UML 2.0 w akcji. Przewodnik oparty na projektach

Autorzy: Patrick Graessle, Henriette Baumann, Philippe Baumann
Tłumaczenie: Marek Pętlicki
ISBN: 83-246-0646-7
Tytuł oryginału: Uml 2.0 in Action: A Project-based Tutorial
Format: B5, stron: 224

Poznaj język UML i wykorzystaj jego możliwości

- Opanuj podstawy języka
- Stwórz modele systemów biznesowych i informatycznych
- Zaplanuj integrację systemów przy użyciu języka UML

Kluczowym elementem dużych projektów programistycznych jest modelowanie, pomocne również przy tworzeniu średnich i małych projektów. Język UML to narzędzie służące właśnie do modelowania. Za pomocą diagramów i wykresów będących jego elementami można przedstawić zarówno sam system informatyczny, jak i jego związki z otoczeniem biznesowym, w którym będzie on wykorzystywany. Gdy system tworzony jest przez duży zespół projektowy, zastosowanie języka UML daje gwarancję poprawnego zinterpretowania zarówno założeń, jak i zadań systemu, a także zgodnej z nimi realizacji projektu.

Książka „UML 2.0 w akcji. Przewodnik oparty na projektach" prezentuje język UML w sposób gwarantujący jego błyskawiczne opanowanie. Jej autorzy koncentrują się wyłącznie na tych zastosowaniach UML-a, które mogą okazać się przydatne w modelowaniu. Wszystkie zagadnienia teoretyczne podane są wraz z konkretnymi przykładami ich przydatności praktycznej. Dzięki tej publikacji poznasz elementy języka i nauczysz się wykorzystywać je podczas projektowania. Dowiesz się, jak tworzyć projekty systemów biznesowych i informatycznych oraz jak stosować UML w opisach zależności pomiędzy nimi.

Oto kwestie poruszane w tej książce:

- przegląd elementów i zastosowań języka UML 2.0
- modelowanie procesów biznesowych
- diagramy przypadków użycia
- diagramy aktywności i sekwencji
- diagramy klas i stanów
- modelowanie systemu informatycznego i jego otoczenia

Przekonaj się, jak język UML usprawnia pracę nad projektami

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

ROZDZIAŁ 2

Podstawy i tło historyczne

Wiele zagadnień omawianych w kolejnych rozdziałach opiera się na pewnych podstawach, które wprowadzimy w tym rozdziale.

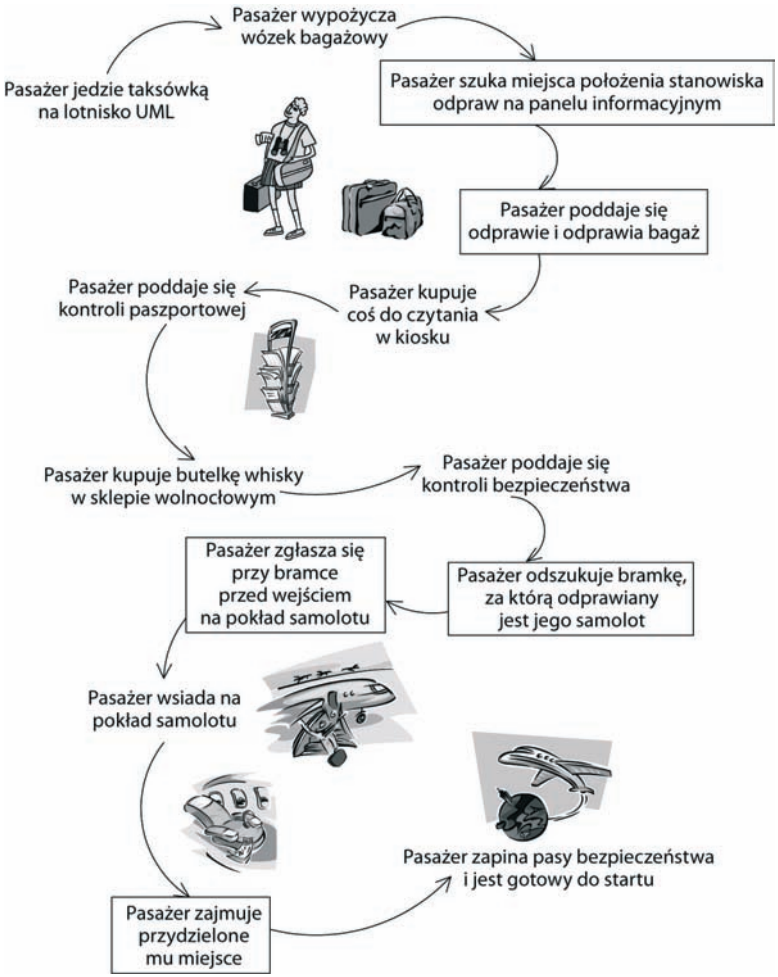
2.1. Wprowadzenie do studium przypadku

Do naszych analiz wykorzystamy lotnisko — nazwiemy je „lotniskiem UML". Dzięki temu każdy Czytelnik, który miał okazję lecieć samolotem, bez problemu zrozumie wszystkie przykłady.

Rozważania ograniczymy do tych obszarów pracy lotniska, z którymi podróżny może zetknąć się na co dzień, zatem największą uwagę poświęcimy procedurom **odprawy pasażera** i jego **wejścia na pokład** samolotu. Rysunek 2.1 ilustruje, w jaki sposób da się wyróżnić usługi związane z pasażerami spośród innych związanych z pracą lotniska. Rysunek ten przedstawia schemat operacji, w których musi wziąć udział pasażer, zanim znajdzie się na swoim fotelu w samolocie i zapnie pasy, a samolot będzie gotów do startu. Nie wszystkie etapy działań pasażera wiążą się jednak z interesującymi nas obszarami pracy lotniska. Te, które nas interesują, zostały na rysunku ujęte w ramkę.

Tego typu sekwencja nosi nazwę **scenariusza** (ang. *scenario*). Scenariusz przedstawiony na rysunku jest tylko jednym z możliwych, ponieważ mogą wystąpić liczne sytuacje odbiegające od tego schematu, na przykład takie:

- pasażer posiada jedynie bagaż podręczny;
- pasażer nie dokonuje zakupów w kiosku z prasą;
- pasażer przybył na lotnisko za późno i jest zmuszony poddać się przyspieszonej odprawie;



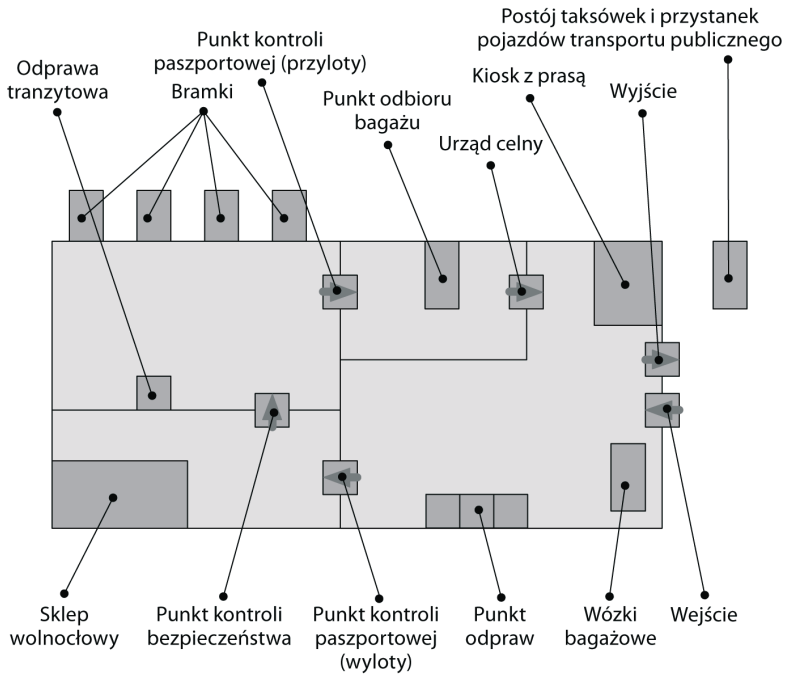
Rysunek 2.1. Studium przypadku — pasażer wybiera się w podróż samolotem

- pasażer zgubił swój bilet;
- pasażer przyleciał innym samolotem i musi po prostu dokonać przesiadki, czyli pozostaje w obszarze tranzytowym;
- pasażer poddał się odprawie, ale zasypia w poczekalni i pomimo wielokrotnego wywoływania przez megafon nie zgłasza się, więc samolot odlatuje bez niego;
- pasażer zostaje zatrzymany przy odprawie paszportowej, ponieważ jego paszport jest nieważny.

Warto zastanowić się, które z powyższych przypadków mają rzeczywiste znaczenie podczas procedury odprawy pasażerów oraz czy można sobie wyobrazić inne sytuacje, ważniejsze od wymienionych.

Schemat hipotetycznego lotniska przedstawiony na rysunku 2.2 powinien być pomocny przy wyobrażaniu sobie zdarzenia z naszego studium przypadku. Z procedurą odprawy łączy się także wiele innych obszarów pracy lotniska, sąsiadujących z tym dotyczącym samego pasażera. Wśród nich można wskazać działanie:

- kas biletowych;
- kiosku z prasą;
- sklepu wolnocłowego;
- stanowiska odprawy paszportowej;
- kontroli lotu;
- stanowiska informacyjnego;
- stanowiska obsługi i transportu bagażu.



Rysunek 2.2. Schemat lotniska UML

Procedura odprawy pasażerów wymaga wymiany danych z tymi obszarami. Konieczna jest też komunikacja z innymi sferami pracy lotniska. Będą one omawiane przy okazji definiowania modelu biznesowego oraz integracji systemu. Studium przypadku będzie rozbudowywane w kolejnych rozdziałach książki przy okazji wprowadzania kolejnych zagadnień.

Lotnisko UML jest niewielkim portem lotniczym, więc i nasze studium przypadku nie będzie nazbyt skomplikowane.

Celem tego studium przypadku jest dostarczenie spójnego przykładu zastosowania UML-a. W każdym z kolejnych rozdziałów model będzie rozszerzany o kolejne kwestie. Na początek należy jednak wyjaśnić kilka szczegółów:

- Bilet lotniczy składa się z samego biletu oraz z kilku dodatków (do czterech). **Bilet** ma postać książeczki, w której każdemu etapowi podróży jest poświęcona osobna sekcja. Na przykład może on zawierać jeden kupon na lot z Zurychu do Frankfurtu, drugi na lot z Frankfurtu do Londynu, a trzeci na powrót z Londynu do Zurychu. Podczas każdej odprawy odpowiedni kupon zostaje zamieniony na odpowiednią kartę pokładową. Sam bilet natomiast pozostaje w posiadaniu pasażera.
- Należy rozróżnić pojęcia lot i numer lotu. **Numer lotu** może na przykład mieć postać LH435 lub LX016. Numer ten określa regularny przelot z lotniska początkowego do lotniska przeznaczenia. **Lot** natomiast określa się przy użyciu numeru lotu i daty, na przykład LH435 26 czerwca 2006 roku. Można zatem powiedzieć, że lot to rzeczywista realizacja numeru lotu. Lot może być również odwołany z powodu niekorzystnych warunków pogodowych. Numer lotu jest stosowany przez cały okres regularnego wykonywania przez linię lotniczą przelotów na określonej trasie.
- Należy rozróżnić także trzy formy odprawy:
  - **zwykła odprawa** z bagażem w punkcie odpraw;
  - **ekspresowa odprawa** bez bagaży w specjalnym punkcie odpraw;
  - **automatyczna odprawa** bez bagaży.

2.2. Modele, perspektywy i diagramy

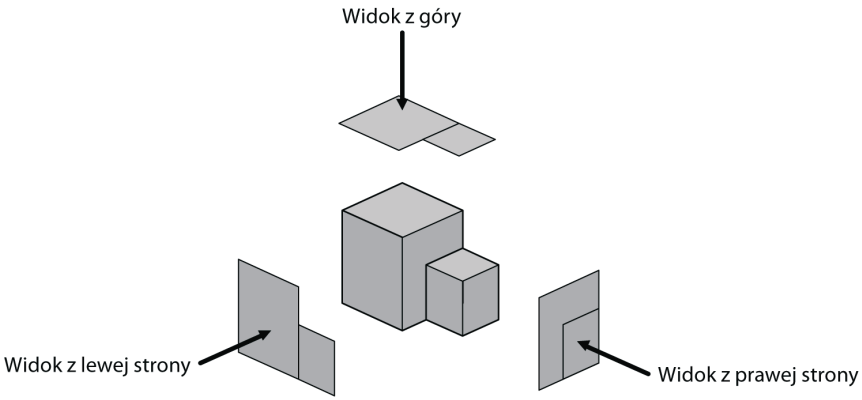
2.2.1. Czym jest model?

Modele są budowane w kontekście biznesowym lub informatycznym w celu lepszego zobrazowania istniejących lub przyszłych systemów. Należy pamiętać, że model nigdy nie będzie w pełni odpowiadał rzeczywistości. Modelowanie jest nieodłącznie związane z **podkreślaniem** i **pomijaniem**: podkreślane są szczegóły istotne, pomijane zaś szczegóły zbędne. Jak jednak można odróżnić jedne od drugich? Nie ma, niestety, uniwersalnej odpowiedzi na to pytanie. Odpowiedź zależy od tego, **po co** model jest tworzony i **kto** ma być jego odbiorcą.

Zastanówmy się, jakie szczegóły są podkreślane, a jakie pomijane w następujących modelach:

- model samochodu do badań aerodynamicznych
- model budynku w skali 1:50
- plan trasy metra
- mapa
- schemat organizacyjny

Im więcej informacji wprowadzi się do modelu, tym bardziej stanie się on skomplikowany, a przez to trudniejszy do zrozumienia. Na przykład mapa Europy zawierająca informacje polityczne, geologiczne, demograficzne i związane z transportem będzie z pewnością bardzo trudna do odczytania. Najprostszym rozwiązaniem tego problemu będzie z pewnością poświęcenie poszczególnym zagadnieniom osobnych, wyspecjalizowanych map. Z tego powodu analizowany obiekt jest często ujmowany w różnych **perspektywach**. Perspektywy te są ze sobą powiązane na wiele sposobów. Jeśli jedna z nich ulega zmianie, pozostałe również należy odpowiednio zmodyfikować. Jeśli na przykład Holandia powiększy swój obszar o kolejny fragment lądu odebrany Morzu Północnemu, należy odpowiednio zmodyfikować wszystkie wyspecjalizowane mapy tego obszaru.



Rysunek 2.3. Różne perspektywy postrzegania tego samego obiektu

Podobne zasady dotyczą modelu budynku. Jeśli do istniejącego budynku zostanie dodane nowe skrzydło, należy odpowiednio zmodyfikować poszczególne ujęcia, w tym rzuty pięter, poszczególne rzuty perspektywiczne oraz trójwymiarowy, drewniany model. Zasadę tę w sposób schematyczny ilustruje rysunek 2.3. W podrozdziale 2.4 („Modele studium przypadku") bliżej zajmiemy się powiązaniami między modelami systemu analizowanego w tej książce. Poszczególne perspektywy w każdym z modeli zostaną omówione w rozdziałach 3., 4. i 5.

2.2.2. Do czego potrzebne są modele?

Model systemu realizuje następujące funkcje:

- **Komunikacja** między elementami systemu. Aby system działał poprawnie, wszystkie jego elementy muszą być skonstruowane zgodnie z pewnym zasadniczym, jednolitym założeniem. Szczególnie ważne jest to, aby każdy użytkownik systemu rozumiał zastosowaną terminologię, żeby klienci zgadzali się z przyjętymi założeniami, a twórcy systemu rozumieli te założenia w taki sam sposób. Równie istotna jest kwestia tego, by decyzje były później tak samo czytelne, jak w momencie ich podejmowania.

- **Wizualizacja** wszystkich procedur z punktu widzenia klientów, ekspertów i użytkowników. Wszystkie skumulowane procedury związane z systemem powinny być zaprezentowane w taki sposób, aby każdy zainteresowany mógł je zrozumieć. Wiemy jednak z doświadczenia, że często podczas prezentacji projektu w postaci diagramu (zamiast opisu słownego) odbiorca odruchowo reaguje negatywnie. Ważne jest zatem pokonanie tej reakcji. Jest ona bowiem zwykle spowodowana strachem przed nieznanym, a diagramy na pierwszy rzut oka mogą sprawiać wrażenie czegoś skomplikowanego. Ta książka zawiera wskazówki dotyczące również sposobu analizy diagramów.
- **Weryfikacja** faktów pod kątem kompletności, spójności i poprawności. Szczególnie jasna definicja wzajemnych związków umożliwia zadawanie pytań i udzielanie odpowiedzi. Przy każdym prezentowanym diagramie będziemy przedstawiać stosowną listę pytań.

Proponujemy Czytelnikowi, aby odpowiedział sobie na następujące pytania:

- Kiedy ostatnio podczas dyskusji nad systemem okazało się, że cele klienta i dostawcy są zupełnie sprzeczne?
- Kiedy ostatnio miałeś wrażenie, że po raz kolejny omawiasz ten sam temat?
- Kiedy ostatnio żałowałeś, że podczas ustalania ważnego konsensusu w dyskusji nie miałeś włączonego dyktafonu?

2.2.3. Cel i grupa docelowa modelu

W życiu codziennym często okazuje się, że wyniki ciężkiej pracy nad projektem zwyczajnie grzęzną na czyimś biurku pod zwałami papieru. Można zapytać, dlaczego tak się dzieje. Efekt końcowy prac nad modelem jest uzależniony od dwóch podstawowych czynników: odbiorcy modelu (**dla kogo?**) i rzeczywistego jego przeznaczenia (**po co?**). Jeśli obydwa te czynniki nie zostaną ustalone w sposób jednoznaczny, istnieje realne ryzyko, że w wyniku prac powstaną modele niezawierające informacji ważnych dla odbiorcy. A jeśli model nie podkreśla tego, co istotne, i nie pomija tego, co zbędne, jego odbiorca z pewnością uzna, że jest on bezużyteczny.

Aby zdefiniować grupę docelową modelu, należy odpowiedzieć na następujące pytania:

- Jakiego poziomu **zaawansowania biznesowego** należy odczekiwać od odbiorców? Czy należy założyć, że mają oni podstawową wiedzę na temat obiektu, czy też model ma zawierać fundamentalne szczegóły dotyczące modelowanych zdarzeń i procesów?
- Jaki poziom **szczegółowości** jest potrzebny odbiorcom? Jaki poziom komplikacji jest dopuszczalny w modelu? Jeśli procesy i systemy będą podatne na ciągłe zmiany, bardzo szczegółowy model może być mało realistyczny. Dzieje się tak z tego powodu, że w większości przypadków nie ma możliwości stałego utrzymania takiego modelu w aktualnej postaci. Mniej szczegółowy model wymaga mniejszych nakładów pracy, jeśli chodzi o jego utrzymanie, jest jednak oczywiście mniej dokładny.

- Ile **czasu** grupa docelowa może poświęcić na analizę i interpretację modelu? Aby uniknąć przysypania modelu stosem papierów, należy zadbać o odpowiedni poziom szczegółowości i komplikacji — w przeciwnym wypadku nikt nie będzie zaprzątał sobie nim głowy.
- Jakim **językiem** posłużyć się przy definicji modelu? Czy grupa docelowa zrozumie biznesową terminologię techniczną? Czy zrozumie terminologię informatyczną? Oto prosty przykład: jeśli na butelce z wodą widnieje etykieta z napisem „woda", mamy pewność, że prawie każdy, kto ją przeczyta, domyśli się, jaka jest zawartość naczynia. Jeśli jednak na etykiecie napisać „H_2O", to nawet pomimo technicznej poprawności tego opisu mamy pewność, że przekaz dotrze do węższej grupy odbiorców, na przykład pracowników laboratorium chemicznego. Takie rozwiązanie ma jednak swoje zalety: odbiorca uzyskuje dodatkową, precyzyjną informację na temat składu chemicznego substancji. Z tego przykładu wynika wniosek, że „etykiety" (czyli nomenklaturę) należy zawsze uważnie dobierać pod kątem założonej grupy odbiorców przekazu.
- Jaki zastosować poziom **abstrakcji**? Im mniej abstrakcyjny jest model, tym bardziej czytelny i zrozumiały będzie dla odbiorcy. Dzieje się tak z tego powodu, że niższy poziom abstrakcji jest bliższy percepcji użytkownika i stosowanemu przez niego językowi. Z drugiej strony modele o wysokim poziomie abstrakcji są bardziej uniwersalne i łatwiej je przełożyć na system informatyczny. Takie modele również łatwiej analizować pod kątem poprawności formalnej. Specjaliści do spraw informatyki z reguły preferują modele abstrakcyjne. Użytkownicy natomiast często zupełnie gubią się w zetknięciu z modelami tego typu.

Wskazówki praktyczne

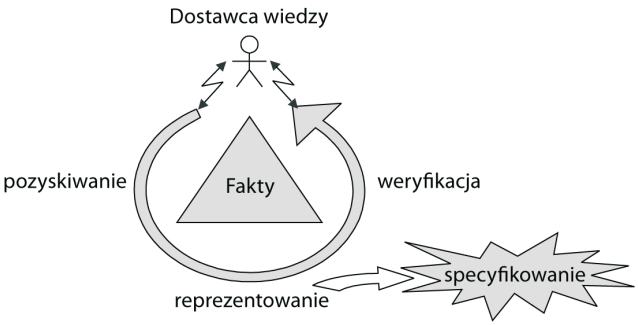
Przy wyborze poziomu abstrakcji, czytelności i szczegółowości modelu często jesteśmy zmuszeni do **kompromisów**. Można oczywiście stworzyć kilka modeli różniących się poziomem formalności i szczegółowości, optymalizując je dla różnych grup odbiorców. W ten sposób komunikacja między twórcami modelu, klientami, użytkownikami i deweloperami będzie odbywać się sprawniej. Należy jednak unikać przesady — model powinien być dostosowany do grupy odbiorców i założonego wykorzystania.

Analiza i **wzorce projektowe** są przykładami modeli opisujących powszechnie stosowane techniki projektowania i modelowania. Należy, o ile to możliwe, poszukiwać tego typu modeli w internecie, książkach[1], prasie fachowej. Można też po prostu konsultować się ze współpracownikami.

[1] Jako przykład można tu wymienić książkę Martina Fowlera, *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1999[0].

2.2.4. Proces analizy

Rysunek 2.4 przedstawia proces analizy składający się z faz **pozyskiwania**, **reprezentowania** i **weryfikowania** faktów.



Rysunek 2.4. Proces analizy

Ten etap jest realizowany przez analityka. W procesie analizy powstaje specyfikacja wynikająca z modelu i innych dostępnych dokumentów. Analityk współpracuje z osobami posiadającymi wiedzę na temat modelowanego systemu: klientami, użytkownikami i ekspertami w analizowanej dziedzinie.

- Fakty są **pozyskiwane** w wyniku współpracy pomiędzy analitykiem a ekspertami. Ta współpraca polega na tym, że eksperci są dostawcami wiedzy w danej dziedzinie, a analitycy są dostawcami wiedzy metodologicznej.
- Fakty są **reprezentowane** w postaci diagramów i dokumentów przygotowywanych z reguły przez analityka.
- Fakty są **weryfikowane** wyłącznie przez dostawców wiedzy w danej dziedzinie, ponieważ tylko oni mogą zdecydować, czy są one poprawne. Weryfikacja faktów jest etapem absolutnie koniecznym. Bez niego twórcy systemu otrzymają piękne diagramy, lecz będzie istniało ryzyko, że prezentowane informacje są nieprawdziwe. Innymi słowy, praca nad modelem bez weryfikacji jego poprawności jest zupełnie bezużyteczna!

Porady praktyczne

Niemożliwe jest stworzenie i zweryfikowanie użytecznego modelu bez opanowania technicznych podstaw analizowanego problemu. Gdzie szukać **dostawców wiedzy**, którzy będą służyć informacjami na temat modelowanego systemu? Z naszych doświadczeń wynika, że szczególnie dobre wyniki można uzyskać, korzystając ze współpracy z grupami osób takich, jak:

- uczestnicy i kontrolerzy procesów biznesowych;
- użytkownicy systemów informatycznych o funkcjonalności zbliżonej do modelowanego systemu lub związanej z nim;

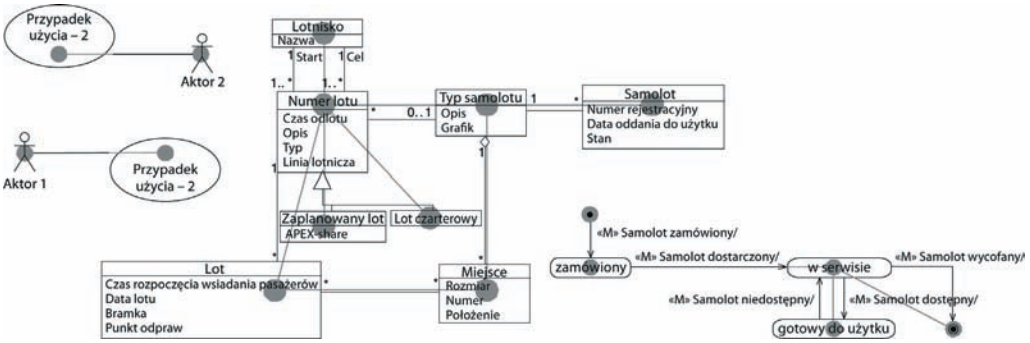
- klienci, którzy często bywają krytycznymi, lecz również kreatywnymi dostawcami wiedzy;
- partnerzy biznesowi;
- eksperci w analizowanej dziedzinie;
- zarząd firmy;
- niezależni obserwatorzy.

W procesie analizy i poznawania procesów biznesowych pomocne bywają następujące techniki:

- obserwacja pracowników przy pracy;
- branie udziału w analizowanych procesach biznesowych;
- przyjęcie roli uczestnika zewnętrznego (na przykład klienta);
- ankiety;
- przeprowadzanie wywiadów;
- organizowanie burz mózgów z udziałem wszystkich zaangażowanych grup;
- prowadzenie dyskusji z ekspertami;
- analiza istniejących formularzy, dokumentacji, specyfikacji i narzędzi pracy;
- opisywanie struktury organizacyjnej i zasad przepływu informacji (diagramy organizacyjne itp.).

2.2.5. Diagramy w roli perspektyw

Każdy diagram UML odgrywa rolę jednej **perspektywy** modelu systemu. W zależności od typu diagramu na pewne zagadnienia kładzie się nacisk, inne zaś są pomijane. Większość diagramów występuje w postaci graficznej (przykład na rysunku 2.5), co znaczy, że składają się one z elementów graficznych połączonych liniami:

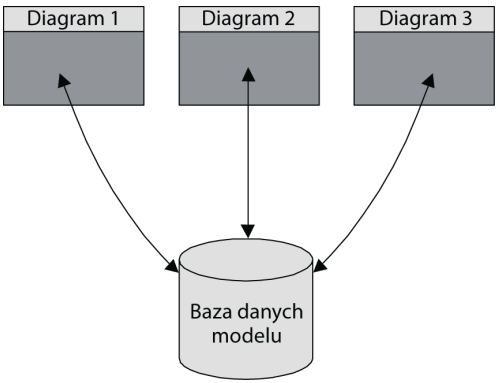


Rysunek 2.5. Diagram graficzny

W celu odczytu diagramów należy zapoznać się ze znaczeniem elementów i linii. Poszczególne rodzaje diagramów oraz występujące w nich symbole będziemy omawiać stopniowo w kolejnych rozdziałach książki.

Nawet w przypadku **komputerowych narzędzi wspomagania inżynierii oprogramowania** (ang. *Computer Aided Software Engineering* — CASE) diagramy UML są traktowane jako perspektywy. Narzędzia te do przechowywania informacji o modelu wykorzystują bazy danych. Każdy diagram (w roli perspektywy) zawiera część informacji na temat całości. Dzięki temu narzędzia CASE pomagają zachować spójność każdej perspektywy. Jeśli na przykład nazwa klasy ulega zmianie w diagramie klasy, diagram stanów wykorzystujący tę klasę również ulega zmianie.

Baza danych zawierająca elementy modelu to główny element odróżniający narzędzie CASE od zwykłego programu graficznego (rysunek 2.6). Każdy diagram UML można narysować za pomocą ołówka i kartki papieru lub dowolnego programu graficznego. W takim przypadku jednak każdy diagram będzie po prostu rysunkiem. Dopiero zastosowanie narzędzia CASE wyposażonego w bazę danych i zgodnego ze specyfikacją UML pozwala na stworzenie kolekcji danych, zarządzanie nią i modyfikowanie informacji bez obawy o naruszenie spójności tej kolekcji.



Rysunek 2.6. Narzędzie CASE jako baza danych

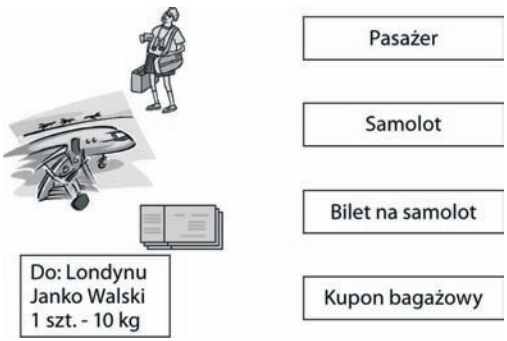
UML zawiera własny model bazy danych — jest to metamodel UML, element składowy specyfikacji UML (*OMG: Unified Modeling Language: Infrastructure, Version 2.0, Final Adopted Specification* — wrzesień 2003 i *OMG: Unified Modeling Language: Superstructure, Version 2.0, Revised Final Adopted Specification* — wrzesień 2004; obydwa dokumenty można znaleźć w serwisie *http://www.omg.org*). Wszystkie elementy występujące w diagramach UML oraz w opisach elementów są wyszczególnione w metamodelu UML. Można w nim znaleźć na przykład informację o tym, że klasa może zawierać atrybuty i metody. Ten model danych języka UML jest fundamentem baz danych narzędzi modelujących, czyli na przykład wszystkich narzędzi CASE. Niestety, większość narzędzi CASE jest zasobożerna, kosztowna, niedopracowana, kłopotliwa w obsłudze i wymagająca czasochłonnego szkolenia. Mimo tych wad w większości przypadków (z wyjątkiem naprawdę małych projektów) ich wykorzystanie daje niepodważalne korzyści.

2.3. Systemy informacyjne a systemy informatyczne

Praktycznie w każdym zawodzie przynajmniej część obowiązków jest związana z obróbką informacji. Tak dzieje się od tysiącleci i to było jedną z przyczyn powstania pisma. Jeden z najstarszych zapisków w dziejach Europy dotyczył listy magazynowej zaopatrzenia pałacu w Knossos na Krecie. Gdybyśmy byli w stanie poznać procedury działania pracowników zaopatrzenia sprzed 3500 lat, moglibyśmy odwzorować procesy biznesowe ich działań. Moglibyśmy na przykład zamodelować kontakty z dostawcami i odbiorcami w procesie wymiany towarów oraz przeanalizować zapisy ich działalności biznesowej. To samo dotyczy handlarzy oliwą w starożytnym Rzymie 1500 lat później, kupców Hanzy w piętnastowiecznych północnych Niemczech czy też firmy Lloyd's w Londynie na początku dwudziestego wieku.

W każdym z powyższych przykładów do zarządzania zadaniami wykorzystywane były mniej lub bardziej skomplikowane systemy informacyjne. Celem tych systemów było (i nadal jest) zarządzanie informacjami niezbędnymi do prowadzenia interesów. Oczywiście, wszystko odbywało się bez udziału komputerów. Systemy informacyjne były oparte na innych technikach, takich jak kreda i tablica czy też kartoteki. Obecnie komputery pozwalają na implementowanie systemów informacyjnych w postaci systemów informatycznych. To daje nowe możliwości, niewyobrażalne dla handlarza oliwą w starożytnym Rzymie. Główny cel tych systemów pozostał jednak ten sam — jest nim udostępnianie danych niezbędnych w codziennych procesach biznesowych. W tej książce będziemy zajmować się przede wszystkim systemami informatycznymi, ponieważ przyjmujemy założenie, że systemy informacyjne modelowane za pomocą techniki UML są implementowane z użyciem technologii informatycznych.

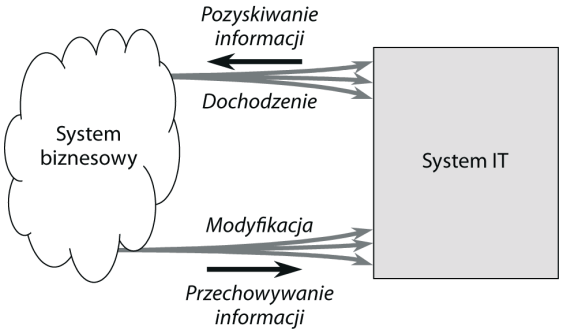
W naszej analizie systemu **obsługi pasażerów** na lotnisku zajmujemy się pasażerami, biletami lotniczymi i prawdziwymi odlotami. W systemie informacyjnym mamy jednak do czynienia z reprezentacjami (**odzwierciedleniami**) pasażerów, biletów i odlotów. Te reprezentacje składają się z **informacji** na temat pasażerów, biletów i odlotów zapisanych w systemie informatycznym. Są to informacje niezbędne do obsługi procesów biznesowych lotniska. Sytuację tę przedstawia rysunek 2.7.



Rysunek 2.7. Obiekty świata rzeczywistego i ich reprezentacje

System informatyczny działa z użyciem komputerów — jest to system, który dostarcza informacji niezbędnych do prowadzenia interesów. Informacje te są najczęściej dostarczane w rezultacie zapytania zadanego przez użytkownika. Oczywiście, aby odpowiadać na zapytania, system informatyczny musi być zasilony informacjami.

Rysunek 2.8 przedstawia schemat współpracy między systemami biznesowymi i informatycznymi. W zakresie procesów biznesowych systemu biznesowego informacja jest odczytywana i zapisywana w systemie informatycznym.



Rysunek 2.8. System informatyczny

Techniki modelowania omawiane w tej książce przydadzą się nie tylko na etapie tworzenia systemu informatycznego — będą również pomocne na etapie analizy systemów informacyjnych. Aby to zademonstrować, posłużymy się drugim przykładem (obok przeważającego w książce przykładu lotniska UML). Do tego drugiego przykładu będziemy wracać w różnych miejscach książki.

Tym razem proszę wyobrazić sobie biuro średniowiecznego kupca należącego do Hanzy. Kupiec ten nazywa się Hafenstein. Hanza była potężnym, działającym w okresie średniowiecza stowarzyszeniem zrzeszającym kupców w Niemczech północnych i w innych krajach bałtyckich.

Szefem biura jest skrupulatny sekretarz Hildebrand. W biurze przechowywane są różne księgi, między innymi księga rozliczeń dziennych, bilans sprzedaży oraz spis klientów. Każdą księgą opiekuje się inny księgowy. Nikt oprócz uprawnionego księgowego nie może wprowadzać zmian w księdze i tylko on wie dokładnie, w którym miejscu można znaleźć konkretną informację.

W naszej terminologii biuro, Hildebrandt, księgowi i księgi składają się na system informacyjny. Dzięki temu przykładowi chcemy w różnych miejscach książki zademonstrować, że chociaż system informacyjny jest dziś zwykle zaimplementowany w postaci systemu informatycznego, w rzeczywistości koncepcyjnie niewiele ma wspólnego z komputerami. Można go fizycznie zaimplementować na wiele różnych sposobów.

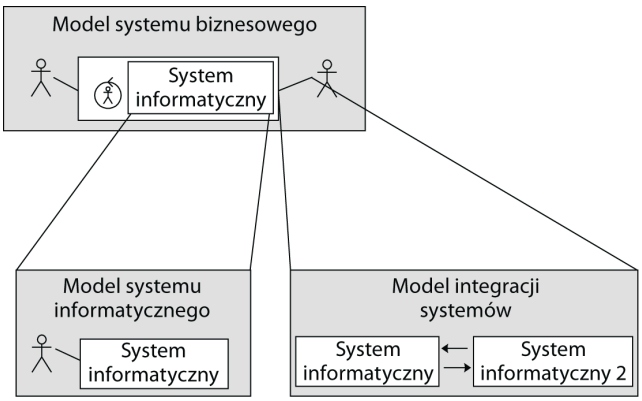
2.4. Modele studium przypadku

W naszym studium przypadku skonstruujemy trzy modele różnych systemów:

1. Model systemu biznesowego obsługi pasażerów, czyli zagadnienia biznesowe związane z systemem informatycznym. Model ten będzie obrazował procesy biznesowe, pasażerów, partnerów biznesowych, pracowników itp. Omówimy go w rozdziale 3.
2. Model systemu informatycznego obsługi pasażerów. Będzie on oparty o model systemu biznesowego obsługi pasażerów. Omówimy go w rozdziale 4.
3. Model integracji opisujący integrację systemów z otoczeniem, w szczególności interfejsy łączące system ze światem zewnętrznym. Również w tym przypadku fundamentem będzie model systemu biznesowego obsługi pasażerów. Omówimy go w rozdziale 5.

Aby zbudować i zintegrować system informatyczny, niezbędne będą wszystkie trzy wymienione wyżej modele: sam model systemu informatycznego nie wystarczy. Taka zależność zachodzi nie tylko w naszym studium przypadku, lecz również we wszystkich innych systemach.

Jak widać na rysunku 2.9, model systemu biznesowego stanowi podbudowę dla dwóch pozostałych modeli. Z tego powodu stanowi punkt wyjścia większości działań w ramach projektu. Zastosowanie **ujednoliconego języka modelowania** ma zatem ogromne zalety, ponieważ dzięki niemu jeden model będzie zrozumiały i użyteczny zarówno dla pracowników różnych działów biznesowych, jak i dla informatyków. Takie podejście umożliwia płynną wymianę modeli pomiędzy różnymi obszarami prac nad projektem. Również weryfikacja poprawności modeli jest dzięki temu znacznie uproszczona. Jesteśmy przekonani, że UML służy jako łącze między wymogami technicznymi a sferą funkcjonalną systemów informatycznych.



Rysunek 2.9. Modele studium przypadku

2.5. Historia UML-a — metody i notacje

Technologia informacyjna w swojej krótkiej historii zdążyła się już dorobić bardzo wielu **metod** i **notacji**. Istnieją metody i notacje dla projektu, struktury, przetwarzania i przechowywania informacji. Istnieją również metody planowania, modelowania, implementacji, składania, testowania, dokumentacji, dostosowywania itp. Niektóre ze stosowanych koncepcji są dość podstawowe i z tego powodu można je postrzegać jako wykraczające poza zakres technologii informacyjnej. Jednym z przykładów takich cech jest dziedziczenie, zjawisko znane ze świata rzeczywistego, będące jednocześnie podstawowym założeniem programowania zorientowanego obiektowo.

Mniej więcej do lat siedemdziesiątych twórcy oprogramowania traktowali proces programowania jako działalność niemal artystyczną. Jednakże systemy stawały się coraz bardziej skomplikowane, co spowodowało, że tworzenie oprogramowania i jego rozwój ze względów praktycznych nie mogły już być traktowane jako zadanie dla kreatywnych indywidualistów. Taka indywidualistyczna postawa doprowadziła stopniowo do **kryzysu branży**.

Kryzys doprowadził do wypracowania **metodologii inżynierskich** (inżynierii oprogramowania) i technik programowania strukturalnego. Zostały opracowane metody strukturalizacji systemów oraz procesu ich projektowania, tworzenia i rozwijania. Podejścia procesowe, na przykład metoda **HIPO** (ang. **Hierarchy Input Processing Output**), kładły główny nacisk na funkcjonalność systemów. Dzięki tej metodzie system jest dzielony na mniejsze elementy za pomocą rozkładu funkcjonalnego.

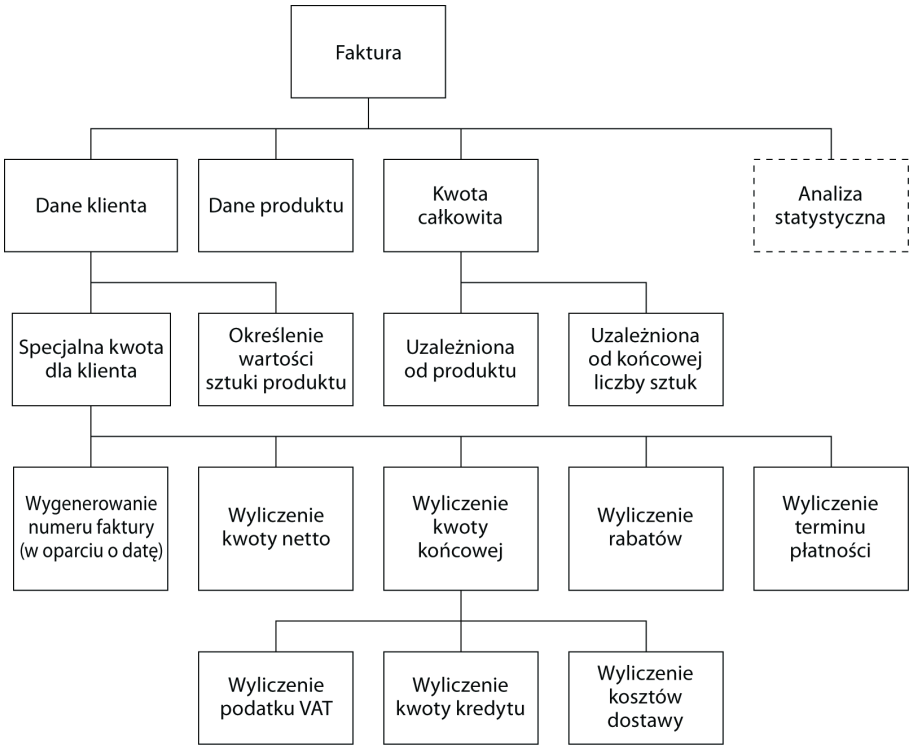
Rysunek 2.10 przedstawia graficzną reprezentację (diagram hierarchiczny) podfunkcji systemu wystawiania faktur. Każdy element jest opisany za pomocą schematu: *dane wejściowe-przetwarzanie-dane wyjściowe*.

Mniej więcej w tym samym okresie opracowano metodologie zorientowane na struktury danych. Jedną z nich jest metoda Jacksona, gdzie struktura programu jest oparta na graficznej reprezentacji struktur danych.

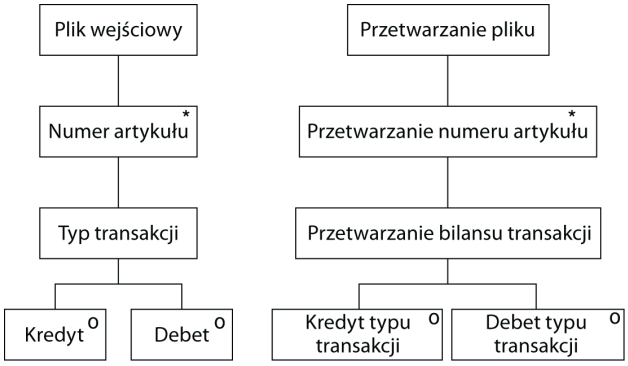
Na rysunku 2.11 w lewej kolumnie została przedstawiona struktura danych listy magazynowej. W prawej kolumnie przedstawiono zaś strukturę programu wypracowaną w oparciu o strukturę danych.

We wszystkich tych metodach i notacjach systemy są podzielone na dwie części — sekcję danych i sekcję procedur. Ten podział jest bardzo wyraźny w starszych językach programowania, takich jak COBOL. Diagramy przepływowe, strukturalne, HIPO i Jacksona są wykorzystywane do reprezentowania różnych funkcji. Oczywiście, te wczesne metody były jedynie podstawą do opracowania nowych systemów.

W latach osiemdziesiątych znacznie rozwinięto klasyczną analizę strukturalną. Twórcy oprogramowania uzyskali nowe narzędzia w postaci diagramów związków encji do modelowania danych oraz sieci Petriego do modelowania procesów.



Rysunek 2.10. Diagram HIPO



Rysunek 2.11. Diagram Jacksona

Wraz ze wzrostem poziomu komplikacji systemów projektowanie od zera stawało się metodą coraz mniej uzasadnioną ekonomicznie. Na znaczeniu zyskały takie cechy, jak łatwość utrzymania i możliwość ponownego użycia (ang. *reusability*). Opracowano języki programowania zorientowane obiektowo, a wraz z nimi pierwsze języki modelowania zorientowanego obiektowo (w latach siedemdziesiątych i osiemdziesiątych). W latach dziewięćdziesiątych pojawiły się

pierwsze publikacje na temat analizy obiektowej i projektowania obiektowego. W połowie lat dziewięćdziesiątych istniało ponad 50 metod zorientowanych obiektowo i tyle samo formatów projektowych. Potrzeba opracowania **ujednoliconego** języka modelowania stawała się coraz bardziej oczywista.

Na początku lat dziewięćdziesiątych szeroko rozpowszechnione były metodologie zorientowane obiektowo autorstwa Grady'ego Boocha i Jamesa Rumbaugha. W październiku 1994 roku firma Rational Software Corporation (od lutego 2003 wchodząca w skład korporacji IBM) rozpoczęła pracę nad ujednoliconym językiem modelowania. Na pierwszym etapie prac wprowadzono standardy dla notacji (język), ponieważ to zagadnienie uznano za najłatwiejsze w opracowaniu. Wykorzystano między innymi **metodę Boocha** (ang. **Booch Method** autorstwa Grady'ego Boocha), **OMT** (ang. **Object Modeling Technique** autorstwa Jamesa Rumbaugha) oraz **OOSE** (ang. **Object-Oriented Software Engineering** autorstwa Ivara Jacobsena) wraz z elementami innych metodyk. Opracowaną w ten sposób notację opublikowano jako język UML w wersji 0.9. Celem tych prac nie było sformułowanie zupełnie nowej notacji, lecz zaadaptowanie, rozwinięcie i uproszczenie istniejących metodyk, takich jak diagramy klas, diagramy użycia Jacobsona czy diagramy stanu Harela. W UML-u wykorzystano środki reprezentacji opracowane na potrzeby metod strukturalnych. Z tego powodu w diagramach aktywności języka UML znajdujemy wpływy diagramów przepływowych i sieci Petriego.

UML nie wyróżnia się zatem swoją nietypową formą, lecz tym, że narzuca daleko idącą standaryzację języka o formalnie zdefiniowanym znaczeniu stosowanych symboli.

W dalszych pracach nad językiem UML udział wzięły znane firmy, takie jak IBM, Oracle, Microsoft, Digital, Hewlett-Packard i Unisys. W 1997 roku opublikowano wersję 1.1 języka UML. Została ona zaakceptowana przez OMG. UML w wersji 1.2, zawierający pewne zmiany i poprawki, pojawił się w roku 1998, rok później powstała wersja 1.3, a UML 1.5 wypuszczono w marcu 2003. Tymczasem od roku 2000 trwały już prace nad wersją 2.0 UML-a, która została zaakceptowana przez OMG w postaci Final Adopted Specification w czerwcu 2003. Gdy oryginalna wersja tej książki trafiała do druku (czerwiec 2005), jeszcze nie zakończyła się ostatnia faza wdrażania standardu przez OMG (nie nastąpiło jeszcze przyjęcie specyfikacji jako **obowiązującej**).

2.6. Specyfikacja wymagań

Modele tworzonego systemu są integralną częścią każdej specyfikacji wymagań. W tej książce można znaleźć podstawy potrzebne do tworzenia tego typu modeli. Niestety, nie istnieje uniwersalna receptura tworzenia specyfikacji wymagań. Dobór szczegółów i poziom szczegółowości modelu zależą od wielu czynników. Z naszego doświadczenia wynika, że najważniejsze są odpowiedzi na następujące pytania:

- Kto definiuje specyfikację?
- Dla kogo jest ona przygotowywana?
- Jaki jest jej podmiot?

2.6.1. Wspomaganie procesów decyzyjnych

Modele i perspektywy zaprezentowane w tej książce stanowią podstawowe elementy składowe, z których — jak z cegieł — konstruuje się modele zdefiniowane w specyfikacjach wymagań. Poniższa tabela może być pomocna przy podejmowaniu decyzji dotyczących modeli i perspektyw.

Model (co?)	Perspektywa	Twórca (kto?)	Odbiorcy (dla kogo?)	Cel (po co?)
System biznesowy	Perspektywa zewnętrzna	Użytkownik	Użytkownik	Dokumentacja biznesowa
			Informatyk	Podstawy specyfikacji systemu informatycznego
	Perspektywa wewnętrzna	Użytkownik	Użytkownik	Dokumentacja biznesowa, opis procedur
			Informatyk	Podstawy specyfikacji systemu informatycznego
System informatyczny	Perspektywa zewnętrzna	Użytkownik	Informatyk Użytkownik	Wymagania w stosunku do systemu informatycznego
	Perspektywa strukturalna	Informatyk	Informatyk	Specyfikacja systemu informatycznego
	Perspektywa wydajności	Użytkownik Informatyk	Informatyk	Specyfikacja systemu informatycznego
	Perspektywa interakcji	Użytkownik Informatyk	Informatyk	Specyfikacja systemu informatycznego
Integracja systemów	Perspektywa procesów	Użytkownik	Informatyk	Specyfikacja integracji systemu informatycznego
	Perspektywa statyczna	Informatyk	Informatyk	Specyfikacja integracji systemu informatycznego

2.6.2. Weryfikacja

Wszystkie perspektywy omówione w tej książce opisują model, dokumentując wymagania w stosunku do niego z punktu widzenia użytkownika. Oznacza to, że wszystkie omawiane modele i perspektywy mają następujące wspólne cechy:

- mogą być skonstruowane wyłącznie przy ścisłej współpracy z użytkownikami;
- ich poprawność z punktu widzenia biznesowego może być zweryfikowana wyłącznie przez użytkowników,

Pomimo że model systemu informatycznego jest tworzony z myślą o docelowych odbiorcach, czyli informatykach zaangażowanych w implementację, nie ma możliwości jego realizacji bez udziału użytkowników, którzy muszą dostarczyć wymagania w stosunku do modelu. Reprezentują oni punkt widzenia użytkownika i są dostawcami wiedzy z dziedzin biznesowych.

W proces tworzenia i weryfikacji specyfikacji wymagań zaangażowane bywają różne grupy użytkowników, dlatego ważne jest wykorzystanie do tych celów ujednoliconego języka modelowania. Dzięki temu łatwiej uniknąć nieporozumień wynikających z błędnej interpretacji specyfikacji.

2.7. UML 2.0

2.7.1. Przegląd cech języka UML 2.0

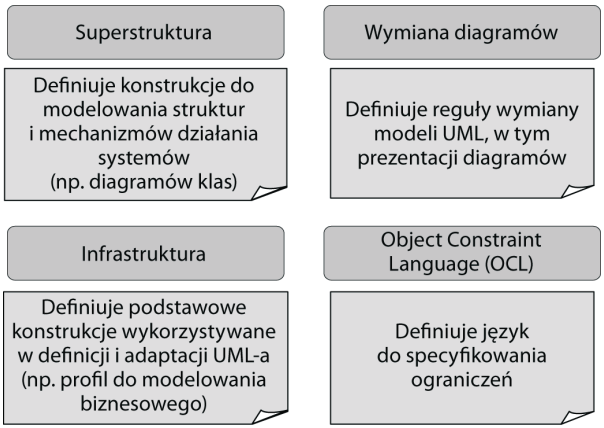
Książka *UML 2.0 w akcji. Przewodnik oparty na projektach* jest oparta na nowej wersji języka UML, czyli UML 2.0. W tej wersji struktura dokumentacji języka została zupełnie zmieniona. Obecnie istnieją dwa dokumenty opisujące UML:

- *UML 2.0 Infrastructure* opisuje podstawowe konstrukcje stanowiące fundament języka UML. Ten dokument nie przyda się bezpośrednio użytkownikowi (lub odbiorcy) języka UML; jest on przeznaczony raczej dla twórców narzędzi modelujących.
- *UML 2.0 Superstructure* definiuje konstrukcje języka UML 2.0 dostępne dla użytkownika — czyli te elementy UML-a, z którymi użytkownicy mają bezpośredni kontakt.

Ta wersja UML-a została opracowana przede wszystkim (ale nie tylko) przy następujących założeniach:

- zmiana struktury i przedefiniowanie UML w taki sposób, by uprościć jego wykorzystanie, implementację i adaptację;
- infrastruktura UML-a zawierająca takie cechy, jak:
  - wspólny rdzeń w postaci metajęzyka, za pomocą którego UML może zdefiniować sam siebie;
  - mechanizm umożliwiający doskonalenie języka.
- nadstruktura (ang. *superstructure*) UML-a posiadająca takie cechy, jak:
  - możliwość prostego tworzenia modeli w oparciu o komponenty;
  - możliwość udoskonalania elementów specyfikacji architektury;
  - możliwość dostarczania lepszych narzędzi do modelowania zachowań.

Oprócz specyfikacji *UML Infrastructure* i *UML Superstructure* opublikowane zostały dodatkowe dokumentacje, między innymi język **Object Constraint Language** (**OCL**) oraz diagram Interchange (dotyczący wymienności diagramów). Obecnie dokumenty te wchodzą w skład pakietu UML 2.0, co obrazuje rysunek 2.12.



Rysunek 2.12. Pełny pakiet specyfikacji UML 2.0

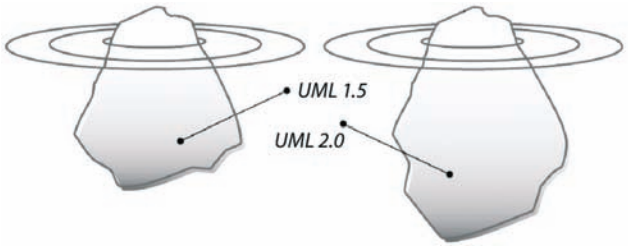
UML 2.0 jako całość jest bardziej rozbudowany i skomplikowany od swoich poprzednich wersji. Zakres dokumentacji dotyczącej tego języka również znacznie się poszerzył. Dokumentacja UML 1.5 wraz ze specyfikacją OCL miała rozmiar 730 stron, natomiast dokumentacja UML 2.0 — również z OCL — ma rozmiar ponad 1050 stron.

Pomijając już nawet fakt, że część dokumentacji „nie dotyczy" zwykłego użytkownika, przeczytanie całości przekracza możliwości przeciętnego członka zespołu twórców systemu. Nie chodzi tu wyłącznie o rozmiar mierzony liczbą stron, lecz przede wszystkim o poziom komplikacji konstrukcji UML-a. Z tego powodu bardzo ważnym zagadnieniem stało się opracowanie uproszczonego podzbioru elementów tego języka.

Konsekwencją takiego podejścia są dwa ważne spostrzeżenia dotyczące książki *UML 2.0 w akcji. Przewodnik oparty na projektach*. Książka ma za zadanie wykreować uproszczony obraz UML-a. Taki cel jest uzasadniony tym bardziej, że wersja 2.0 nie jest wcale bardziej przystępna dla użytkownika od wersji wcześniejszych.

Na szczęście naprawdę niewiele cech UML 2.0 ma wpływ na poziom szczegółowości zagadnień przedstawionych w tej książce. Zawiera ona niewiele zmian w porównaniu z wcześniejszymi wydaniami (w języku niemieckim), które oparte były na poprzednich wersjach UML-a. Ograniczony zakres tematyki pozwala na zachowanie stabilności merytorycznej tej publikacji dla każdej nowej wersji UML-a.

W książce świadomie prezentujemy jedynie wierzchołek góry lodowej i robimy to z pełną konsekwencją. Prezentowany wierzchołek (rysunek 2.13) ma ciągle ten sam rozmiar, niezależnie od tego, że część góry ukryta pod wodą stale się powiększa. Im bardziej ta sytuacja się rozwija, tym większe mamy przekonanie, że ten wierzchołek w zupełności wystarcza oczekiwanym odbiorcom książki, czyli członkom zespołów informatycznych wdrażających projekty.



Rysunek 2.13. Góra lodowa UML-a

Chcielibyśmy również podkreślić nowe możliwości, jakie daje UML 2.0. Jednym z założeń przyjętych przy tworzeniu UML 2.0 było stworzenie formalnej i kompletnej semantyki. Wykorzystanie tej cechy UML 2.0 pozwala w automatyczny sposób generować z modeli szkielety systemów. Taka cecha UML-a daje następujące możliwości:

- model opisany za pomocą UML-a nie jest abstrakcją, a odzwierciedleniem rzeczywistego systemu;
- błędy popełnione podczas konstrukcji modelu można wykrywać i poprawiać w sposób ciągły już od wczesnych etapów rozwoju;
- nie ma konieczności stosowania etapów pośrednich, takich jak modyfikowanie kodu niewchodzącego w skład projektu modelu;
- ten sam model może działać na różnych platformach (sprzętowych i programowych).

Te zalety okupione są jednak koniecznością uzyskania dogłębnej wiedzy na temat UML-a i włożenia dużego wysiłku w opracowanie modeli w sposób umożliwiający wykorzystanie pozytywnych cech tego języka.

2.7.2. Wpływ zmian w UML-u na modelowanie systemu biznesowego

Możliwości UML-a w zakresie modelowania systemów biznesowych zostały zwiększone dzięki zmianom wprowadzonym w elementach modelowania aktywności. Oto kilka przykładów zmian i usprawnień.

Diagramy aktywności nie są już po prostu szczególnymi przypadkami diagramów stanu. Ta zmiana sama w sobie nie ma większego znaczenia dla użytkownika UML-a. Jednak w połączeniu z nową autonomią metamodelu oferuje nowe, interesujące możliwości.

- Do tej pory osobne etapy w diagramie aktywności były interpretowane jako aktywności. Obecnie cały diagram interpretuje się jako **aktywność**, natomiast etapy (nazywane wcześniej aktywnościami) obecnie określa się mianem **akcji**. Akcja może wykorzystywać prostą operację lub wywoływać inną aktywność. Dzięki temu przy tworzeniu modelu można przyjąć zasadę zstępującą (od ogółu do szczegółu).
- Podział modelu na składowe nie musi wiązać się z koniecznością synchronizacji.
- Aktywność może mieć więcej niż jeden stan początkowy. Dzięki temu można zaplanować jednoczesne uruchomienie kilku zdarzeń.
- Aktywność może mieć zdefiniowane parametry wejściowe i wyjściowe.

Jedno z usprawnień diagramu sekwencji polega na uzupełnieniu go o koncepcję **operatorów**. Operatory umożliwiają połączenie kilku akcji lub aktywności w ramach jednego diagramu sekwencyjnego. Mogą one na przykład zostać użyte do zamodelowania wpływu innych diagramów sekwencyjnych na poszczególne sekwencje diagramu. Odpowiednie operatory można również wykorzystać do reprezentacji iteracji. Dzięki nowo wprowadzonym operatorom diagramy sekwencji również można tworzyć metodą zstępującą.

OCL jest obecnie integralną częścią UML-a. Tego języka można użyć do opisu argumentów, inwariantów, warunków wstępnych i wyjściowych w ramach modeli UML. Dzięki temu modele systemów i procesów biznesowych można projektować o wiele precyzyjniej.

2.7.3. Wpływ zmian w UML-u na modelowanie systemu informatycznego

Diagramy prezentowane w tej książce, reprezentujące różne perspektywy systemów IT, nie uległy znaczącym zmianom.

Największe zmiany dotyczyły notacji diagramów sekwencji. Tutaj nowa jest możliwość modularyzacji dzięki referencjom interakcji. W książce nie wprowadziliśmy jednak żadnych zmian, jeśli chodzi o poziom szczegółowości diagramów sekwencji. To samo dotyczy diagramów klas oraz przypadków użycia.

Najciekawszym zmianom w przypadku modelowania systemów informatycznych uległy diagramy stanu: punkty połączeń pozwalają na przykład na lepszy poziom modularyzacji diagramów stanu. Ze względu na założenie dotyczące uproszczenia prezentowanych zagadnień UML-a zdecydowaliśmy się jednak nie omawiać szczegółowo tego zagadnienia.

2.7.4. Wpływ zmian w UML-u na modelowanie integracji systemów

Oczywiście, usprawnienia w modelowaniu zachowań miały wpływ na perspektywę procesu w modelu integracji systemów. Zostały wprowadzone znaczące udoskonalenia polegające na wprowadzeniu parametrów wejściowych i wyjściowych dla aktywności (zobacz punkt 2.7.2, „Wpływ zmian w UML-u na modelowanie systemu biznesowego").

Niewielkie zmiany nastąpiły także w obszarze perspektyw statycznych, co ma wpływ na projektowanie obiektów biznesowych z diagramami klas.

Oprócz zmian wprowadzonych wraz z wersją 2.0 na znaczeniu zyskał profil **Enterprise Application Integration** (**EAI**), bardzo użyteczny przy integracji systemów. Oprócz podstawowych operacji niezbędnych przy integracji systemów profil EAI pozwala na prezentację metamodeli danych również dla nieobiektowych języków programowania. Zagadnienia te są jednak wykorzystywane na znacznie wyższym poziomie szczegółowości niż przyjęty w tej książce.

2.7.5. Podsumowanie

Dla zwykłego użytkownika UML 2.0 nie stanowi znacznej rewolucji w stosunku do wcześniejszych wersji, z pewnością jest jednak postrzegany jako usprawnienie koncepcji istniejących dotychczas. Warto rozważyć wykorzystanie UML-a 2.0 w przyszłych wdrożeniach projektów. Z drugiej strony, nic nie stoi na przeszkodzie, aby w istniejących projektach nadal stosować wykorzystaną wcześniej terminologię. W przypadku nowych projektów zalety UML-a 2.0 (dokładniejsze modelowanie) na pewno zrekompensują jego wady (na przykład większy nakład pracy).