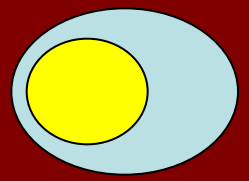
Faza analizy (modelowania)
Faza projektowania

- Celem fazy określania wymagań jest udzielenie odpowiedzi na pytanie: co i przy jakich ograniczeniach system ma robić? Wynikiem tej analizy jest zbiór wymagań czyli zewnętrzny opis systemu
- Celem fazy analizy jest udzielenie odpowiedzi na pytanie: jak system ma działać? Wynikiem jest logiczny model systemu, opisujący sposób realizacji przez system postawionych wymagań, lecz abstrahujący od szczegółów implementacyjnych
- Celem fazy projektowania jest udzielenie odpowiedzi na pytanie: jak system ma zostać zaimplementowany? Wynikiem jest projekt oprogramowania, czyli opis systemu implementacji

Model analityczny

Z reguły wykracza poza zakres odpowiedzialności systemu ponieważ:

- Ujęcie w modelu pewnych elementów nie będących częścią systemu czyni model bardziej zrozumiałym
- Na etapie modelowania może nie być jasne, które elementy modelu będą realizowane przez oprogramowanie, a które w sposób sprzętowy lub ręcznie
- Dostępne środki mogą nie pozwolić na realizację systemu w całości. Celem analizy może być wykrycie tych fragmentów które mogą być szczególnie ważne



Dlaczego modelujemy?

- *Podjęcie decyzji, jakie modele tworzyć, ma wielki wpływ na to, w jaki sposób zaatakujemy problem jaki kształt przyjmie rozwiązanie*
- *Każdy model może być opracowany na różnych poziomach szczegółowości*
- *Najlepsze modele odpowiadają rzeczywistości*
- *Żaden model nie jest wystarczający. Niewielka liczba niemal niezależnych modeli to najlepsze rozwiązanie w wypadku każdego niebanalnego systemu.*

Model analityczny

Model oprogramowania powinien być jego uproszczonym opisem – opisującym wszystkie istotne cechy oprogramowania na wysokim poziomie abstrakcji.

Cechy modelu:

- Uproszczony opis systemu
- Hierarchiczna dekompozycja funkcji systemu
- Model logiczny jest opisany za pomocą notacji zgodnej z pewną konwencją
- Zbudowany przy użyciu dobrze znanych metod i narzędzi
- Używany do wnioskowania o przyszłym oprogramowaniu

Model analityczny

- Pokazuje co system ma robić
- Jest zorganizowany hierarchicznie, według poziomów abstrakcji
- Unika terminologii implementacyjnej
- Pozwala na wnioskowanie „od przyczyny do skutku” i odwrotnie

Notacja w fazie analizy

Do opisu modelu stosuje się następujące rodzaje notacji:

- Język naturalny
- Notacje graficzne
- Specyfikacje – częściowo ustrukturalizowany zapis tekstowy i numeryczny

Szczególną rolę tutaj odgrywają notacje graficzne (wzorując się na innych dziedzinach techniki)

Notacja graficzna

- Diagramy graficzne ułatwiają szybkie przekazywanie podstawowych informacji, nie pozwala jednak na ukazanie wszystkich szczegółów.
- Nadmierne nagromadzenie szczegółów na diagramie graficznym może skutecznie zniweczyć jego czytelność
- Notacje graficzne uzupełniane są o dodatkowe informacje tzw. specyfikacje

Funkcje notacji

- Są one podstawowym narzędziem pracy analityka i projektanta
- Notacje wykorzystywane w fazie analizy są często wykorzystywane do współpracy z użytkownikiem (muszą być proste, przejrzyste, zrozumiałe)
- Ułatwia pracę w grupie. Służą szybkiemu i precyzyjnemu przekazywaniu idei
- Notacje są podstawą implementacji oprogramowania
- Służą do zapisu dokumentacji technicznej

Analiza strukturalna

Metody strukturalne tworzenia oprogramowania, opierają się na wyróżnianiu w tworzonym oprogramowaniu dwóch rodzajów składowych: *pasywnych* odzwierciedlających fakt przechowywania w systemie pewnych danych oraz *składowych aktywnych* odzwierciedlających fakt wykonywania w systemie pewnych operacji.

Metody obiektowe wyróżniają w systemie składowe, które łączą w sobie możliwość przechowywania danych oraz wykonywania operacji

Analiza strukturalna

- Analiza strukturalna zaczyna się od budowy dwóch różnych modeli systemu: modelu, będącego opisem pasywnej części systemu oraz modelu funkcji opisującego aktywną część systemu.
- Te dwa modele są następnie integrowane i wynikiem jest model przepływów danych.
- Zwolennicy analizy strukturalnej twierdzą, że budowa dwóch oddzielnych modeli ułatwia zrozumienie różnych aspektów systemu.
- Krytycy zwracają uwagę, na to że integracja tych dwóch modeli może być bardzo trudna.

Analiza obiektowa

System jest analizowany w sposób obiektowy jeśli:

- Jest dzielony na obiekty posiadające:
  - Tożsamość, Stan, Zachowanie
- Obiekty są grupowane w klasy złożone z obiektów o podobnych stanach i zachowaniu

Metody obiektowe są wygodnym narzędziem analizy złożonych systemów, w szczególności systemów o dużej stronie pasywności i złożonej funkcjonalności

Analiza obiektowa

Analiza i programowanie obiektowe ułatwia:

- Ukrywanie danych (hermetyzację)
- Wykorzystywanie gotowych elementów
- Ponowne wykorzystanie oprogramowania
- Szybkie prototypowanie
- Programowanie oparte na zdarzeniach
- Programowanie przyrostowe

UML
(ang. Unified Modeling Language)

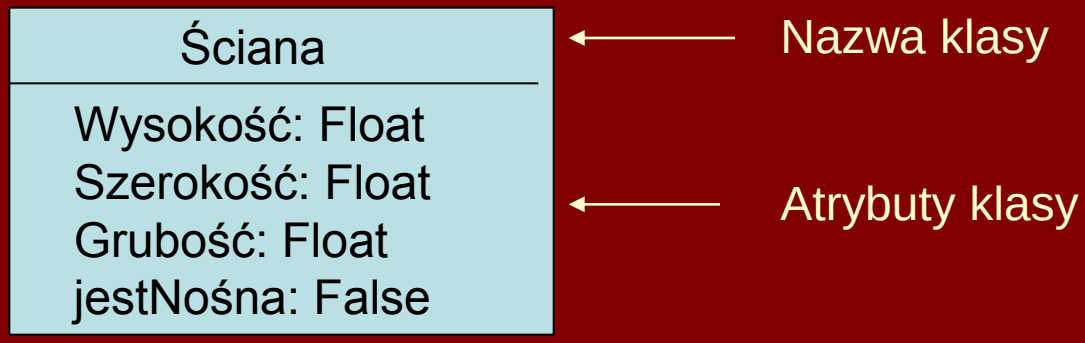
Graficzny język do obrazowania, specyfikowania, tworzenia i dokumentowania systemów informatycznych.

Klasy

- Modelowanie polega między innymi na zidentyfikowaniu bytów istotnych z pewnego szczególnego punktu widzenia. Byty te składają się na słownictwo systemu, który jest modelowany.
- DOM: ściany, drzwi, okna, szafki, oświetlenie
- Każdy byt znacząco różni się od pozostałych – ma swój własny zestaw właściwości
- Pojedyncze ściany, drzwi i okna rzadko istnieją w oderwaniu od siebie
- Wszystkie takie byty identyfikowane są jako klasy. Nie jest pojedynczym obiektem lecz zbiorem wielu obiektów.

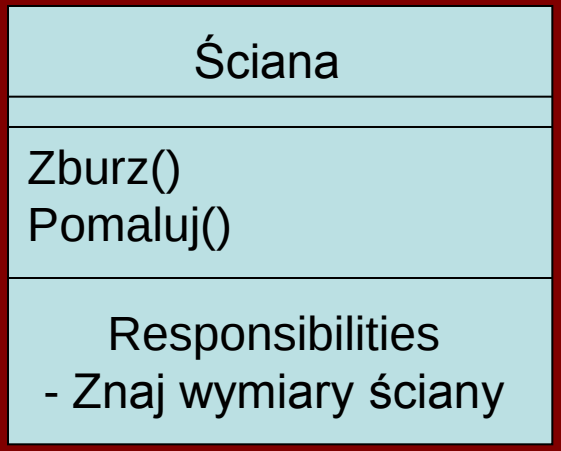
Klasy w UML

- Klasa to opis zbioru obiektów o tych samych atrybutach, związkach i znaczeniu. Jej symbolem graficznym jest prostokąt.
- Atrybut to nazwana właściwość klasy. Klasa może mieć dowolną liczbę atrybutów – lub nie mieć ich wcale. Atrybut reprezentuje właściwość pewnego modelowanego bytu, która jest określona dla wszystkich jego wystąpień.



Klasy w UML

- Operacja to implementacja pewnej usługi, której wykonania można zażądać od każdego obiektu klasy. Jest to abstrakcja, tego co można zrobić z każdym obiektem tej klasy.
- Odpowiedzialność klasy, czyli za jakie zadania dana klasa jest odpowiedzialna.



Związki

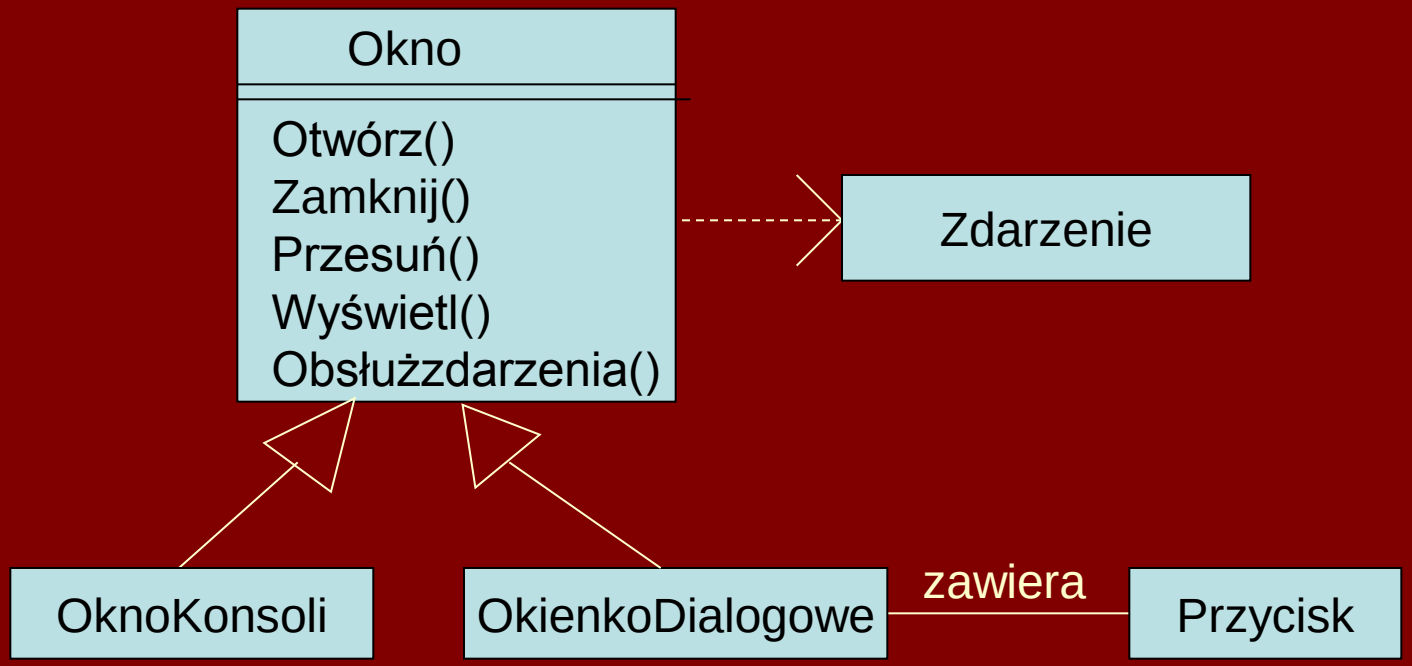
- Niewielka liczba klas występuje samotnie. Większość z nich współpracuje ze sobą na wiele sposobów.
- W modelowaniu obiektowym najważniejszymi rodzajami związków są *zależność*, *uogólnienie* i *powiązanie*.
- Zależność: zmiany dokonane w specyfikacji jednego elementu mogą mieć wpływ na inny element, który używa tego pierwszego.

Związki w UML

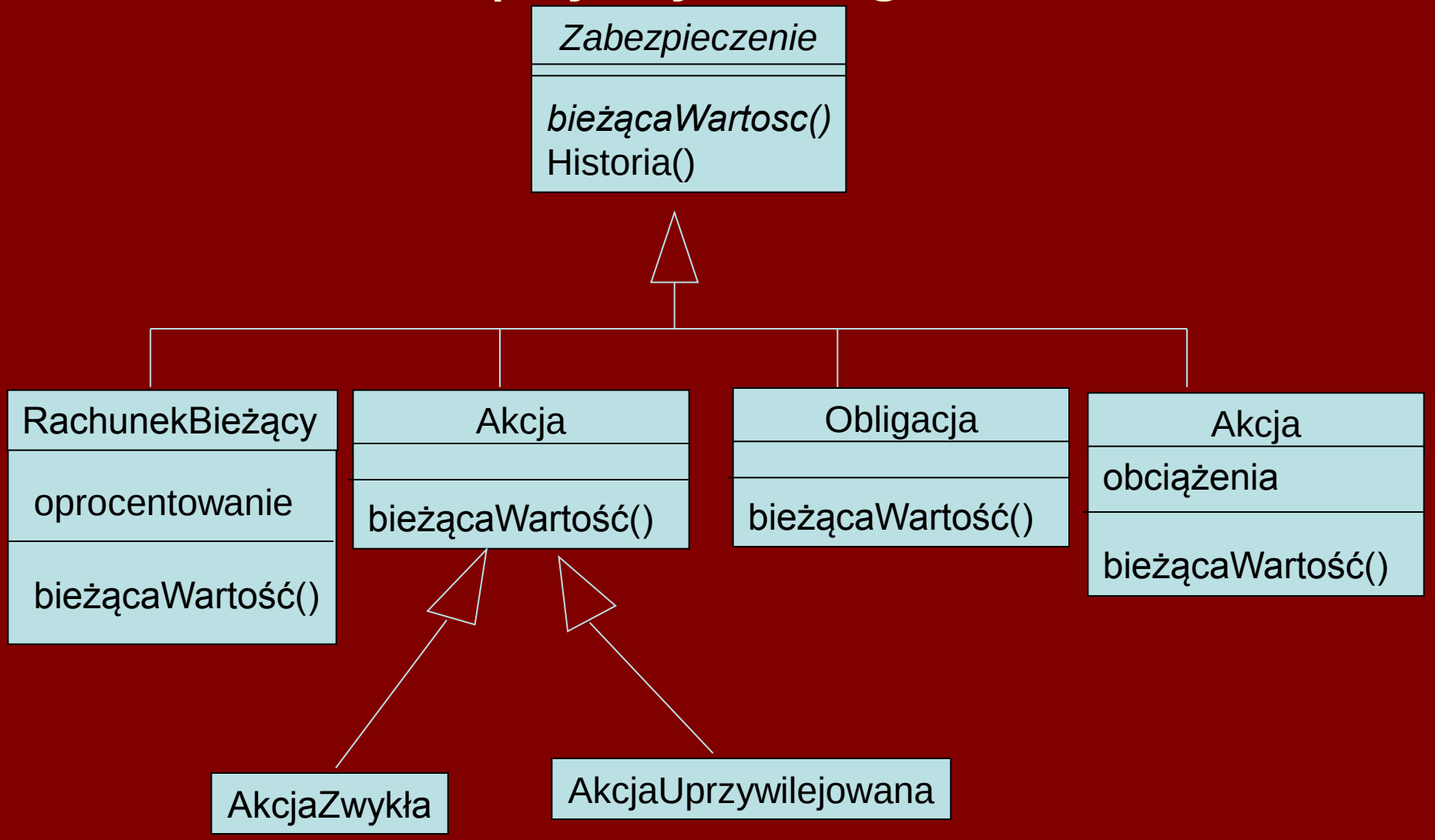
- Uogólnienie: to związek pomiędzy elementem ogólnym a pewnym specyficznym jego rodzajem (nadklasa – podklasa)
- Potomek (podklasa) może wystąpić wszędzie tam gdzie występuje przodek. Potomek dziedziczy wszystkie właściwości, a w szczególności atrybuty i operacje. Często potomek ma także własne właściwości. Operacja potomka, która ma taką samą sygnaturę co operacja przodka jest ważniejsza – polimorfizm.

Związki w UML

- Powiązania: związek, który wskazuje że elementy jednego elementu są połączone z obiektami innego.



Modelowania dziedziczenia pojedynczego



Przypadki użycia

- Przypadki użycia służą do utrwalania oczekiwanego zachowania budowanego systemu bez konieczności określania sposobu implementacji tego zachowania.
- Umożliwiają wypracowanie porozumienia między programistami a użytkownikami i ekspertami
- Ułatwiają weryfikację architektury i samego systemu w miarę tworzenia go.
- Przypadek użycia określa zbiór ciągów akcji, z których każdy reprezentuje interkację elementów z otoczenia systemu z samym systemem. Te działania są w istocie funkcjami systemowymi.
- Przypadki użycia są podstawą do opracowywania testów

Przypadki użycia w UML

- W UML diagramy przypadków użycia służą do obrazowania zachowania systemu, podsystemu klub klasy w taki sposób, by użytkownicy mogli zrozumieć jak z tego bytu skorzystać, a programiści mogli go zaimplementować
- Diagramy użycia można stosować do dwóch celów:

  - Modelowania otoczenia systemu (wyznaczenie granicy wokół całego systemu i na wskazaniu leżących poza nią aktorów, którzy wchodzą w interakcję z systemem)

  - Modelowania wymagań stawianych systemowi (określenie, co system powinien robić, niezależnie od tego jak ma to robić.

Diagramy przypadków użycia

- Modelując otoczenie systemu postępuj zgodnie z podanymi wytycznymi:

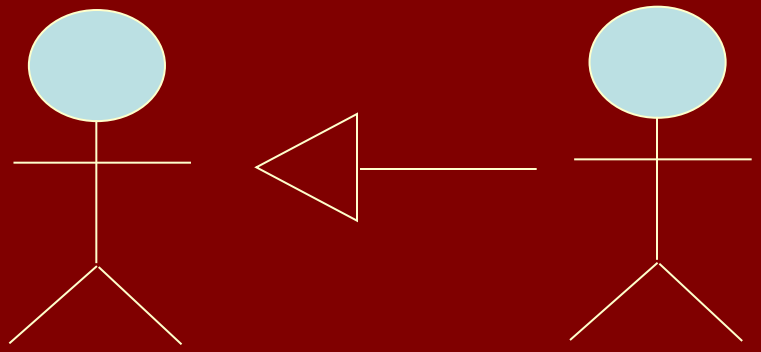
  - zidentyfikuj aktorów działających wokół systemu. Rozważ, które grupy są niezbędne do realizacji funkcji systemu, są w interakcji z urządzeniami zewnętrznymi lub innymi systemami informatycznymi, wypełniają drugorzędne funkcje

  - Uporządkuj podobnych aktorów za pomocą uogólnień

  - Zaludnij tymi aktorami diagram przypadków użycia i zdefiniuj ścieżki komunikacyjne od każdego aktora do przypadków użycia systemu

Diagramu przypadków użycia

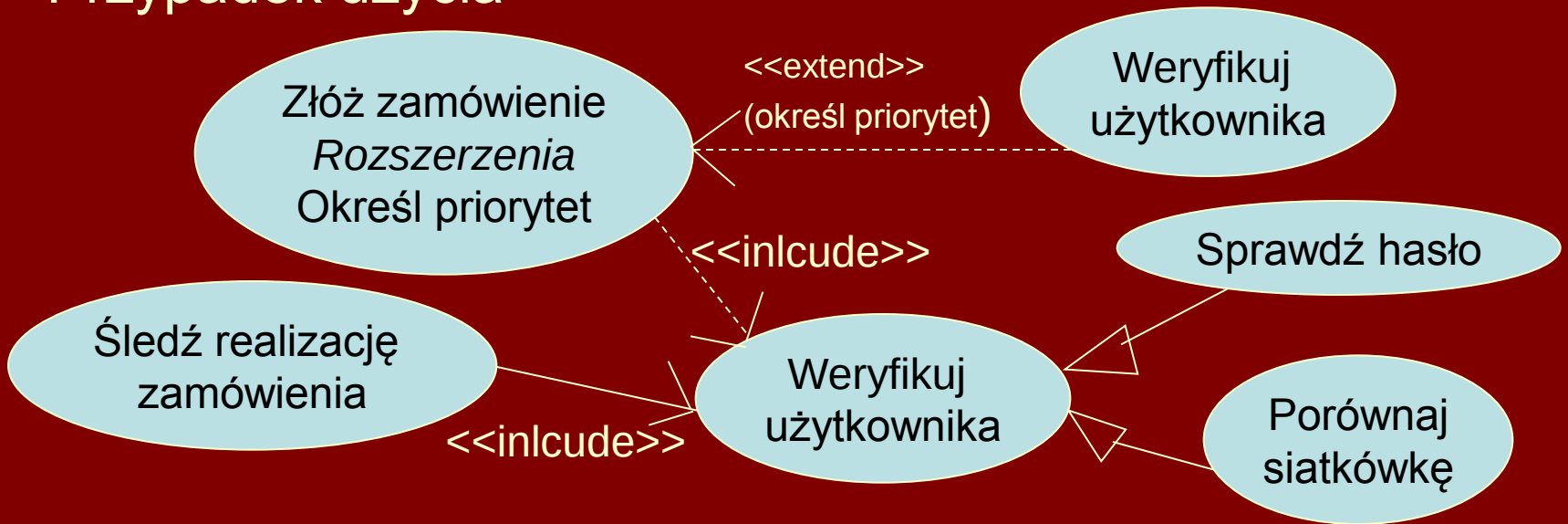
- Aktor:



Klient

Klient indywidualny

- Przypadek użycia



Budowa statycznego modelu klas

- Podejście klasyczne: opiera się na obserwacji, że w wielu systemach pojawiają się podobne klasy i obiekty. Na tej podstawie można poszukiwać podobnych klas w systemie. Typowe klasy to:
- Przedmioty namacalne (samochód, czujnik)
- Role pełnione przez osoby (pracownik, wykładowca)
- Zdarzenia, o których system przechowuje informacje (zamówienie, dostawa)
- Interakcje pomiędzy osobami/systemami (pożyczka, spotkanie)
- Lokalizacje – miejsca
- Grupy przedmiotów namacalnych
- Organizacje (Firma, wydział)
- Koncepcje (miara jakości, zadanie)
- Dokumenty

Analiza funkcji (przypadków użycia)

- Metoda ta opiera się od wyboru pewnej funkcji systemu (niekoniecznie elementarnej). W języku naturalnym opisuje się sposób w jaki system wykonuje tę funkcję. Na tej podstawie tworzy się scenariusz funkcji – przypadek użycia – wprowadzając niezbędne klasy i metody

Weryfikacja klas i obiektów

- Stosowanie tych metod może prowadzić do wykrycia znacznej liczby potencjalnych klas, które nie znajdą się w ostatecznym modelu.
- Należy wziąć pod uwagę następujące czynniki:
  - nieobecność atrybutów i operacji
  - nieliczne atrybuty i operacje
  - tylko jeden obiekt w klasie
  - brak związków z innymi klasami