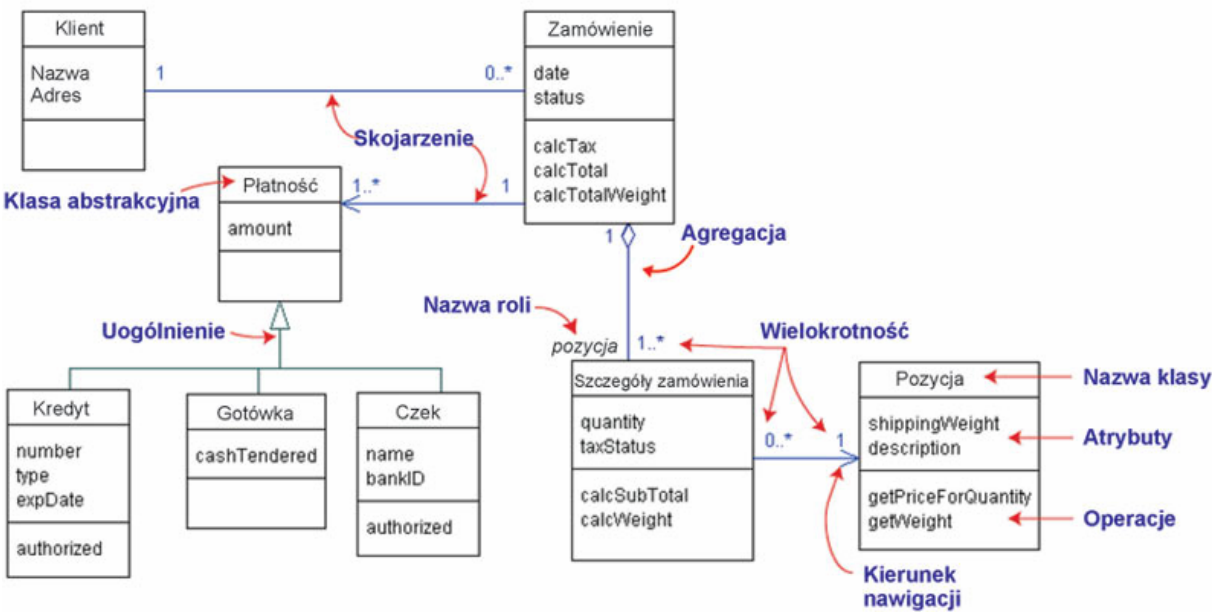
23. OMÓW PODSTAWOWE DIAGRAMY UML (klas, obiektów, przypadków użycia, stanów, przebiegu, czynności, kooperacji, komponentów, wdrożenia)

Diagram klas

przedstawia ogólną panoramę systemu, pokazując klasy i ich wzajemne relacje. Diagramy klas są statyczne - pokazują, co wchodzi w interakcje, a nie co się dzieje podczas tych interakcji.

Poniższy diagram klas modeluje zamówienie z katalogu. Centralna klasa to Zamówienie. Z tą klasą związany jest Klient, który dokonuje zakupu, oraz Płatność. Są trzy rodzaje Płatności: Gotówka, Czek lub Kredyt. Zamówienie zawiera SzczegółyZam (pozycje transakcji), każdy z powiązaną Pozycją.



Notacja klas w języku UML to prostokąt podzielony na trzy części: nazwa klasy, atrybuty i operacje. Nazwy klasy abstrakcyjnych, takich jak Płatność, są pisane kursywą. Relacje między klasami są zilustrowane przez łączące je linie.

Nasz diagram klas zawiera trzy rodzaje relacji:

- **Skojarzenie** - związek między instancjami dwóch klas. Skojarzenie dwóch klas zachodzi wtedy, gdy jedna klasa musi wiedzieć o drugiej, aby wykonywać swoje zadania. Na diagramie skojarzeniem jest linia łącząca dwie klasy.
- **Agregacja** - skojarzenie, w którym jedna z klas należy do kolekcji. Agregacja jest zakończona rombem wskazującym tę część, która zawiera całość. Na naszym diagramie Zamówienie ma kolekcję SzczegółyZam.
- **Uogólnienie** - łącze dziedziczenia, które wskazuje, że jedna klasa jest nadrzędna w stosunku do drugiej. Uogólnienie ma trójkąt wskazujący klasę nadrzędną. Płatność to klasa nadrzędna klas Gotówka, Czek i Kredyt.

Skojarzenie ma dwa końce. Koniec może mieć **nazwę roli**, która wyjaśnia naturę skojarzenia. Na przykład SzczegółZam jest pozycją każdego Zamówienia. Strzałka **możliwości nawigacji** pokazuje kierunek, w którym można przechodzić lub odpytywać skojarzenie. SzczegółZam można zapytać o Pozycję, ale nie odwrotnie. Strzałka informuje też, kto jest "właścicielem" implementacji skojarzenia; w tym przypadku, SzczegółZam ma Pozycję. Skojarzenia bez strzałek mają możliwości nawigacji dwukierunkowe.

Wielokrotność końca skojarzenia to dopuszczalna liczba instancji klasy skojarzonych z jedną instancją na drugim końcu. Wielokrotności są pojedynczymi liczbami albo zakresami liczb. W naszym przykładzie może być tylko jeden Klient na każde Zamówienie, ale Klient może mieć dowolną liczbę Zamówień.

Poniższa tabela opisuje najczęściej używane wielokrotności.

Wielokrotności	Znaczenie
0..1	Brak instancji lub jedna instancja. Notacja n . . m oznacza od n do m instancji.
0..* lub *	Bez ograniczenia liczby instancji (łącznie z brakiem instancji).
1	Dokładnie jedna instancja
1..*	Przynajmniej jedna instancja

Więcej informacji na stronie http://bdn.borland.com/article/images/31863/classdiagram.html

Każdy diagram klas zawiera klasy, skojarzenia i wielokrotności. Możliwości nawigacji i role to elementy opcjonalne, które zwiększają czytelność diagramu klas.

Diagram obiektów

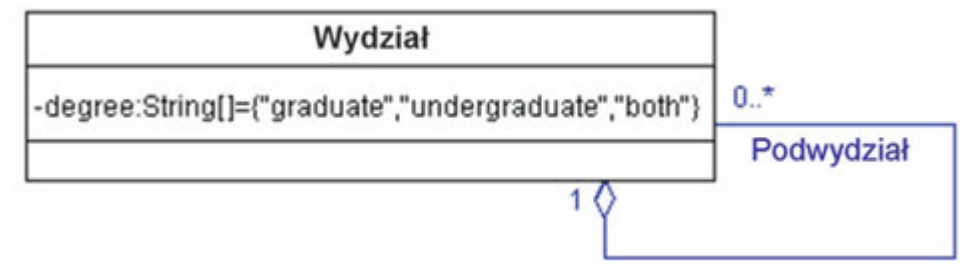
Diagram obiektów obrazuje zbiór obiektów i ich związków w ustalonej chwili. Jest grafem złożonym z krawędzi i wierzchołków.
Zawiera na ogół:

- Obiektów
- Wiązania

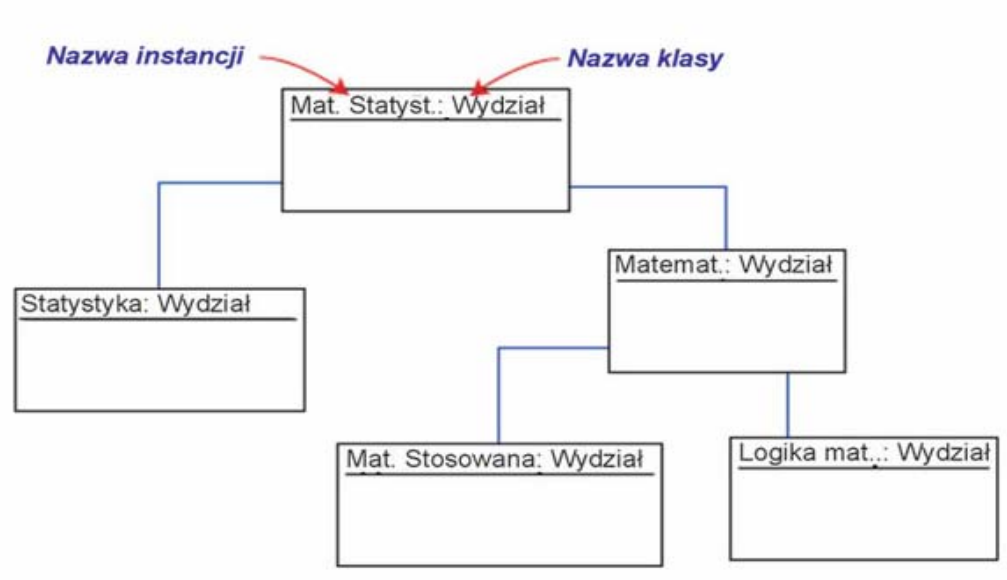
Diagramów obiektów będziesz używać na ogół w jednym celu a mianowicie do modelowania struktur obiektowych.

Diagramy obiektów zamiast klas pokazują instancje. Przydają się do wyjaśniania drobnych elementów ze skomplikowanymi relacjami, zwłaszcza rekurencyjnymi.

Poniższy niewielki diagram klas pokazuje, że Wydział uniwersytetu może zawierać wiele innych Wydziałów.



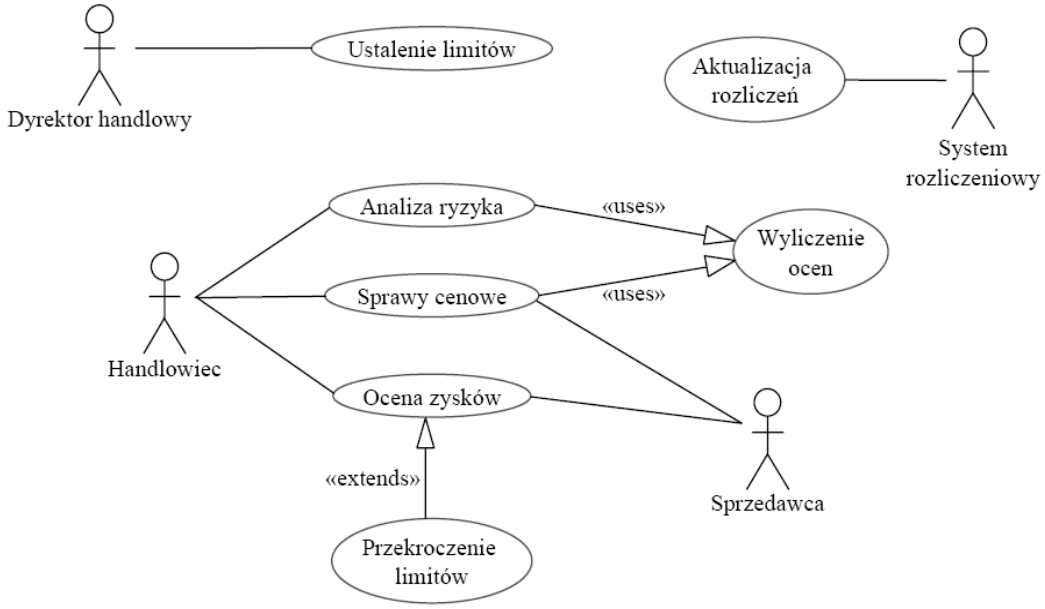
Poniższy diagram obiektów konkretyzuje diagram klas, zastępując go rzeczywistym przykładem.



Każdy prostokąt na diagramie obiektów odpowiada pojedynczej instancji. Nazwy instancji na diagramach UML są podkreślone. Nazwy klas lub instancji mogą zostać pominięte na diagramach obiektów, pod warunkiem, że sens diagramu pozostaje jasny.

Diagramy przypadków użycia

Diagramy przypadków użycia są bardzo proste. Rys.1 przedstawia diagram przypadków użycia dla pewnej (hipotetycznej) firmy zajmującej się pośrednictwem w sprzedaży.



Rys.1. Przykładowy diagram przypadków użycia

Model przypadków użycia dostarcza bardzo abstrakcyjnego poglądu na system z pozycji aktorów, którzy go używają. Nie włącza jakichkolwiek szczegółów, co pozwala wnioskować o systemie na odpowiednio ogólnym, abstrakcyjnym poziomie. Diagram przypadków użycia zawiera znaki graficzne oznaczające aktorów (ludziki) oraz przypadki użycia (owale z wpisanym tekstem). Te oznaczenia połączone są liniami odwzorowującymi powiązania poszczególnych aktorów z poszczególnymi przypadkami użycia.

Podstawowym zastosowaniem takich diagramów jest dialog z użytkownikami zmierzający do sformułowania wymagań na system. Stąd wynika, że diagramy i opis przypadków użycia muszą być bardzo naturalne, proste i nie mogą wprowadzać elementów komputerowej egzotyki i żargonu. W późniejszej fazie przypadki użycia mogą być wyspecyfikowane bardziej precyzyjnie przy pomocy notacji lub diagramów obiektowych. Następną fazą w postępowaniu opartym na przypadkach użycia jest rozpisanie ich przy pomocy notacji wprowadzanych w innych diagramach UML. Należy podkreślić, że tworzenie przypadków użycia jest niezdeterminowane i subiektywne. Na ogół różni analitycy tworzą różne modele.

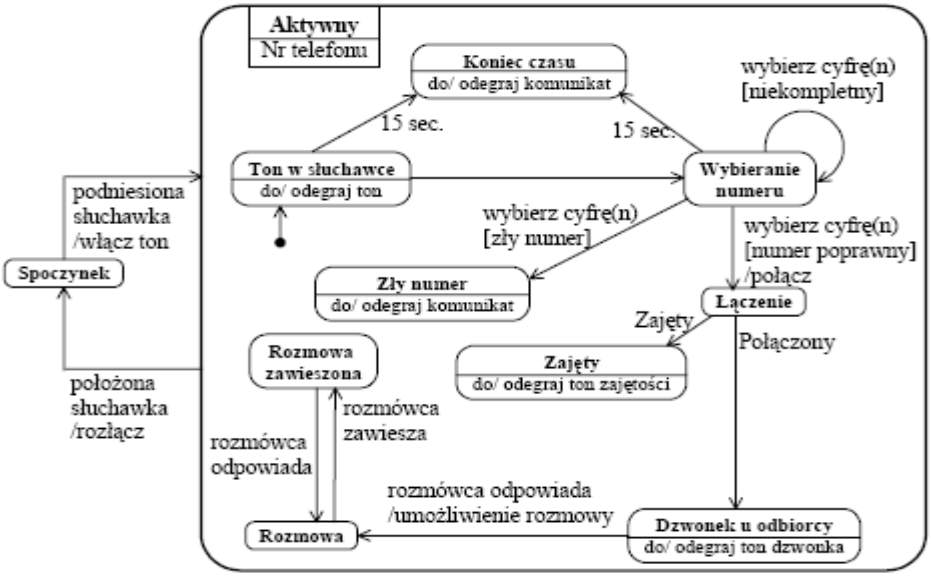
Metoda przypadków użycia wymaga od analityka określenia wszystkich aktorów związanych w jakiś sposób z projektowanym systemem. Każdy aktor używa lub będzie używać systemu na kilka specyficznych sposobów (przypadków użycia). Zazwyczaj aktorem jest osoba, ale może nim być także pewna organizacja (np. biuro prawne) lub inny system komputerowy. Należy zwrócić uwagę, że metoda modeluje aktorów, a nie konkretne osoby. Jedna osoba może pełnić rolę wielu aktorów; np. być jednocześnie sprzedawcą i klientem. I odwrotnie, jeden aktor może odpowiadać wielu osobom, np. strażnik budynku. Aktor jest pierwotną przyczyną napędzającą przypadki użycia. Jest on sprawcą zdarzeń powodujących uruchomienie przypadku użycia, jak też odbiorcą informacji wyprodukowanych przez przypadki użycia. Aktor reprezentuje rolę, którą może grać w systemie jakiś jego użytkownik, np. kierownik, urzędnik, klient. Można tworzyć aktorów abstrakcyjnych, na podobieństwo klas abstrakcyjnych. Np. aktor „urzędnik” jest nadklasą dla aktorów „kasjer” i „dyrektor”; powiązania z konkretnymi przypadkami użycia mogą być ustalone zgodnie z tą hierarchią klas aktorów.

Przypadek użycia reprezentuje sekwencję operacji lub transakcji wykonywanych przez system w trakcie interakcji z użytkownikiem; np. potwierdzenie pisma, złożenie zamówienia. Przypadki użycia reprezentują przepływ zdarzeń w systemie i są uruchamiane (inicjowane) przez aktorów.

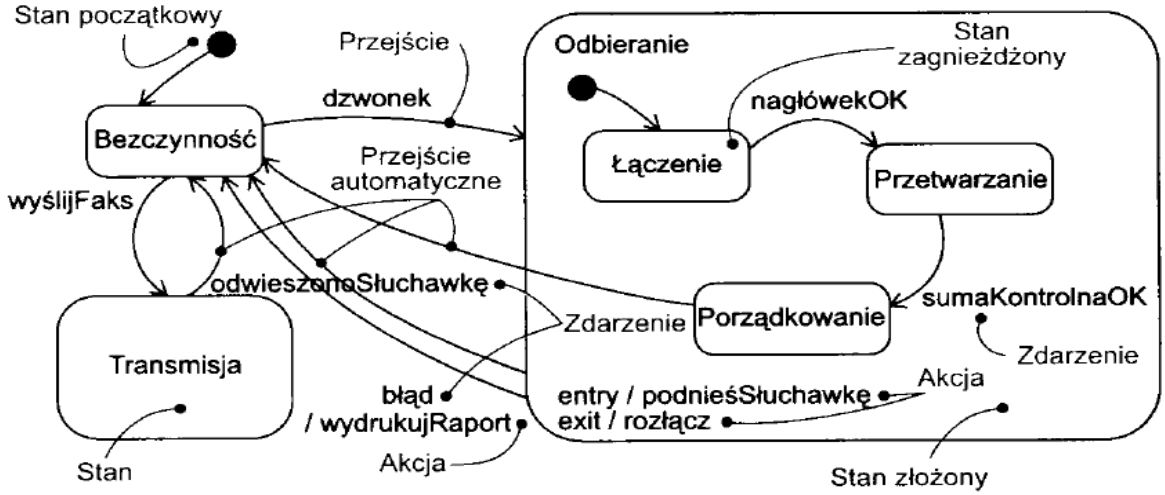
Przypadek użycia jest zwykle charakteryzowany przez krótką nazwę. Opis przypadku użycia może być jednak dowolnie rozbudowany, w szczególności może zawierać następujące fragmenty:
 Jak i kiedy przypadek użycia zaczyna się i kończy?
 Opis interakcji przypadku użycia z aktorami, włączając w to *kiedy* interakcja ma miejsce i *co* jest przesyłane.
 Kiedy i do czego przypadek użycia potrzebuje danych zapamiętanych w systemie, lub jak i kiedy zapamiętuje dane w systemie?
 Wyjątki przy przepływie zdarzeń.
 Jak i kiedy używane są pojęcia dziedziny problemowej?
UML wprowadza także bardzo proste powiązania pomiędzy przypadkami użycia, oznaczane strzałkami dodatkowymi napisami «extends» (rozszerza) i «uses» (używa), Rys.1. Przypadek użycia podłączony przez strzałkę «extends» oznacza, że niekiedy (nie zawsze) dany przypadek użycia jest rozszerzony przez inny przypadek użycia. Przypadek użycia podłączony przez strzałkę «uses» oznacza wspólny fragment w wielu przypadkach użycia, który warto jest wyodrębnić ze względu na jego podobieństwo koncepcyjne oraz ze względu na późniejszą możliwość uniknięcia wielokrotnej implementacji tego fragmentu. Taki fragment jest niekiedy określany jako blok ponownego użycia (*reuse*).
Typowa dokumentacja przypadków użycia powinna zawierać następujące elementy:
 Krótki opis przypadku użycia.
 Przepływ zdarzeń opisany nieformalnie.
 Związki pomiędzy przypadkami użycia.
 Uczestniczące obiekty.
 Specjalne wymagania (np. czas odpowiedzi, wydajność).
 Obrazy interfejsu użytkownika.
 Ogólny pogląd na przypadki użycia (powiązania w postaci diagramów).
 Diagramy interakcji dla każdego aktora.

Diagram stanów

Diagram stanów w swojej idei nawiązuje do automatu skończonego. Opisuje on stany pewnego procesu, które są istotne z punktu widzenia modelu pojęciowego tego procesu, oraz przejścia pomiędzy stanami. W swojej pierwotnej idei diagram stanów miał odwzorowywać stany obiektów pewnej klasy podczas ich cyklu życiowego oraz przejścia (*transitions*) pomiędzy tymi stanami powodowane przez zdarzenia lub komunikaty. Jak się wydaje (sądząc z przykładu zamieszczonego w dokumentacji UML) ta pierwotna idea uległa jednak na tyle silnej modyfikacji, że w istocie diagramy stanów *nie są* tymi diagramami stanów, o których jest mowa w teorii automatów. Są to dość klasycznie diagramy przepływu sterowania (*flowcharts*), z szeregiem drugorzędnych opcji syntaktycznych i semantycznych.
Na Rys.13 przedstawiony został przykładowy diagram stanów (z oczywistych względów uproszczony). Stany, w istocie reprezentujące stany sterowania procesu, zostały przedstawione w postaci prostokątów z zaokrąglonymi rogami. Przejścia pomiędzy stanami zostały zaznaczone w postaci strzałek. Tego rodzaju diagramy mogą być zagnieżdżane, tj. dowolny „stan" reprezentowany na diagramie może być uszczegółowiony w postaci odrębnego diagramu.



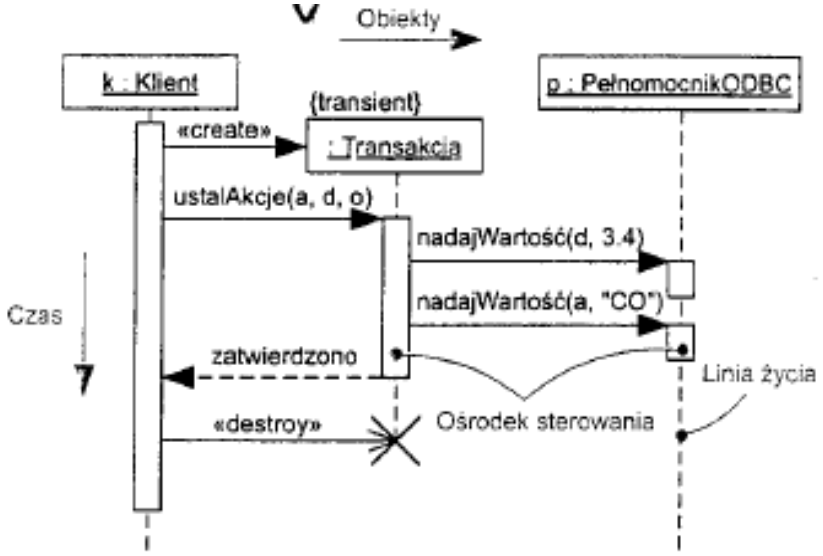
Rys.13. Diagram stanów



Rys. 24.1. Diagram stanów

Diagramy przebiegu

Na diagramie przebiegu uwypukla się kolejność komunikatów w czasie. Opracowanie takiego diagramu rozpoczynamy od umieszczenia w jego górnej części, wzdłuż osi X, obiektów uczestniczących w interakcji.

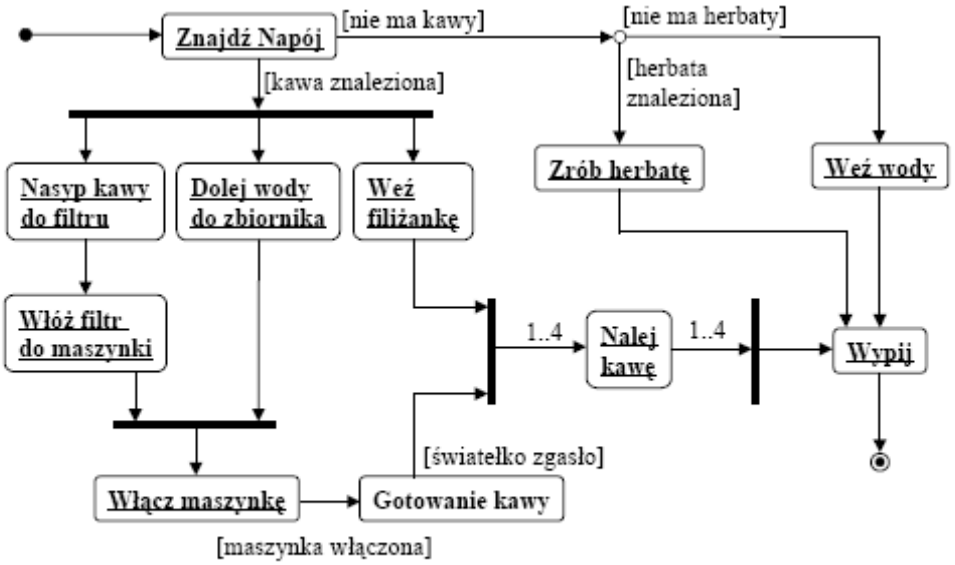


Obiekt inicjujący interakcję znajduje się zazwyczaj po lewej stronie diagramu a coraz podrzędniejsze kolejno po prawej. Następnie dopisujemy komunikaty wysłane i odbierane przez te obiekty. Komunikaty są uporządkowane w Czesie wzdłuż osi Y w myśl zasady: im późniejsza chwila wysłania, tym komunikat umieszczony niżej. Taki sposób obrazowania ułatwia czytelnikowi diagramu zrozumienie przepływu sterowania w czasie.

Diagramy przebiegu mają dwie cechy, które odróżniają je od diagramów kooperacji.
Po pierwsze występują na nich linie życia obiektów – pionowe przerywane. Podczas interakcji mogą powstawać nowe obiekty. Ich linie życia rozpoczynają się w chwili odebrania przez nie komunikatu „create”. Pewne obiekty są niszczone . Ich linie życia kończą się w chwili odebrania przez nie komunikatu „destroy”.
Po drugie na diagramach przebiegu uwzględniony jest ośrodek sterowania – podłużny cienki prostokąt reprezentujący okres wykonywania jakiejś akcji. Zagnieżdżenie sterowania jest przedstawiane za pomocą innego ośrodka sterowania umieszczonego trochę na prawo od jego przodka.

Diagramy aktywności (czynności)

Diagramy aktywności w swej zasadniczej idei są dokładnie tym samym, co diagramy przepływu sterowania (*flowcharts, control flow diagrams*). Jedyną różnicą koncepcyjną jest to, że pojawiają się na nich elementy synchronizacji równoległych procesów (w dość uproszczonej formie, która dla pewnych celów może być niewystarczająca) Zdaniem autorów UML, diagramy aktywności są szczególnym przypadkiem diagramów stanów. Wydaje się jednak, że wprowadzenie diagramów stanów i diagramów aktywności w UML jest redundantne: w gruncie rzeczy różnice sprowadzają się do składni i niezbyt klarownych ustaleń, jakie informacje mogą być prezentowane na każdym z tych typów diagramów. Można sądzić, że obecność tych dwóch środków w ramach jednego języka jest powodowana nie merytoryczną potrzebą, lecz pewnymi zaszłościami historycznymi.

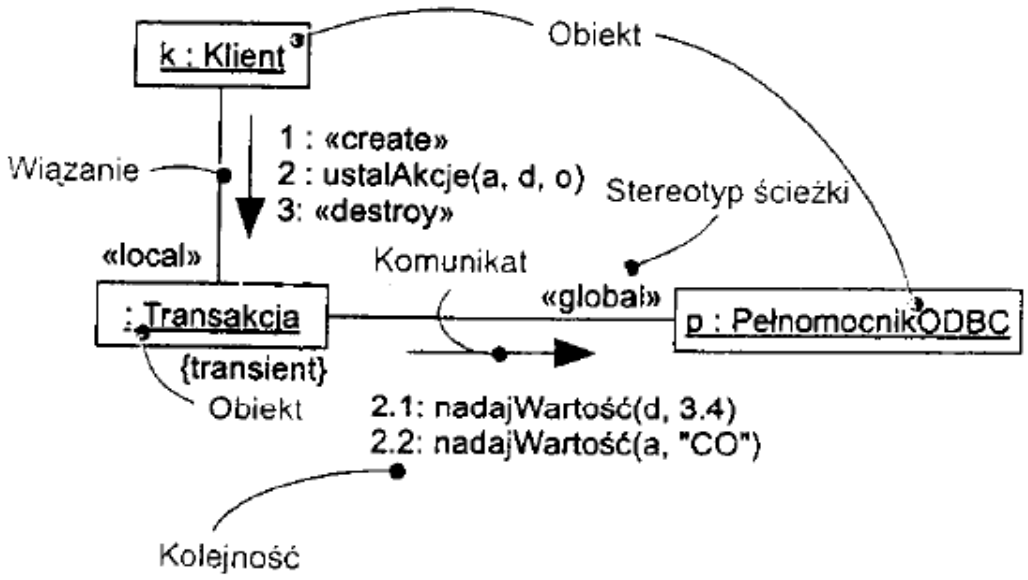


Rys.14. Diagram aktywności

Rys.14 przedstawia diagram aktywności. Grube linie oznaczają miejsce „rozejścia się" i synchronizacji równoległych procesów. Pozostałe elementy składni, semantyki i pragmatyki diagramów aktywności wydają się dość oczywiste. Tak samo jak w przypadku diagramów stanów, zastosowanie diagramów aktywności sprowadza się do przypadków, kiedy opisywany proces lub przypadek użycia jest dostatecznie złożony, ale możliwy do ogarnięcia na diagramie graficznym o niezbyt dużych rozmiarach.

Diagramy kooperacji

Na diagramie kooperacji uwypukla się organizację obiektów uczestniczących w interakcji. Opracowanie takiego diagramu rozpoczynamy od rozmieszczenia tych obiektów jako wierzchołków grafu. Na koniec dodajemy do wiązań komunikaty wysłane i odebrane przez obiekty. Taki diagram ułatwia zrozumienie przepływu sterowania w kontekście organizacji strukturalnej współpracujących obiektów.



Diagramy kooperacji mają dwie cechy różniące je od diagramów przebiegu.
Po pierwsze występują na nich ścieżki. Aby wskazać sposób połączenia jednego obiektu z drugim wystarczy dodać odpowiedni stereotyp ścieżki do drugiego końca wiązani (np. „local" oznaczający, że wyszczególniony obiekt jest w lokalnym zasięgu nadawcy)
Po drugie na diagramach kooperacji uwzględnia się ciąg komunikatów. Aby wskazać kolejność komunikatów w czasie wystarczy poprzedzić go odpowiednim jego numerem w ciągu. Zagnieżdżenie obrazuje się za pomocą notacji Deweya (1 – pierwszy komunikat, 1.1 – pierwszy komunikat zagnieżdżony w komunikacie 1 itd.)

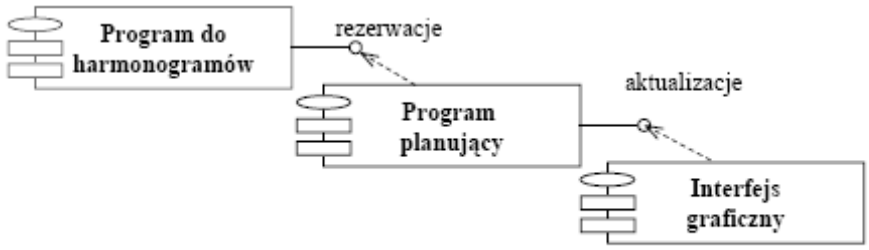
Diagramy przebiegu i kooperacji są równoważne, ponieważ reprezentują tę samą informację w metamodelu UML (a co to?). Można zatem przekształcić jeden w drugi bez groźby utraty informacji.

Diagramy implementacyjne

Diagramy implementacyjne pokazują niektóre aspekty implementacji SI, włączając w to strukturę kodu źródłowego oraz strukturę kodu czasu wykonania (*run-time*). W UML wprowadza się dwa typy takich diagramów: diagramy komponentów pokazujące strukturę samego kodu oraz diagramy wdrożeniowe pokazujące strukturę systemu czasu wykonania.

Diagramy komponentów

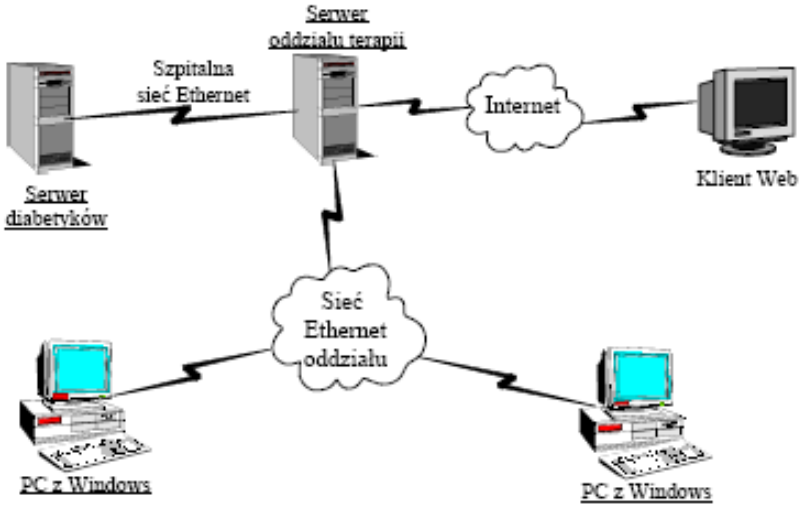
Diagramy komponentów pokazują zależności pomiędzy komponentami oprogramowania, włączając komponenty kodu źródłowego, kodu binarnego oraz kodu wykonywalnego. Poszczególne komponenty mogą istnieć w różnym czasie: niektóre z nich w czasie kompilacji, niektóre w czasie konsolidacji (*linking*) zaś niektóre w czasie wykonania. Diagram komponentów jest przedstawiany jako graf, gdzie węzłami są komponenty, zaś strzałki (z przerywaną linią) prowadzą do klienta pewnej informacji do jej dostawcy. Rodzaj zależności jest zależny od typu języka programowania. Diagram może także pokazywać interfejsy poszczególnych komponentów. Strzałki oznaczające zależności mogą prowadzić od komponentu do interfejsu, Rys.15.



Rys.15. Diagram komponentów

Diagramy wdrożeniowe

Diagram wdrożeniowe pokazują konfigurację elementów czasu wykonania: komponentów sprzętowych, komponentów oprogramowania, procesów oraz związanych z nimi obiektów. Komponenty, które nie istnieją w trakcie czasu wykonania nie pojawiają się na tych diagramach; powinny one być pokazane na diagramach komponentów.
Diagram wdrożeniowe jest grafem, gdzie węzłami są elementy czasu wykonania połączone przez linie odwzorowujące ich połączenia komunikacyjne. UML określa pewną standardowa postać tego rodzaju diagramów, ale poprzez odpowiednie stereotypy (oznaczenia graficzne) można uczynić tego rodzaju diagram bardziej czytelnym. W szczególności, całkowicie legalna jest postać przedstawiona na Rys.16.



Rys.16. Diagram wdrożeniowy