Unified Modeling Language

Referat na seminarium magisterskie
Zagadnienia Programowania
Obiektowego

Dymitr Pszenicyn

Po co UML ?

- Duże przedsięwzięcia informatyczne wymagają modelowania.
- Istniało wiele metodologii modelowania systemów i trzeba było znać je wszystkie żeby się porozumieć.
- Wszystkie metodologie obiektowe miały wady i zalety – UML połączył ich zalety.

Krótka historia UML

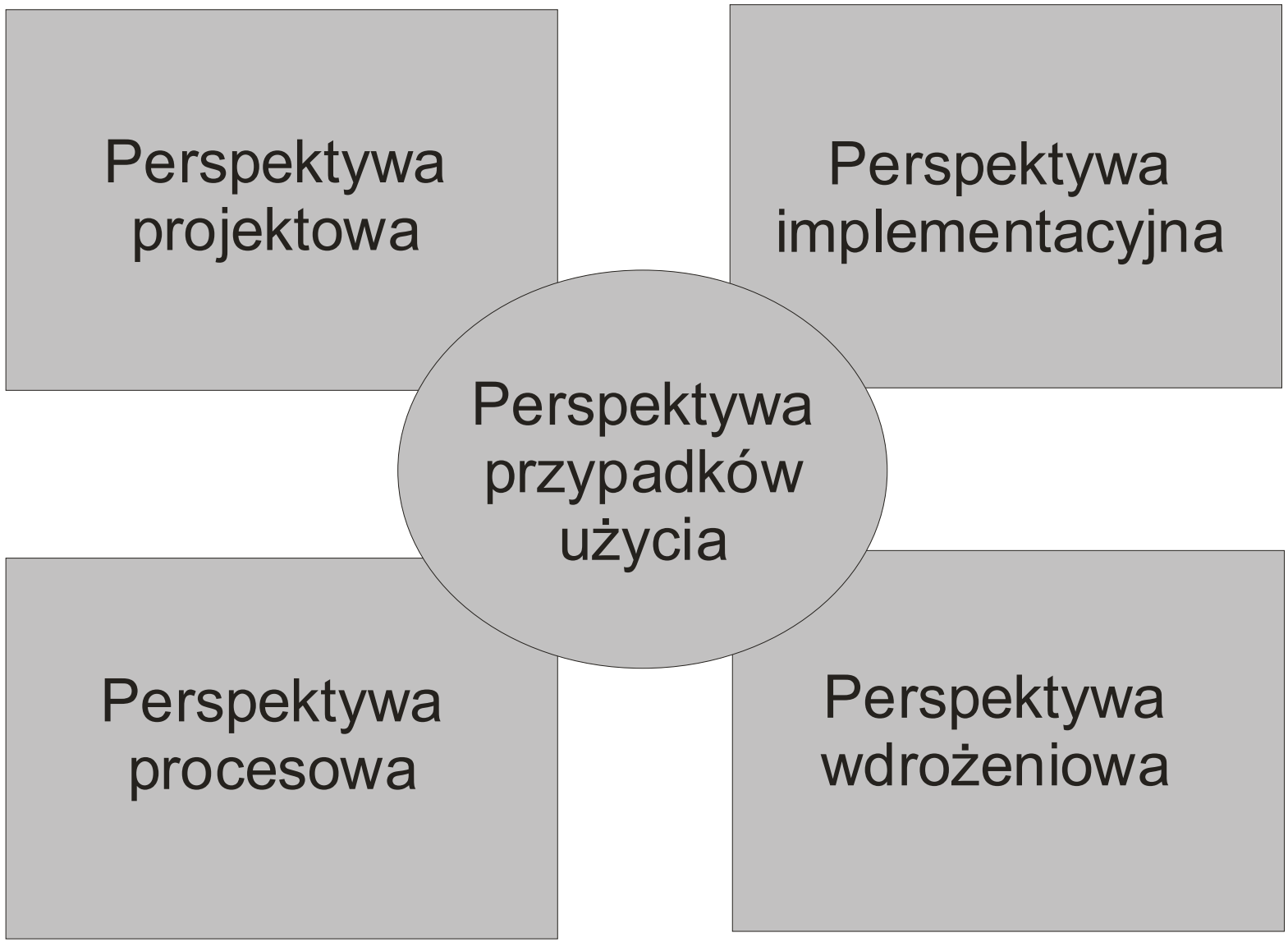
- Grady Booch, Ivar Jacobson i James Rumbaugh w 1995 roku stworzyli Unified Method v0.8
- W 1997 powstał UML v1.0 i został przekazany pod opiekę Object Management Group
- Najnowsza wersja to UML v1.5
- Trwają prace nad UML v2.0

Cechy UML

- Jest przystosowany do opisywania obiektowych systemów.
- Jest niezależny od przyjętego cyklu tworzenia oprogramowania.
- Jest ukierunkowany na przypadki użycia.
- Jest przystosowany do iteracyjnego i przyrostowego rozwijania systemu.
- Umożliwia stosowanie narzędzi do automatycznego tworzenia szkieletu programu z diagramów oraz automatycznego przetwarzania programu na diagramy UML.

Perspektywy

- Każdy duży projekt najlepiej oglądać z wielu prawie niezależnych perspektyw.
- W UML są to:
- perspektywa przypadków użycia
- perspektywa projektowa
- perspektywa procesowa
- perspektywa implementacyjna
- perspektywa wdrożeniowa


Perspektywa projektowa
Perspektywa implementacyjna
Perspektywa przypadków użycia
Perspektywa procesowa
Perspektywa wdrożeniowa

Perspektywa przypadków użycia

- Zachowanie systemu z punktu widzenia użytkowników i analityków.
- Aspekty statyczne wyraża się za pomocą diagramów przypadków użycia.
- Aspekty dynamiczne wyraża się za pomocą diagramów interakcji, diagramów stanów i diagramów czynności.

Perspektywa projektowa

- Przede wszystkim budowa systemu z klas i interfejsów.
- Aspekty statyczne wyraża się za pomocą diagramów klas i diagramów obiektów.
- Aspekty dynamiczne wyraża się za pomocą diagramów interakcji, diagramów stanów i diagramów czynności.

Perspektywa procesowa

- Przede wszystkim budowa systemu z wątków i procesów.
- Aspekty statyczne wyraża się za pomocą diagramów klas i diagramów obiektów.
- Aspekty dynamiczne wyraża się za pomocą diagramów interakcji, diagramów stanów i diagramów czynności.
- Bardzo ważną rolę na diagramach pełnią klasy aktywne które reprezentują procesy i wątki.

Perspektywa implementacyjna

- Przede wszystkim fizyczna budowa systemu z plików i komponentów.
- Aspekty statyczne wyraża się za pomocą diagramów komponentów.
- Aspekty dynamiczne wyraża się za pomocą diagramów interakcji, diagramów stanów i diagramów czynności.

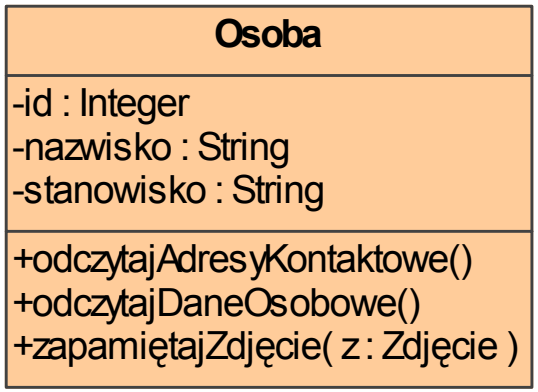
Perspektywa wdrożeniowa

- Przede wszystkim węzły na których będzie uruchamian system.
- Aspekty statyczne wyraża się za pomocą diagramów instalacji.
- Aspekty dynamiczne wyraża się za pomocą diagramów interakcji, diagramów stanów i diagramów czynności.

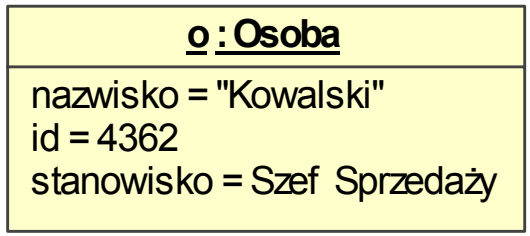
Elementy UML

- Diagramy UML składają się z wielu elementów takich jak klasy, obiekty, komponenty, pakiety i powiązania między nimi.
- Elementy posiadają wiele atrybutów ale tylko nazwa jest wymagana.

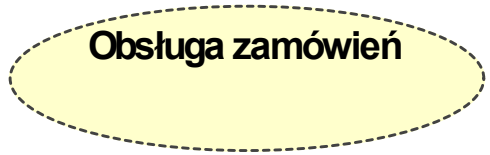
Klasa:



Obiekt:

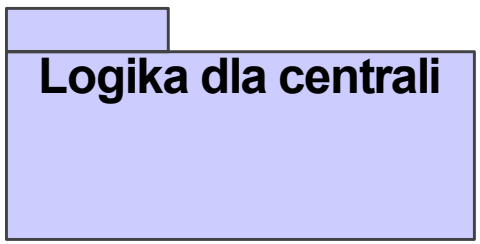


Kooperacja:

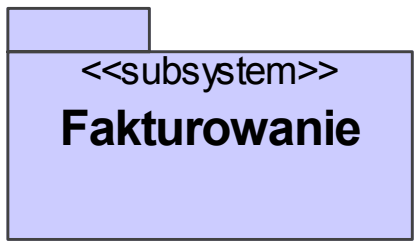


Interfejs:

Pakiet:

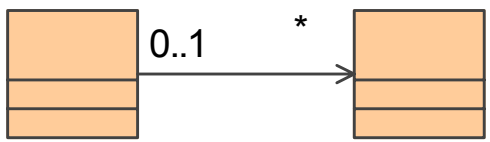


Podsystem:

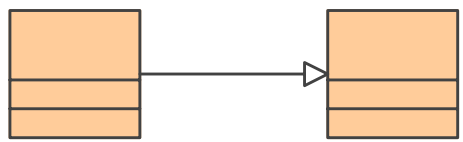


Zależność:

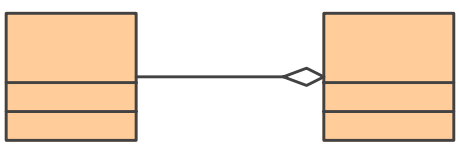
Powiązanie:



Uogólnienie:

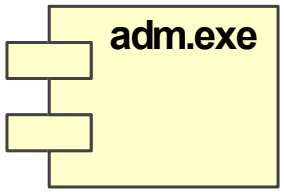


Agregacja:

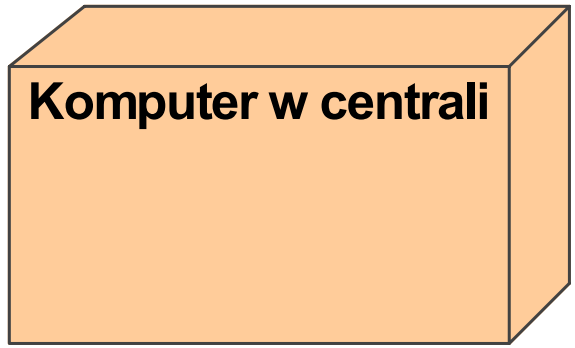


Agregacja całkowita:

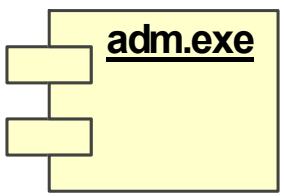
Komponent:



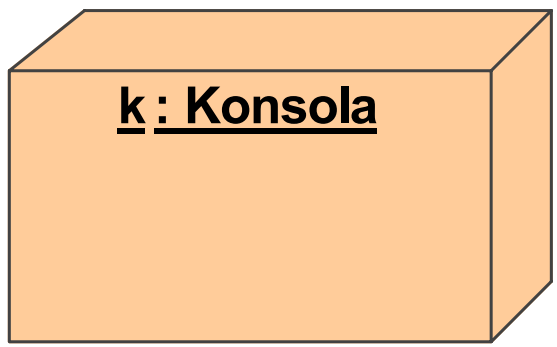
Węzeł:



Instancja komponentu:

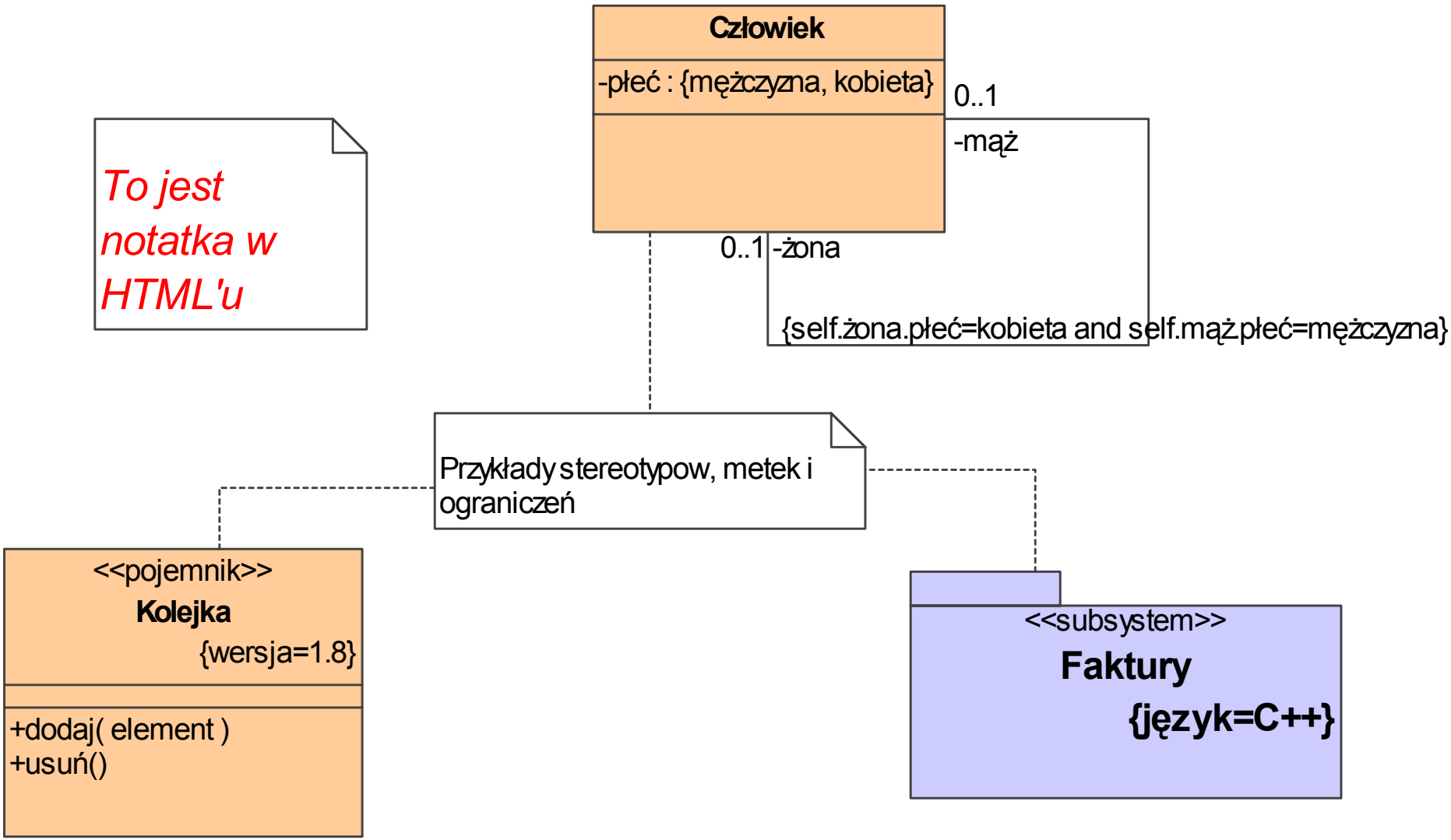


Instancja węzła:



UML jest rozszerzalny

- Notatki pozwalają na dopisywanie komentarzy, ograniczeń i wymagań.
- Stereotypy pozwalają na tworzenie nowych bloków konstrukcyjnych.
- Metki umożliwaiją rozszerzenie listy właściwości bloku konstrukcyjnego.
- Ograniczenia umożliwiają rozszerzanie znaczenia bloku konstrukcyjnego.


Człowiek
-płeć : {mężczyzna, kobieta}
0..1
-mąż
0..1
-żona
{self.żona.płeć=kobieta and self.mąż.płeć=mężczyzna}
To jest notatka w HTML'u
Przykłady stereotypow, metek i ograniczeń
<<pojemnik>>
Kolejka
{wersja=1.8}
+dodaj(element)
+usuń()
<<subsystem>>
Faktury
{język=C++}

Z czego składa się UML

- 12 rodzajów diagramów podzielonych na 3 kategorie:
- 4 diagramy pokazują statyczną strukturę aplikacji
- 5 diagramów pokazuje dynamiczne zachowania systemu
- 3 diagramy pokazują organizację modelu

Statyczna struktura systemu (Structural Diagrams)

Diagram klas (Class Diagram)

- Pokazuje związki (uogólnienie, zależność, powiązanie i agregacja) pomiędzy klasami oraz interfejsami.
- Klasy na diagramie mogą być grupowane w pakiety.
- Klasy aktywne (w grubej ramce) reprezentują wątki i procesy.
- Nazwy klas abstrakcyjnych są pisane kursywą.

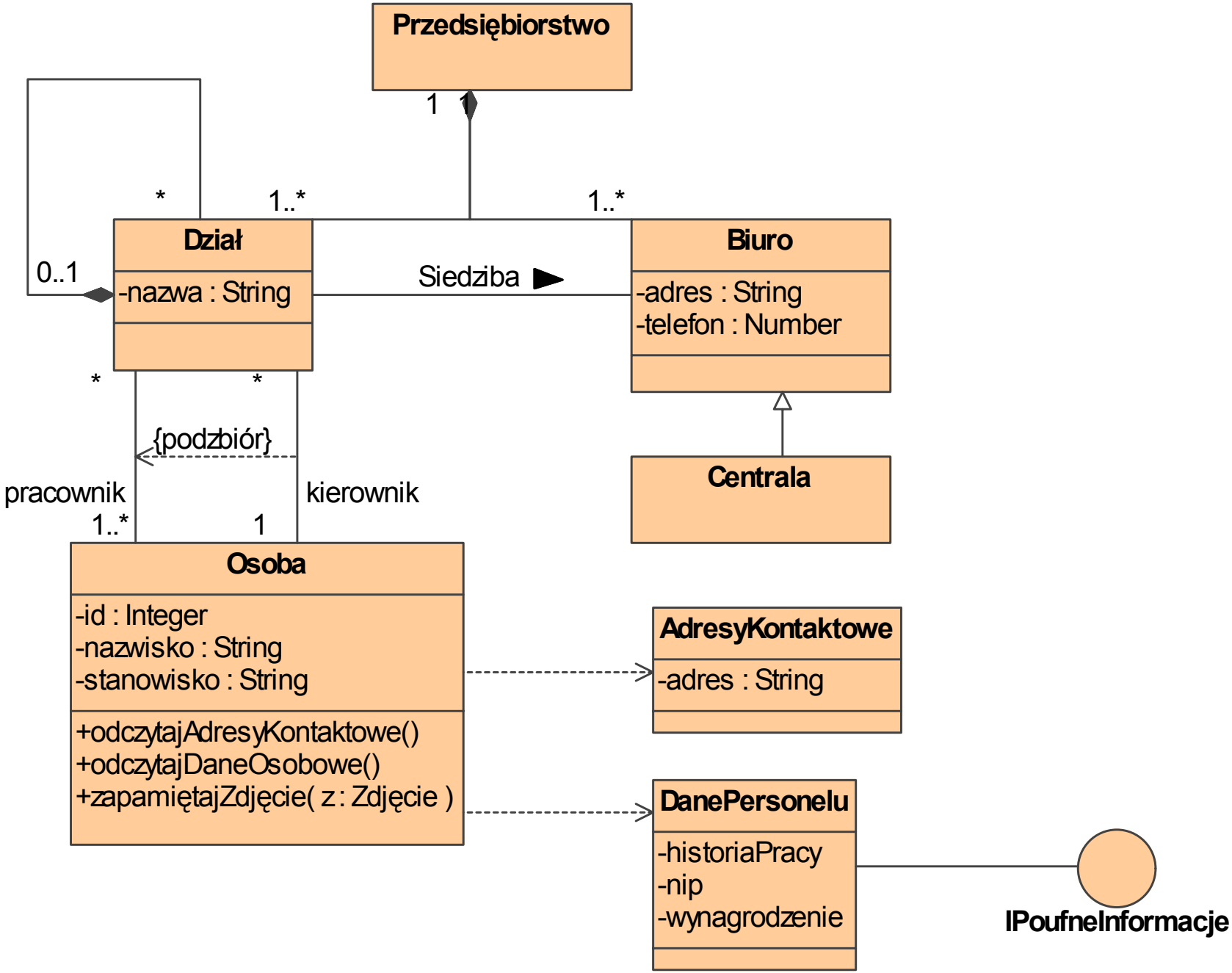

Przedsiębiorstwo
1
1
*
1..*
1..*
Dział
-nazwa : String
0..1
Siedziba
Biuro
-adres : String
-telefon : Number
*
*
{podzbiór}
Centrala
pracownik
kierownik
1..*
1
Osoba
-id : Integer
-nazwisko : String
-stanowisko : String
+odczytajAdresyKontaktowe()
+odczytajDaneOsobowe()
+zapamiętajZdjęcie(z : Zdjęcie)
AdresyKontaktowe
-adres : String
DanePersonelu
-historiaPracy
-nip
-wynagrodzenie
IPoufneInformacje

Diagram obiektów (Object Diagram)

- Pokazuje związki między obiektami oraz klasami.
- Przydatny do przedstawiania przykładów (rzut systemu w danej chwili) i analizowania działającego systemu.
- Nie powinien przedstawiać pełnego zbioru obiektów w systemie, a tylko interesujący nas wycinek.

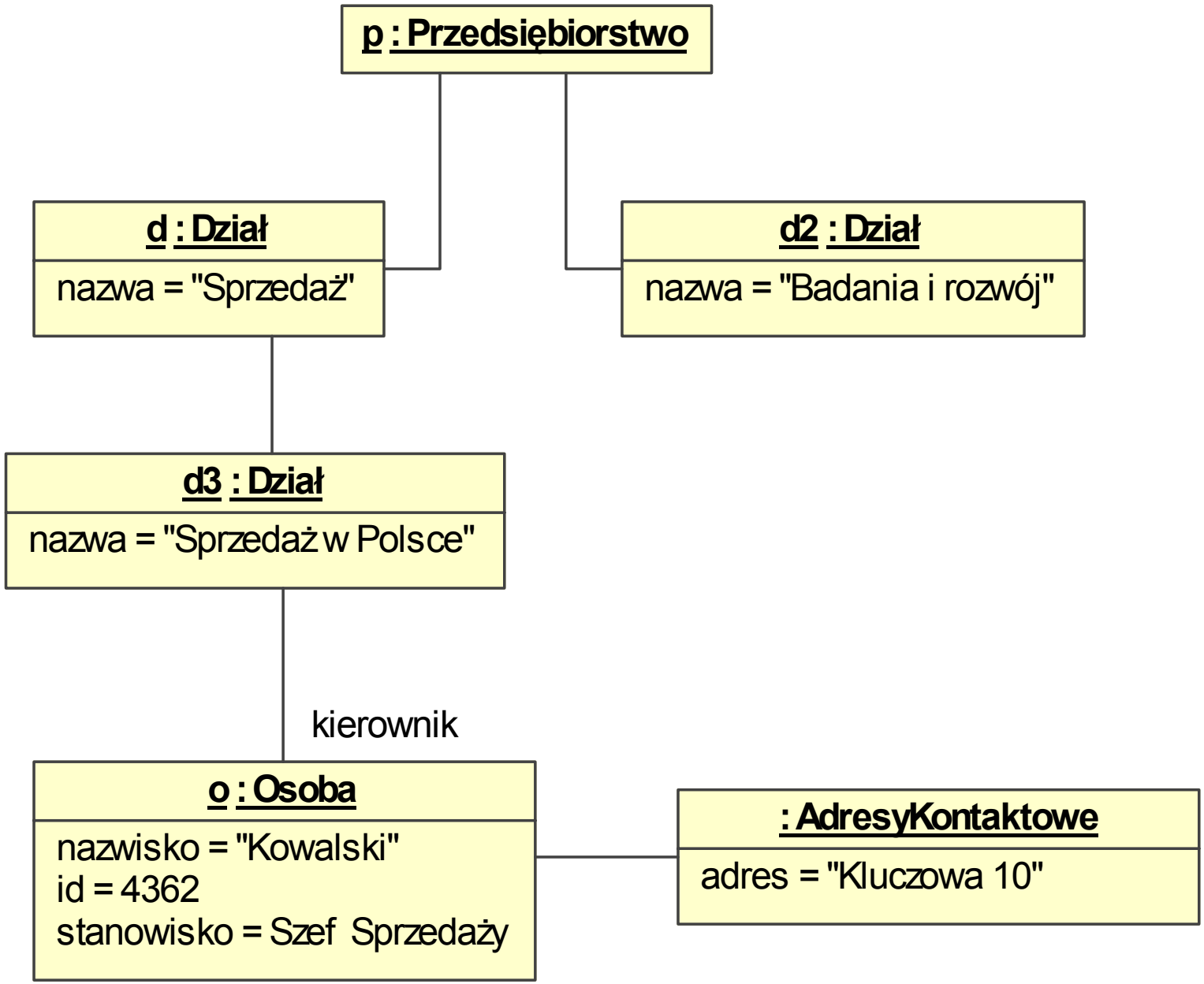

p : Przedsiębiorstwo
d : Dział
nazwa = "Sprzedaż"
d2 : Dział
nazwa = "Badania i rozwój"
d3 : Dział
nazwa = "Sprzedaż w Polsce"
kierownik
o : Osoba
nazwisko = "Kowalski"
id = 4362
stanowisko = Szef Sprzedaży
: AdresyKontaktowe
adres = "Kluczowa 10"

Diagram komponentów (Component Diagram)

- Obrazuje fizyczne, wymienne komponenty systemu, takie jak: pliki, tabele, biblioteki obiektów.
- Bardzo często łączony z diagramem instalacji, pokazuje co gdzie zainstalować.
- Definiowanie interfejsów zapewnia wymienialność komponentów.

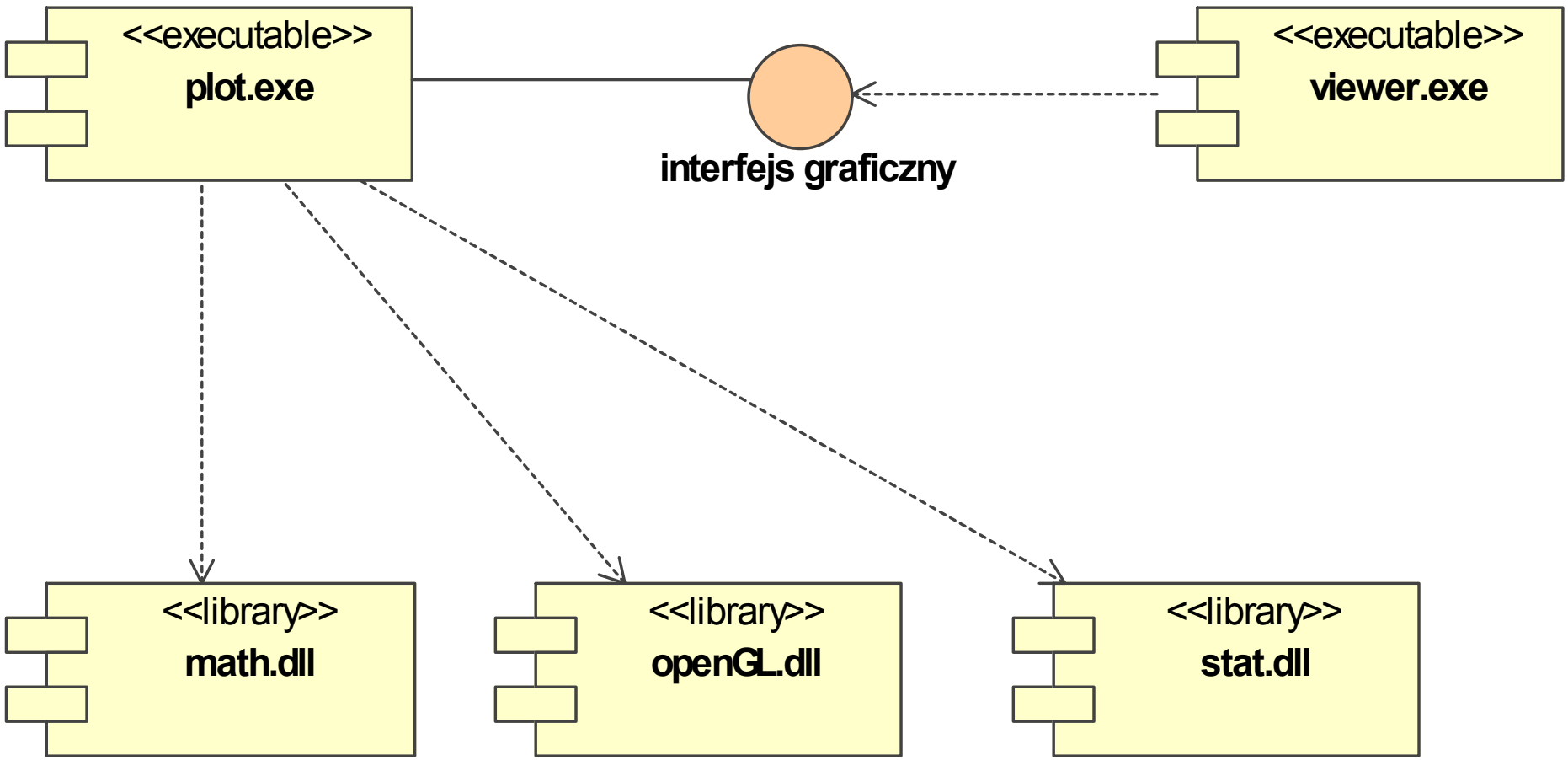
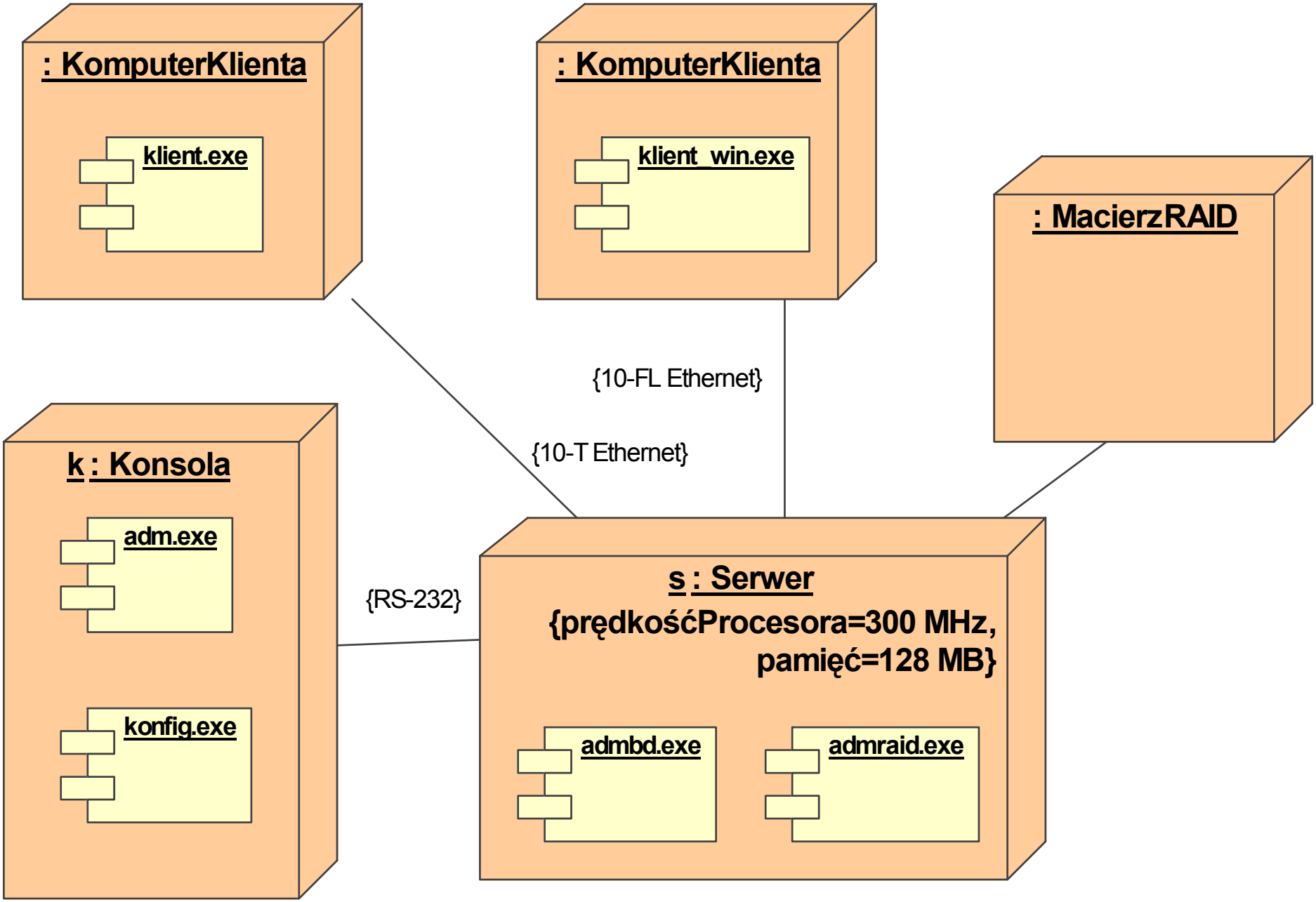
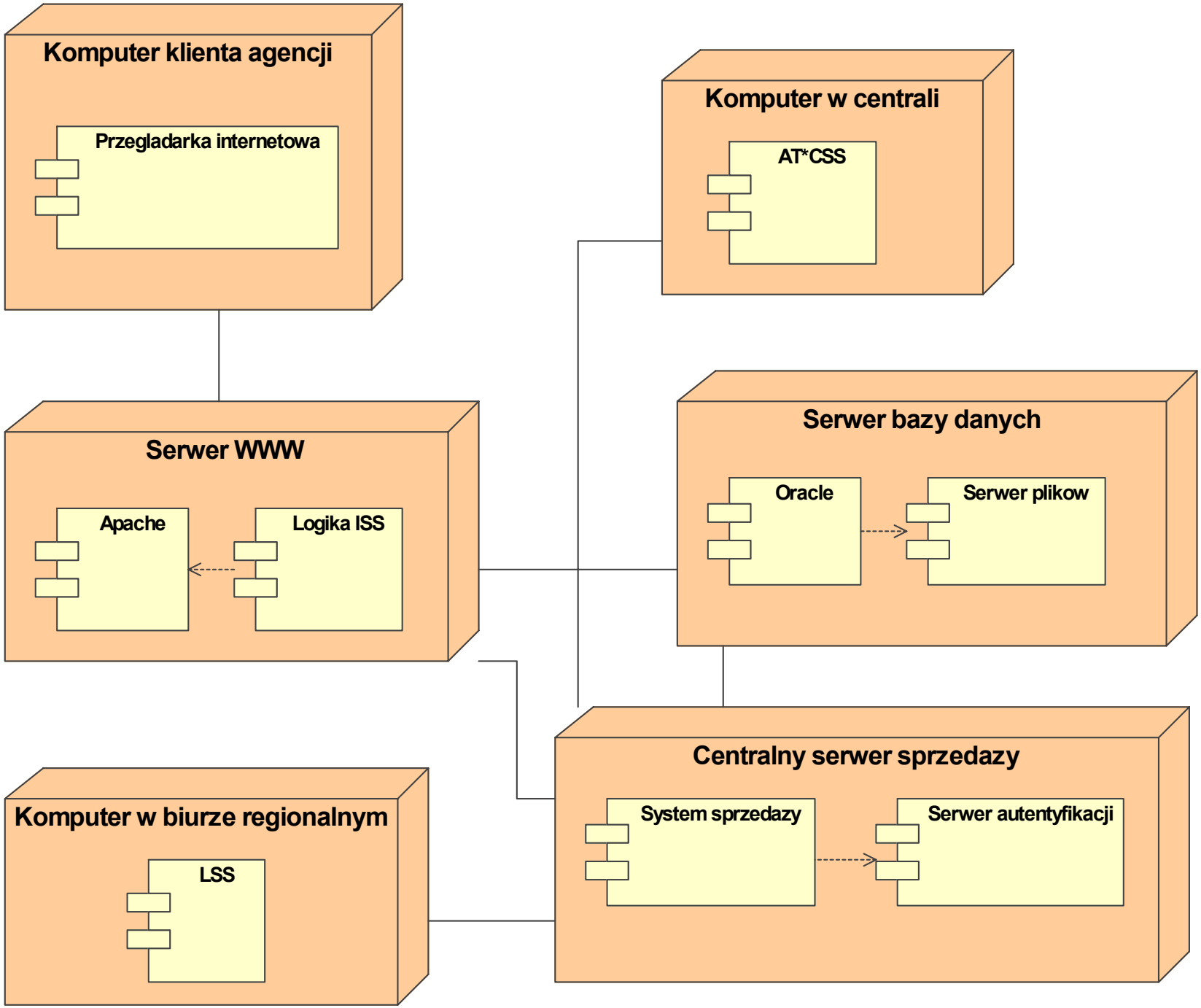

<<executable>>
plot.exe
interfejs graficzny
<<executable>>
viewer.exe
<<library>>
math.dll
<<library>>
openGL.dll
<<library>>
stat.dll

Diagram instalacji (Deployment Diagram)

- „Trójwymiarowe” bloki (tzw. węzły) reprezentują sprzęt, na którym ma działać system (serwery, urządzenia sieciowe itd).
- Prawie zawsze zawierają komponenty.
- Często są stereotypowane za pomocą specjalnych symboli graficznych (komputer albo cała sieć).

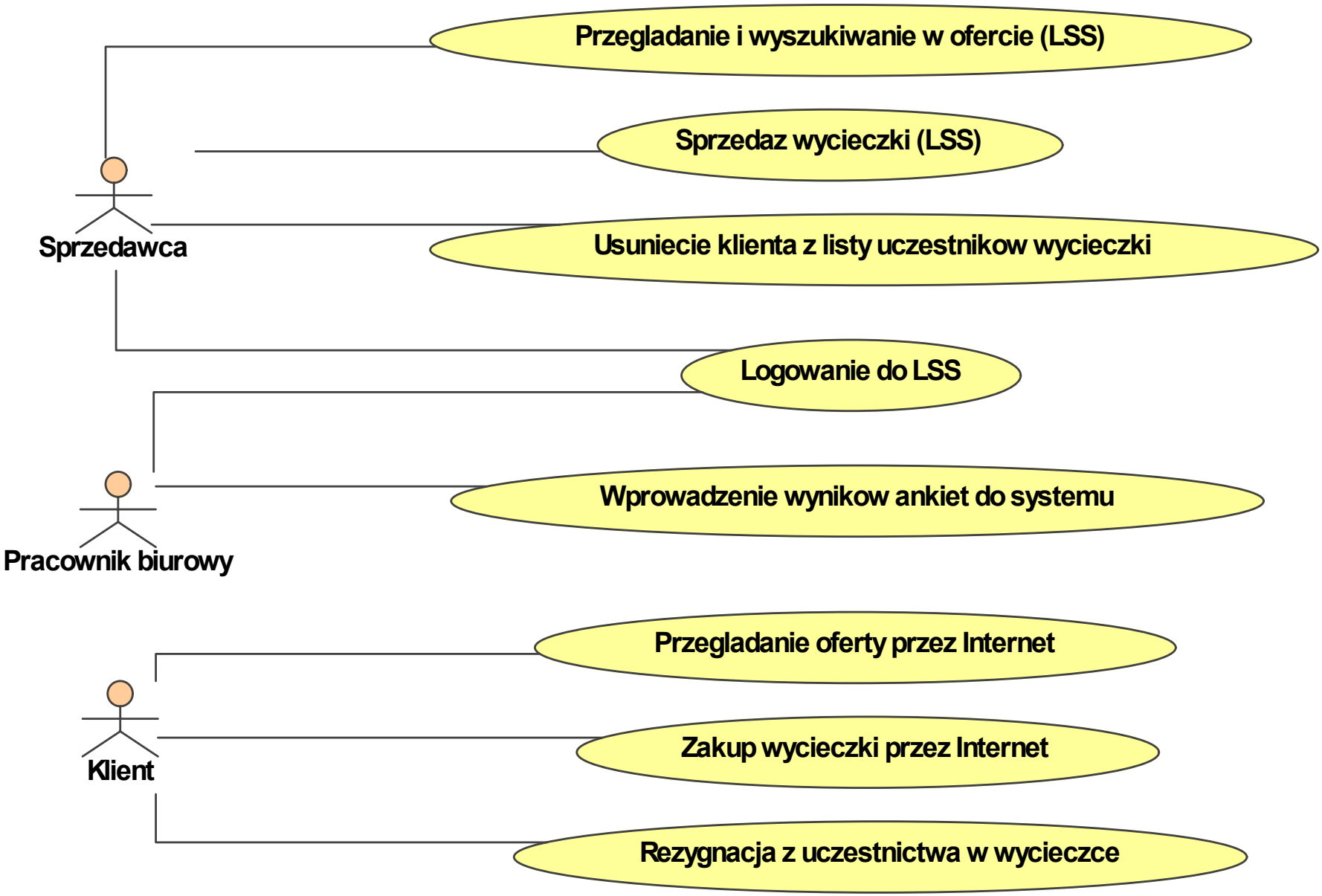

: KomputerKlienta
klient.exe
: KomputerKlienta
klient_win.exe
: MacierzRAID
{10-FL Ethernet}
{10-T Ethernet}
k : Konsola
adm.exe
konfig.exe
{RS-232}
s : Serwer
{prędkośćProcesora=300 MHz,
pamięć=128 MB}
admbd.exe
admraid.exe

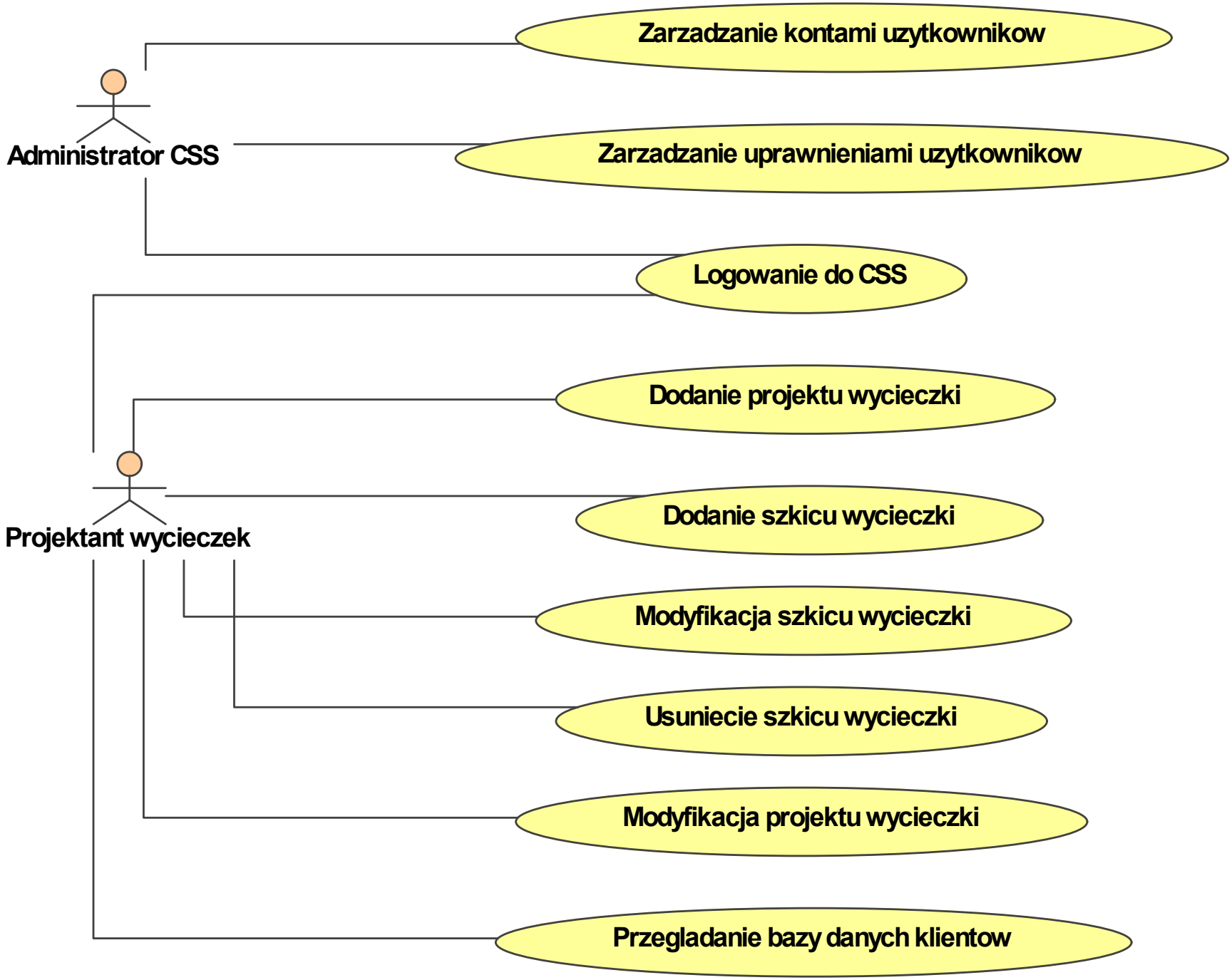

Komputer klienta agencji
Przegladarka internetowa
Komputer w centrali
AT*CSS
Serwer WWW
Apache
Logika ISS
Serwer bazy danych
Oracle
Serwer plikow
Komputer w biurze regionalnym
LSS
Centralny serwer sprzedazy
System sprzedazy
Serwer autentyfikacji

Dynamiczne zachowania systemu (Behavior Diagrams)

Diagram przypadków użycia (Use Case Diagram)

- Zawierają aktorów, przypadki użycia i związki pomiędzy nimi.
- Pozwalają prześledzić możliwości systemu – są bardzo przydatne dla klienta który może zorientować się co naprawdę system będzie robić.
- Są bardziej statyczne od diagramów przebiegu lub czynności.


Przegladanie i wyszukiwanie w ofercie (LSS)
Sprzedaz wycieczki (LSS)
Sprzedawca
Usuniecie klienta z listy uczestnikow wycieczki
Logowanie do LSS
Wprowadzenie wynikow ankiet do systemu
Pracownik biurowy
Przegladanie oferty przez Internet
Zakup wycieczki przez Internet
Klient
Rezygnacja z uczestnictwa w wycieczce


Zarzadzanie kontami uzytkownikow
Administrator CSS
Zarzadzanie uprawnieniami uzytkownikow
Logowanie do CSS
Dodanie projektu wycieczki
Projektant wycieczek
Dodanie szkicu wycieczki
Modyfikacja szkicu wycieczki
Usuniecie szkicu wycieczki
Modyfikacja projektu wycieczki
Przegladanie bazy danych klientow

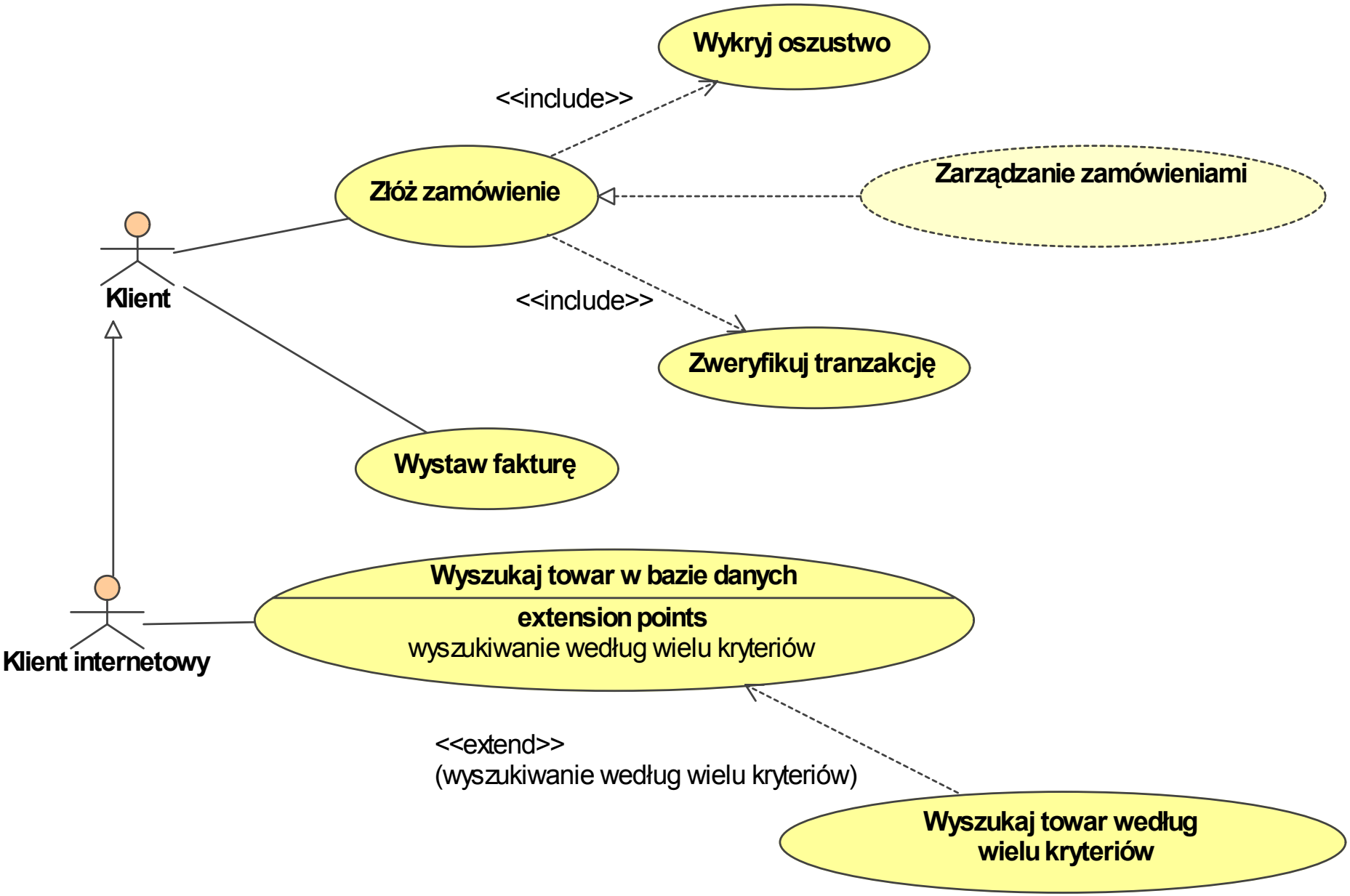

Wykryj oszustwo
<<include>>
Złóż zamówienie
Zarządzanie zamówieniami
Klient
<<include>>
Zweryfikuj tranzakcję
Wystaw fakturę
Wyszukaj towar w bazie danych
extension points
wyszukiwanie według wielu kryteriów
Klient internetowy
<<extend>>
(wyszukiwanie według wielu kryteriów)
Wyszukaj towar według
wielu kryteriów

Diagram przebiegu (Sequence Diagram)

- Ułożenie linii na diagramie jest istotne, gdyż oś Y oznacza czas.
- Diagram przebiegu uwypukla kolejność komunikatów w czasie.
- Jeden diagram powinien przedstawiać tylko jedną sytuację (jeden przepływ sterowania).
- Dla pokazania prostych wariantów można używać iteracji i rozgałęzień.

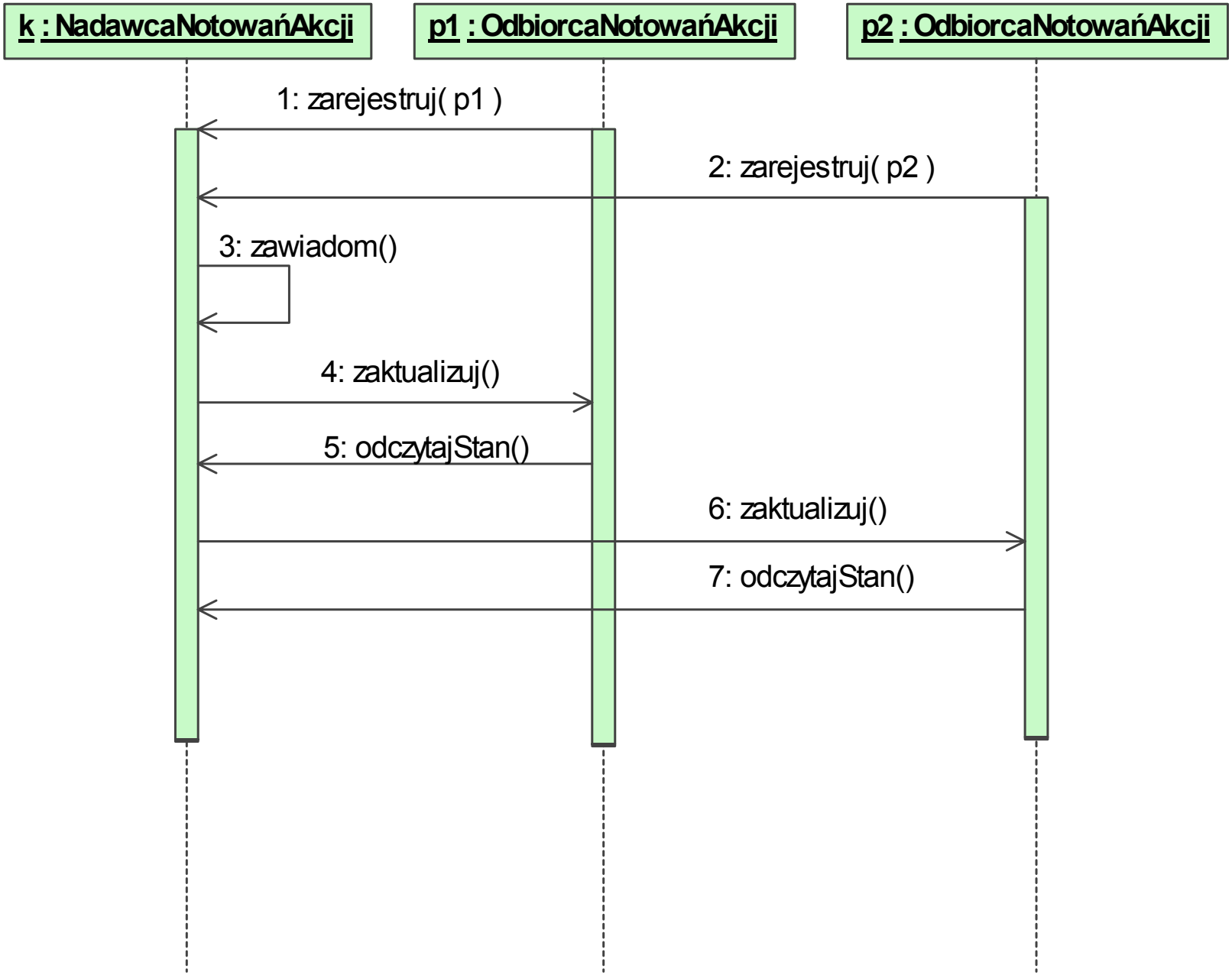

k : NadawcaNotowańAkcji
p1 : OdbiorcaNotowańAkcji
p2 : OdbiorcaNotowańAkcji
1: zarejestruj(p1)
2: zarejestruj(p2)
3: zawiadom()
4: zaktualizuj()
5: odczytajStan()
6: zaktualizuj()
7: odczytajStan()

Diagram współpracy (Collaboration Diagram)

- Ułożenie obiektów względem siebie nie jest istotne.
- Diagram współpracy uwypukla organizację obiektów uczestniczących w interakcji.
- Komunikaty muszą być numerowane (zgodnie z notacją Deweya).
- Dla pokazania prostych wariantów można używać iteracji i rozgałęzień.

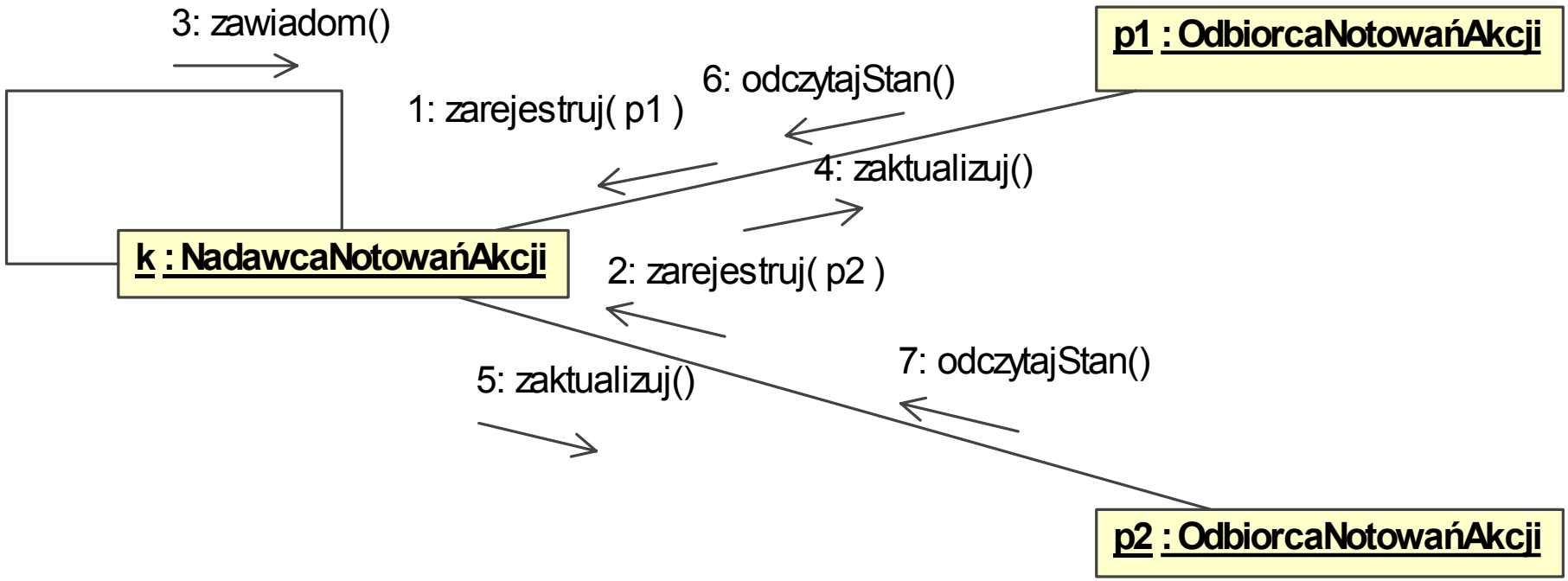

3: zawiadom()
p1 : OdbiorcaNotowańAkcji
6: odczytajStan()
1: zarejestruj(p1)
4: zaktualizuj()
k : NadawcaNotowańAkcji
2: zarejestruj(p2)
7: odczytajStan()
5: zaktualizuj()
p2 : OdbiorcaNotowańAkcji

Diagram czynności (Activity Diagram)

- Jest to schemat blokowy, który przedstawia przepływ sterowania pomiędzy czynnościami.
- Często zawieraja tory, które pomagają grupować stany czynności oraz reprezentują jednostkę odpowiedzialną za przydzielone czynności.
- Dodatkowo można przedstawić przepływ obiektów oraz zmianę ich stanu.
- Można użyć diagramu czynności do rozpisania pojedyńczej operacji ale zwykle jest to bardzo podobne do kodu źródłowego.

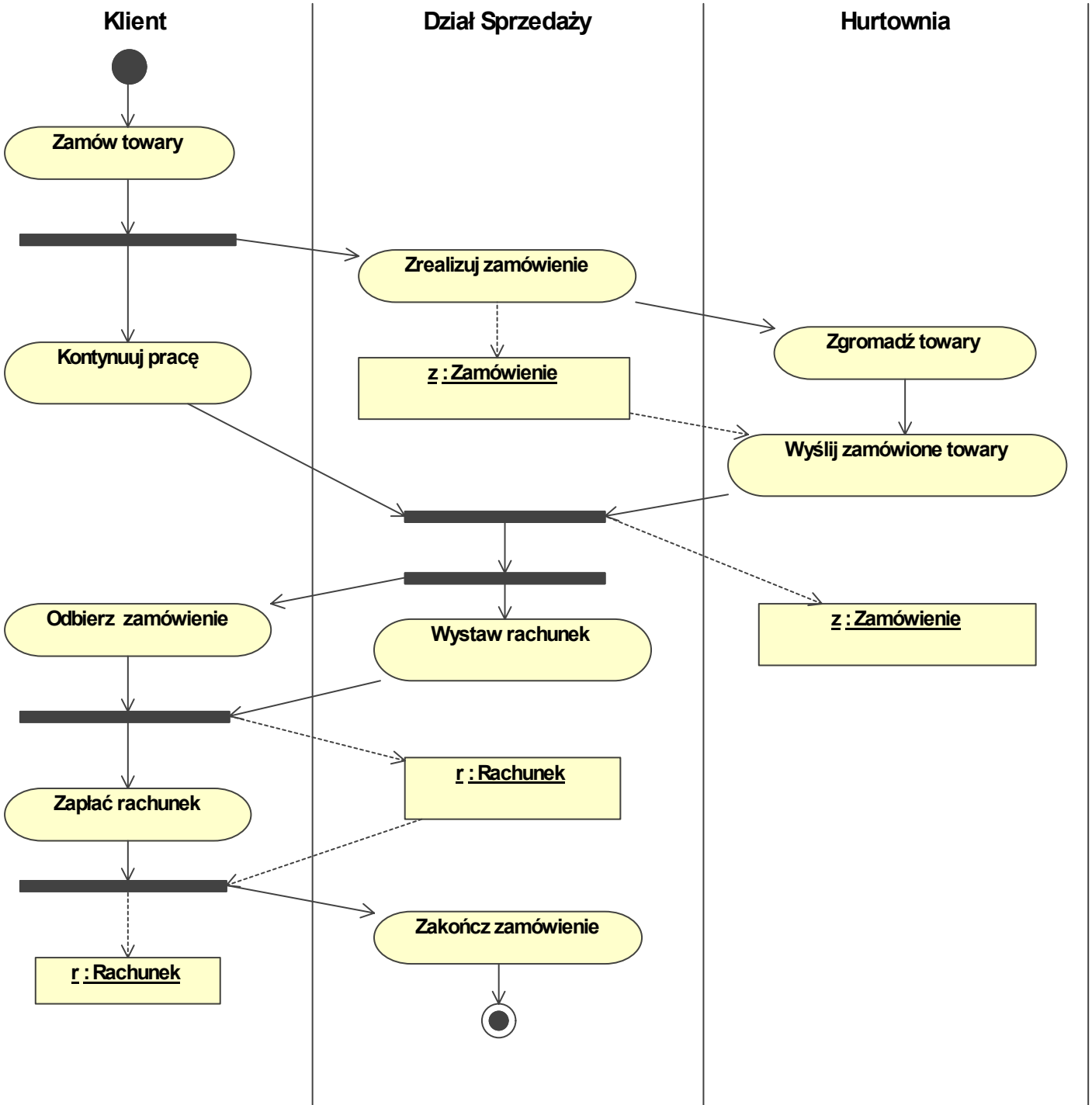
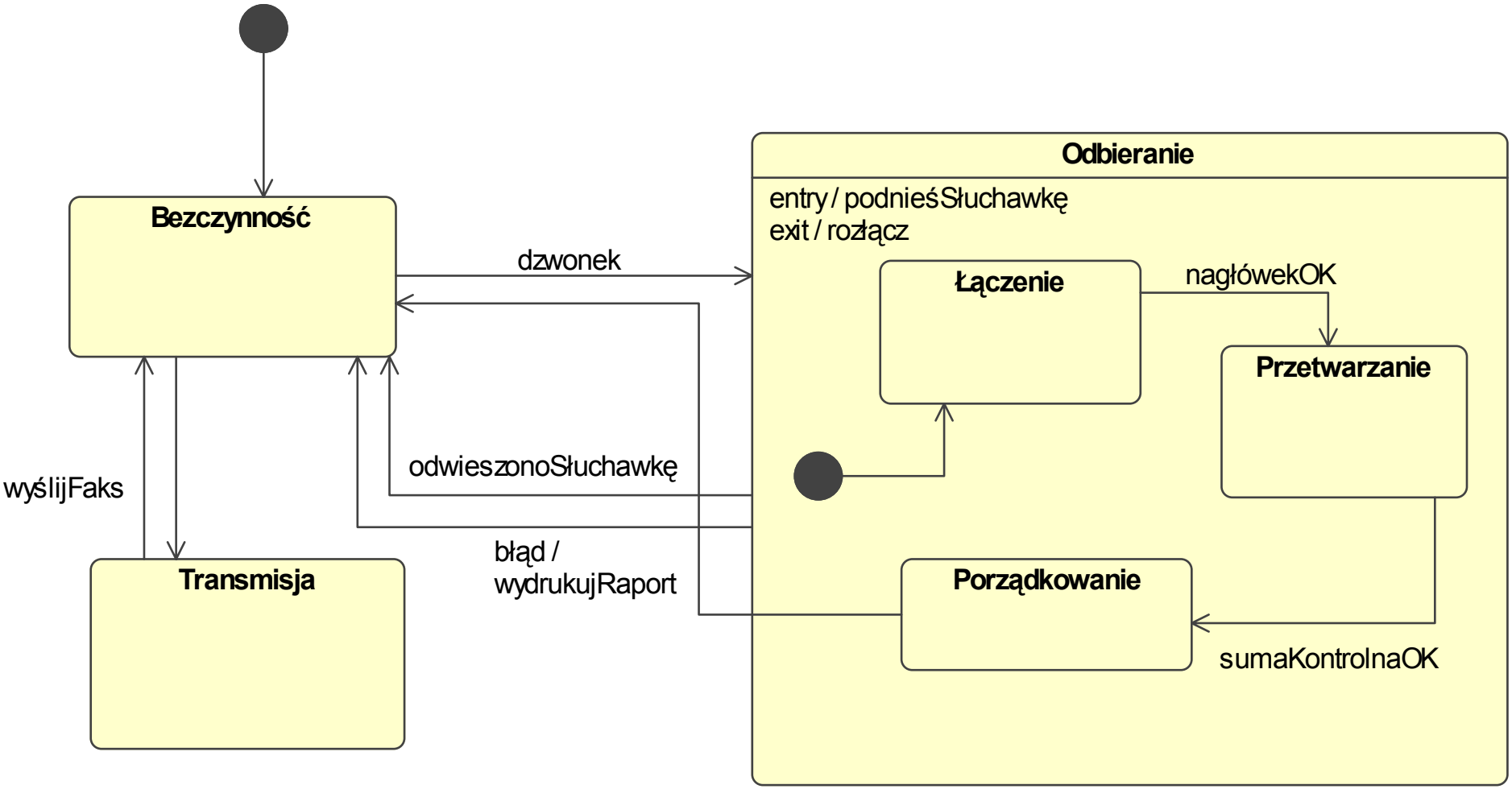

Klient
Dział Sprzedaży
Hurtownia
Zamów towary
Zrealizuj zamówienie
Zgromadź towary
Kontynuuj pracę
z : Zamówienie
Wyślij zamówione towary
Odbierz zamówienie
Wystaw rachunek
z : Zamówienie
r : Rachunek
Zapłać rachunek
Zakończ zamówienie
r : Rachunek

Diagram stanów (Statechart Diagram)

- Służy do opisu stanów pojedyńczego elementu (klasy, przypadku użycia, systemu).
- Można podać akcję wejściową i wyjściową.
- Stany mogą być zagnieżdżane.
- Można tworzyć podstany współbierzne (wtedy oba podstany muszą dojść do stanu końcowego zanim przepływy sterowania się połączą).
- Istnieją stany wznowienia, które służą do zapamiętania ostatnio przyjętego podstanu.


Bezczynność
dzwonek
Odbieranie
entry / podnieśSłuchawkę
exit / rozłącz
Łączenie
nagłówekOK
Przetwarzanie
odwieszonoSłuchawkę
wyślijFaks
błąd /
wydrukujRaport
Transmisja
Porządkowanie
sumaKontrolnaOK

Organizacja modelu
(Model Management Diagrams)

Diagram pakietów (Packages Diagram)

- Pakiety służą do grupowania bytów.
- Składnikami pakietu mogą być klasy, interfejsy, komponenty, węzły, operacje, przypadki użycia, diagramy i inne pakiety. W związku z tym, pakiety mogą wystąpić na prawie każdym diagramie.
- Każdy pakiet ma własną przestrzeń nazw.
- Na wszystkich diagramach, gdy używa się nazwy bytu (np. klasy), można podać pełną nazwę ścieżkowa (rozdzieloną podwójnymi dwukropkami ::).

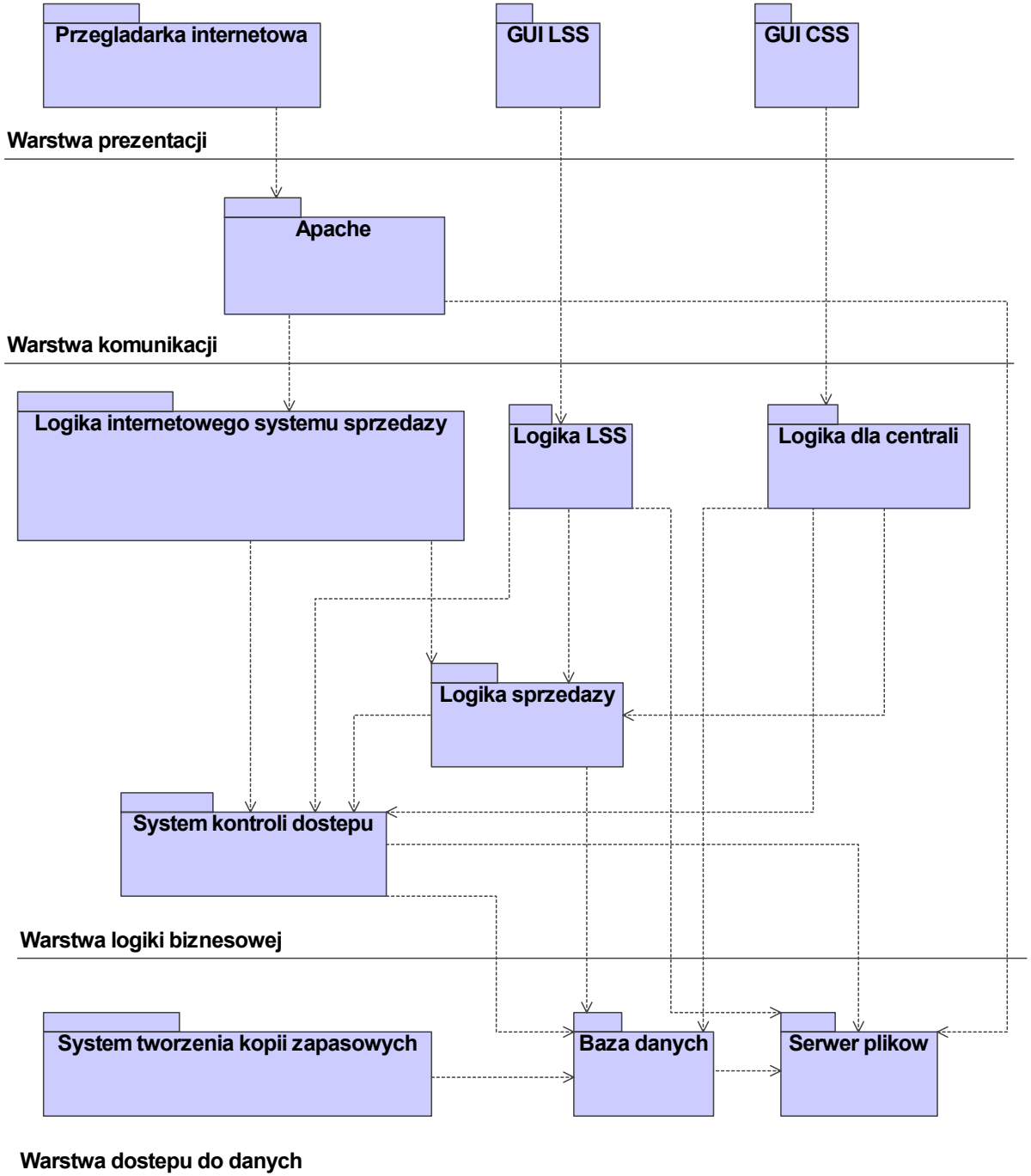
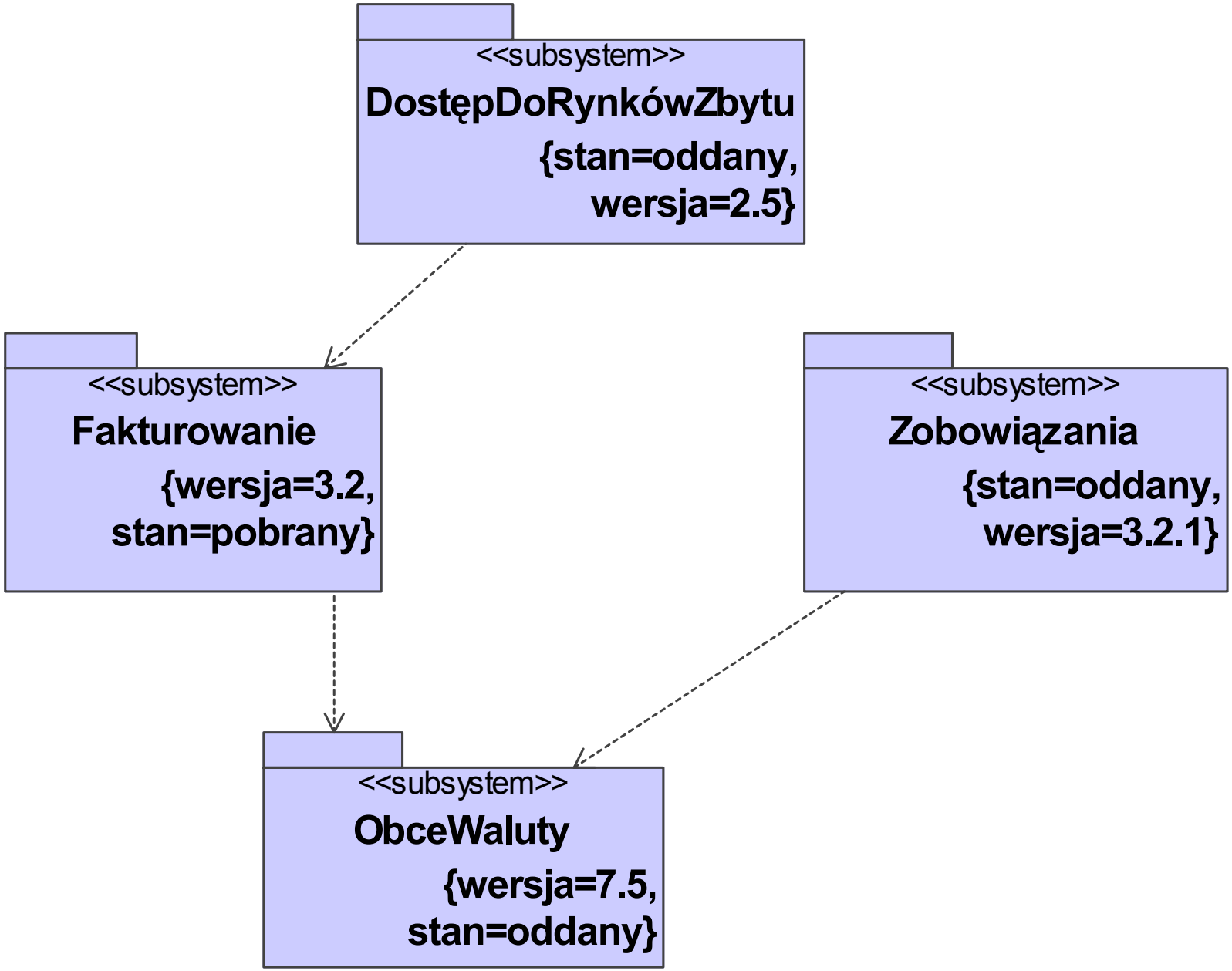

Przegladarka internetowa
GUI LSS
GUI CSS
Warstwa prezentacji
Apache
Warstwa komunikacji
Logika internetowego systemu sprzedazy
Logika LSS
Logika dla centrali
Logika sprzedazy
System kontroli dostepu
Warstwa logiki biznesowej
System tworzenia kopii zapasowych
Baza danych
Serwer plikow
Warstwa dostepu do danych

Diagram Podsystemów (Subsystems Diagram)

- Podsystemy służą do grupowania pakietów.
- Są przydatne tylko w dużych systemach.
- Cały system jest zwykle agregacją podsystemów.
- Między podsystemami może zachodzić uogólnienie.


<<subsystem>>
DostępDoRynkówZbytu
{stan=oddany,
wersja=2.5}
<<subsystem>>
Fakturowanie
{wersja=3.2,
stan=pobrany}
<<subsystem>>
Zobowiązania
{stan=oddany,
wersja=3.2.1}
<<subsystem>>
ObceWaluty
{wersja=7.5,
stan=oddany}

Narzędzia do UML

- MagicDraw http://www.magicdraw.com/
- ArgoUML http://argouml.tigris.org/
- Poseidon http://www.gentleware.com/

Źródła przedstawionych diagramów

- stworzone na prezentację (Dymitr Pszenicyn)
- z książki „UML – przewodnik użytkownika” (Grady Booch, James Rumbaugh, Ivar Jacobson)
- z projektu na laboratorium Inżynieria Oprogramowania (Przemysław Rekucki, Dymitr Pszenicyn, Aleksander Grygiel, Andrzej Mizera, Krzysztof Kowalczyk)
- z projektu na Zespołowy Projekt Programistyczny (Aleksander Grygiel, Dymitr Pszenicyn, Stanisław Skonieczny, Andrzej Awramiuk)

Publikacje o UML

- G. Booch, J. Rumbaugh, I. Jacobson, „UML – przewodnik użytkownika”. WNT, Warszawa 2002
- Formalna specyfikacja UML http://www.omg.org/cgi-bin/doc?formal/03-03-01
- Artykuł o historii UML http://www.omg.org/news/pr99/UML_2001_CACM_Oct99_p29-Kobryn.pdf
- Tutorial UML http://www.smartdraw.com/resources/centers/uml/uml.htm

Publikacje o UML (cd.)

- Artykuł o UML
  http://www.rational.com/media/uml/intro_rdn.pdf
- Spis artykułów o UML
  http://www.rational.com/uml/resources/whitepapers/index.jsp