (4). Iteracja jest udokumentowana znakiem (*) przed każdą nazwą zdarzenia.



4.4.4 Schemat sekwencji

W diagramach sekwencji pracujemy z następującymi elementami:

- Comment
- Actor ”Somebody”
- Mutation Event
- Object
- Iteration
- Lifeline



4.4.5 Widok interakcji- Konstruowanie diagramów komunikacyjnych

Wynik kwerendy roboczej — czego chcemy?
Punktem wyjścia do modelowania zdarzenia zapytania jest oczekiwany wynik. Ogólnie rzecz biorąc,pożądanym rezultatem jest wyświetlenie na monitorze lub wydrukowany dokument, na przykład lista lub paragon.Aby każde zdarzenie za- pytania zostało udokumentowane na diagramie komunikacyjnym, musimy wie- dziećjak ma wyglądać wynik. Należy zadać następujące pytania:

-Jak (dokładnie) ma wyglądać wynik zapytania?
-Jakie elementy powinny zawierać wyniki; które z nich to teksty stałe i które z nich są elementami danych?
Dla każdego elementu szkicu powinniśmy wskazać, czy mamy do czynienia z in- formacją z systemu informatycznego czy elementu stałego, na przykład zdjęcie lub pól etykiety. W formie ekranowej zwykle nie pojawiają się pola etykiety z informacji w systemie informatycznym. Jeśli jednak system informatyczny obsługuje kilka języków,możliwe, że etykiety terenowe pochodzą z informacji w systemie informatycznym.

Identyfikuj zaangażowane klasy — jakich klas potrzebujemy?

Na podstawie naszkicowanego wyniku możemy określić potrzebne klasy z diagramu klas.

4.4.5

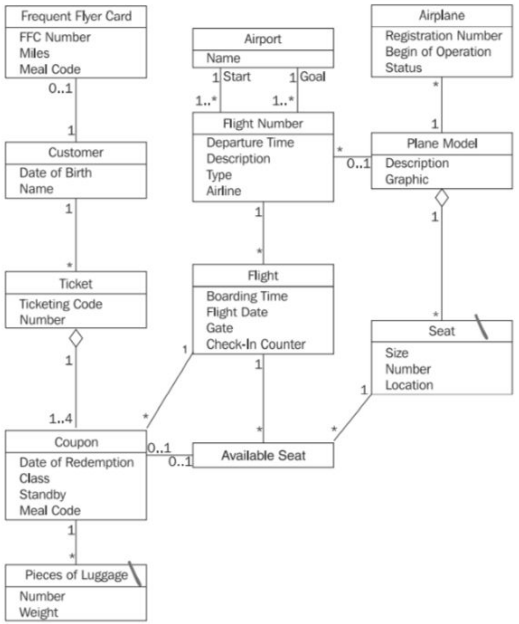
Pytania są następujące:

-Z jakich klas mają być odczytywane informacje? Musimy określić, z których atrybutów i klas każdy element informacji w naszkicowanym wyniku jest obli- czany.

-Jakie klasy są potrzebne do ścieżki dostępu? Zajęcia są albo potrzebne na pod- stawie ich atrybutów, które są niezbędne do wygenerowania wyników zapytania, lub na podstawie ich relacji z innymi klasami.

- Jakich klas brakuje na diagramie klas? W zależności od poziomu uzupełnienia diagramu klas, może się zdarzyć, że nowe, dodatkowe klasy muszą być modelo- wane.

W pierwszym szkicu potrzebne zajęcia można po prostu odręcznie zaznaczyć na wydruku zajęć diagram. Na rysunku klasy potrzebne do wygenerowania karty pokładowej są zaznaczone.



4.4.5

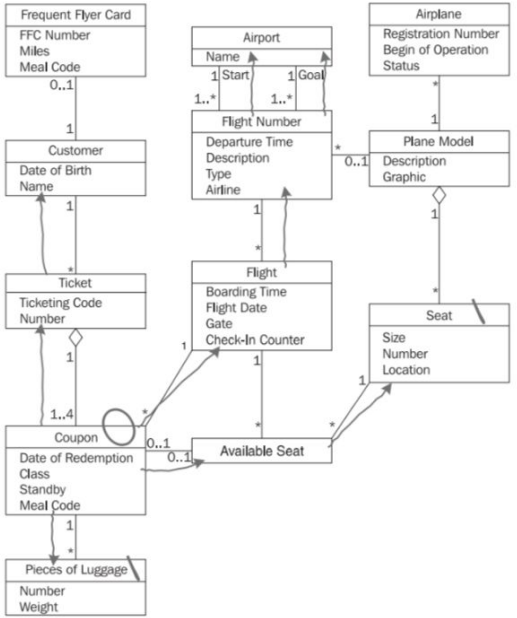
Zapytanie rozpoczyna się od obiektu początkowego. Początkowym obiektem może być klasa lub konkretny obiekt klasy. Jeśli określony obiekt jest adreso- wany bezpośrednio. Jeśli adresowana jest klasa, parametry mogą być dołączone do zapytania jako parametry wyboru, dzięki czemu można wybrać konkretny obiekt. Dalej, potrzebne klasy muszą być dostępne z początkowego obiektu po- przez połączenia. Konkretne pytania to:

-Jaki jest pierwszy obiekt, do którego ma trafić wydarzenie?
-Który obiekt już znam?
-Od czego zacząć zbierać informacje?

Aby wygenerować kartę pokładową w naszym studium przypadku, zaczynamy od kuponu samolotu bilet, dla którego chcemy wygenerować kartę pokładową.Zaprojektuj ścieżkę wydarzenia — dokąd zmierzamy?

Zaczynając od początkowego obiektu, określamy ścieżkę, przez którą potrzebne obiekty może być osiągnięty. Pytanie brzmi:
- Którą ścieżką mogę dotrzeć do wszystkich potrzebnych obiektów na diagramie klas?

W przypadku pierwszego szkicu tę ścieżkę można z kolei zaznaczyć odręcznie na wydruku zajęć diagram. Rysunek pokazuje całkowicie zaznaczony diagram klas dla zdarzenia zapytania wygenerowanie karty pokładowej:



4.4.5

Zmień ścieżkę wydarzenia — jakich dokładnie obiektów potrzebujemy?

Ścieżka zdarzenia musi być wypełniona selekcjami i iteracjami. Selekcje i iteracje są używane, gdy na ścieżce można dotrzeć do kilku obiektów jednej klasy z innej klasy, czyli gdy krotność asocjacji na diagramie klas ma górną granicę większą niż 1. W takich przypadkach musimy określić, czy wszystkie obiekty powinny być iterowane lub poszczególne obiekty powinny być wybierane na pod- stawie określonych kryteriów. Pytaniem jest:

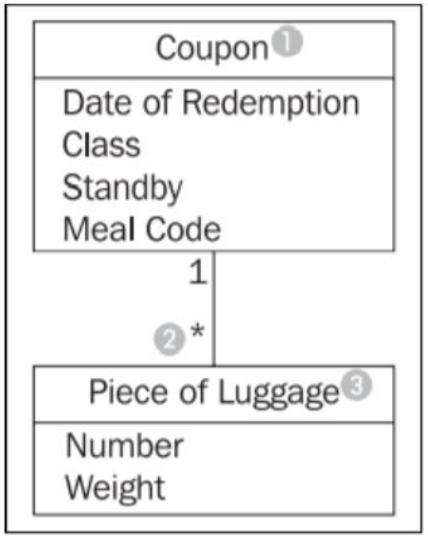
-Jeśli napotkam na swojej drodze więcej niż jeden obiekt, czy potrzebuję ich wszystkich?(iteracja), czy potrzebujesz konkretnego (selekcja)?

4.4.5

W naszym studium przypadku na rysunku możemy napotkać kilka sztuk bagażu wzdłuż ścieżki zdarzenia rozpoczynającej się na kuponie. Potrzebujemy wszystkich sztuk bagażu na kartę pokładową, ponieważ na karcie chcemy wy- drukować numer i wagę całkowitą. Dlatego musimy iterować.
Zidentyfikuj niezbędne atrybuty — czego dokładnie chcemy wiedzieć?
W ostatnim kroku dokumentujemy atrybuty, które są potrzebne do odpowiedzi na zapytanie.Pytania to:
-Jakie atrybuty są potrzebne do uzyskania wyniku zapytania?
-Jakich atrybutów brakuje na diagramie klas?

Wszystkie elementy danych z opracowanego wyniku zapytania muszą być w sta- nie prześledzić wstecz do atrybutów na diagramie klas widoku statycznego. W najprostszym przypadku może to być zrobione ponownie, zaznaczając atrybuty na wydruku diagramu klas. Kiedy zamodelowano schemat komunikacji, można wstawiać atrybuty.
Sprawdź diagram komunikacji — czy wszystko się zgadza?
- Czy projektowany wynik zapytania może być skonstruowany z komunikacją diagramu?
- Czy ścieżki zdarzeń w diagramie komunikacji biegną wzdłuż skojarzeń na sche- macie klas?

- Czy zawsze pokazuje, gdzie jest to potrzebne zgodnie z diagramem klas, jeśli mamy do czynienia z iteracją lub selekcją?
Jeśli odpowiedź na wszystkie te pytania brzmi „tak”, większość możliwych błę- dów zostały wyeliminowane.



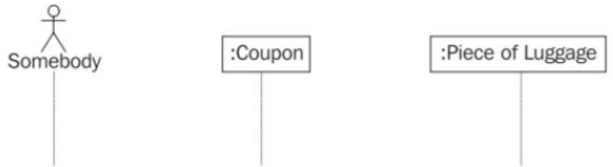
4.4.6 Konstruowanie diagramów sekwencji

Identyfikuj zaangażowane klasy — na co wpływa mutacja wydarze- nia?
Należy zidentyfikować klasy, na które wpływa zdarzenie mutacji. Dzieje się to na podstawie diagramów stanu. Pytania są następujące:

-Które klasy są już dotknięte określonym zdarzeniem mutacji? Aby odpowiedzieć to pytanie będziemy musieli sprawdzić, które diagramy stanów zawierają zdarzenie mutacji. Jeśli zdarzenie mutacji jest obecne na diagramie stanu a klasy, na tę klasę wpływa zdarzenie mutacji, co oznacza, że mutacja zdarzenia musi być wysłane do tej klasy.

-Na jakie inne klasy ma wpływ diagram mutacji? Może być tak, że istnieją klasy, na które ma wpływ zdarzenie mutacji, ale które jeszcze nie są obecne w diagramach stanów.

Na pierwsze pytanie można łatwo odpowiedzieć. Większość narzędzi CASE może generować na przykład listę klas już dotkniętych. Aby znaleźć kolejne klasy, których dotyczy problem, spójrzmy na diagram klas i zastanówmy się, czy coś musi się stać z obiektami każdej klasy, kiedy występuje zdarzenie mutacji. Powinniśmy jeszcze raz spojrzeć na te klasy, które znajdują się w pobliżu niektórych klas już dotkniętych na diagramie klas. Oczywiście, jeśli zostaną znalezione dodatkowe klasy, ich diagramy stanów muszą zostać zaktualizowane poprzez wstawienie zdarzeń mutacji. W naszym studium przypadku zdarzenie mutacji rejestruje bagaż, wpływa na klasy kuponu i bagaż, jak pokazano na rysunku obok.



4.4.6

Określ początkowy obiekt — gdzie znajduje się pierwsze zdarzenie mutacji?

Mutacja zaczyna się od początkowego obiektu. Początkowym obiektem może być klasa lub konkretny obiekt klasy. Jeśli określony obiekt jest adresowany bez- pośrednio, dla określonego przedmiotu lotu, to ten przedmiot musi być znany przed wysłaniem zdarzenia mutacji. Jeśli adresowana jest klasa, parametry mogą być dołączone do zdarzenia mutacji jako parametry selekcji, aby można było wybrać konkretny obiekt. Dalej, potrzebne klasy muszą być dostępne z początkowego obiektu poprzez połączenia. Konkretne pytania to:

Jaki jest pierwszy obiekt, do którego wydarzenie powinno się udać? -Który przedmiot już znam?

Od czego zacząć zbierać informacje?

W naszym studium przypadku podczas zdarzenia mutacji, kupon jest już znany, ponieważ cała odprawa następuje na konkretnym kuponie. Nowy przedmiot ba- gażowy, który ma zostać zarejestrowany, jeszcze nie istnieje. Dlatego inicjał obiekt jest obiektem kuponu.

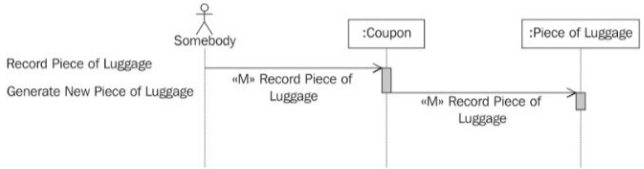
4.4.6

Propaguj zdarzenia — w jaki sposób przekazywane jest zdarzenie mu tacji?

Zaczynając od początkowego obiektu, ustalana jest sekwencja, w której do tknięte obiektyotrzymają zdarzenie mutacji. Pytanie brzmi:

W jakiej kolejności mogę dotrzeć do wszystkich dotkniętych obiektów na diagramie klas?

Zdarzenia mutacji są przekazywane wzdłuż relacji na diagramie klas do wszystkich klas na które ma wpływ zdarzenie mutacji. Na diagramie klas studium przypadku istnieje od kuponu do bagażu, wzdłuż której można wysłać zdarze- nie mutacji.



4.4.6

Określ parametr zdarzenia — co muszą wiedzieć obiekty?

Informacje wymagane do przetworzenia zdarzenia mutacji są przekazywane jako parametry zdarzenia. Pytanie brzmi:

Jakich informacji potrzebują obiekty, aby przetworzyć mutację?

Aby w naszym studium przypadku wygenerować nowy bagaż, potrzebujemy jego wagi. Dlatego waga zostanie przekazana jako parametr zdarzenia.Sprawdź diagram sekwencji — czy wszystko się zgadza?

Aby sprawdzić diagram sekwencji, musimy odpowiedzieć sobie na dwa pytania: Czy wszystkie klasy objęte zdarzeniem mutacji są wymienione na diagramie sekwencji?

Czy zdarzenie mutacji jest przekazywane zgodnie z asocjacją na diagramie klas? Jeśli odpowiedź na oba te pytania brzmi „tak”, to największe błędy zostały wykluczone.

Dziękujemy za uwagę

Skład zespołu:
Mariusz Budzioch, Karol Janik, Arkadziusz Jędruch, Michał Kazanecki, Krzysztof Kubień, Karol Maziarka, Piotr Skała, Mikołaj Wielebnowski