

Systemy rozproszone

dr inż. Jacek Czerniak

jczerniak@ukw.edu.pl

Plan wykładu:

- Wprowadzenie
- Architektury systemów rozproszonych
- Agenci programowi
- Wielowątkowość
- Przetwarzanie równoległe
- CORBA/RMI
- Klastry
- Chmury obliczeniowe

Forma zaliczenia – ZO

8h W i 10 h lab.

- Test „20”
 - 20 – pytań
 - 20 min.
 - 60% zalicza (12 z 20 punktów)

Literatura

1. J. Cooling, Software Engineering for Real-Time Systems, Addison-Wesley, Harlow 2003.
2. J. Górski, Inżynieria oprogramowania w projekcie informatycznym, Mikom, Warszawa 1999.
3. J. Roszkowski, Analiza i projektowanie strukturalne, Helion, Warszawa 1998.
4. R. Barker, C. Longman, Case Method. Modelowanie funkcji i procesów, WNT, Warszawa 1996.
5. Z. Martyniak, Elementy zarządzania informacją i komunikacją w przedsiębiorstwie, AE w Krakowie 1997
6. J. Kisielnicki, H. Sroka, Systemy informacyjne biznesu. Informatyka dla zarządzania, Placet, Warszawa 1999
7. J. Davidson, Kierowanie projektem. Praktyczny poradnik dla tych, którzy nie lubią tracić czasu. Wyd. Liber, Warszawa 2002.
8. Flakiewicz W., Systemy informacyjne w zarządzaniu. Uwarunkowania, technologie, rodzaje, Wydawnictwo C. H. Beck, Warszawa 2002
9. Jabnoun N., Sahraoui S., Enabling a TQM structure through information technology, Competitiveness Review, 1-2/2004, American Society for Competitiveness, Pittsburg 2004

System informacyjny a informatyczny?

Przegląd definicji:

- ✓ Informacja
- ✓ komunikacja
- ✓ system informacyjny
- ✓ system informatyczny

System informacyjny a informatyczny?

INFORMACJA

Proces w wyniku którego jest się poinformowany.

Rezultat jakiegoś procesu informacyjnego.

KOMUNIKACJA

Proces wymiany informacji pomiędzy nadawcą a odbiorcą, przy czym w wymianie tej zachodzi sprzężenie zwrotne.

System informacyjny

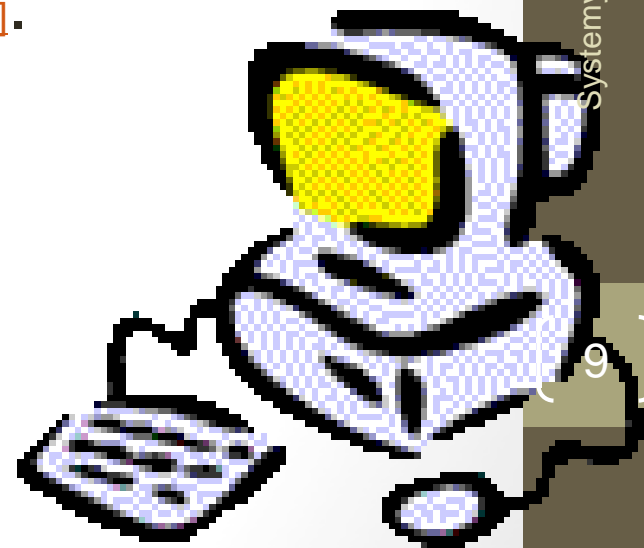
System informacyjny można określić jako posiadającą wiele poziomów strukturę pozwalającą użytkownikowi na przetwarzanie, za pomocą procedur i modeli, informacji wejściowych w wyjściowe.

System informacyjny a informatyczny?

SYSTEM INFORMATYCZNY

Wyodrębniona część systemu informacyjnego, która jest z punktu widzenia przyjętych celów skomputeryzowana^[1].

^[1] J. Kisielnicki, H. Sroka, *Systemy informacyjne biznesu. Informatyka dla zarządzania*, Placet, Warszawa 1999, s. 20.



System informacyjny a informatyczny?

SYSTEM INFORMATYCZNY

Ta część systemu informacyjnego przedsiębiorstwa, w której ramach generowanie i gromadzenie danych źródłowych, ich przetwarzanie i analiza oraz prezentowanie informacji odbywa się przy wykorzystaniu metod, technik, technologii i narzędzi (**komputerów**)^[1].

^[1] A. Januszewski, *Informatyka w przedsiębiorstwie*.

Systemy i procesy informatyzacji, Wyższa Szkoła Zarządzania i Finansów, Bydgoszcz 2001, s. 23.

SYSTEM INFORMACYJNY

Wymagania^[1]:

Dostępność

Aktualność

Rzetelność

Kompletność

Porównywalność

Niezawodność

Przetwarzalność

Elastyczność

Wydajność

Ekonomiczność

Czas reakcji systemu

Szczegółowość

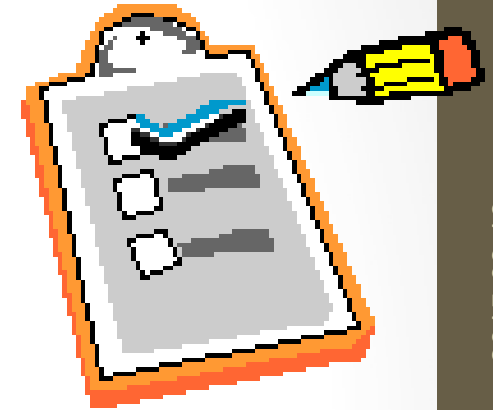
Stabilność systemu

Priorytetowość

Poufność

Bezpieczeństwo

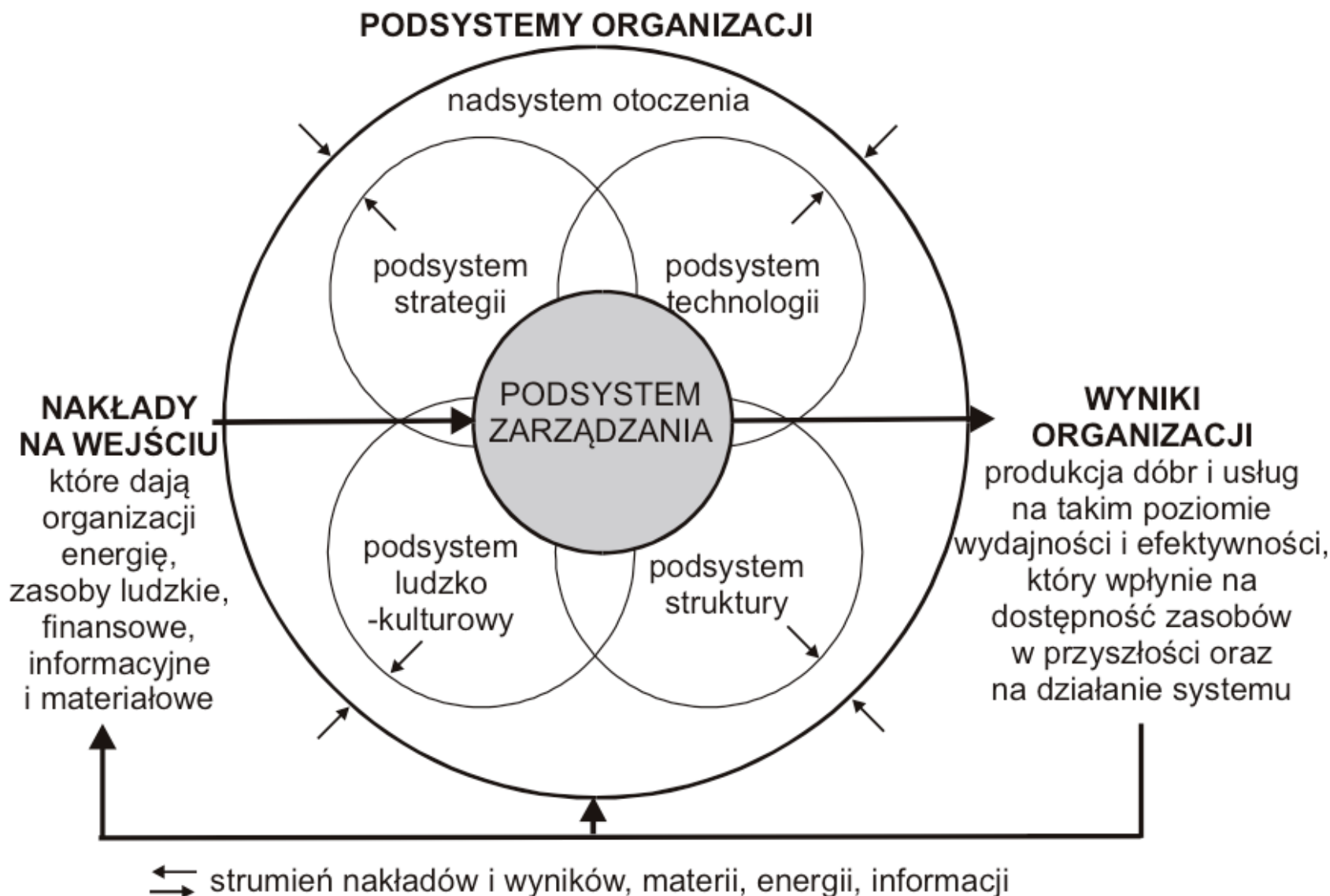
Łatwość użytkowania



[1] J. Kisielnicki, H. Sroka, *Systemy informacyjne biznesu. Informatyka dla zarządzania*,

Placet, Warszawa 1999, s. 35-37.

System informacyjny a informatyczny?



Formalna definicja systemu informatycznego

- Z formalnego punktu widzenia System Informatyczny (SI) danej organizacji można przedstawić w postaci złożenia siedmiu uporządkowanych elementów :
- $SI = \{P, I, T, O, M, R, N\}$
- Znaczenie i zawartość tych poszczególnych elementów będą przedstawiona na kolejnych slajdach

P – personel korzystający z systemu

$$P = \{P_z, P_i, P_u\}$$

P_z – personel szczebla zarządzającego i kierowniczego.

P_i – personel informatyczny.

P_u – pozostali użytkownicy systemu informatycznego.

I – dane i informacje

$$I = \{I_e, I_p, I_m\}$$

I_e – zasoby informacyjne w postaci elektronicznej.

I_p – zasoby informacyjne w postaci papierowej.

I_m – zasoby informacyjne w postaci wiedzy określonych osób.

**T – zbiór urządzeń i narzędzi
technologii informatycznej**

$$T = \{T_s, T_o, T_k\}$$

T_s – sprzęt (urządzenia komputerowe)

T_o – oprogramowanie

T_k – telekomunikacja

O – zbiór stosowanych rozwiązań organizacyjnych

$$O = \{O_s, O_z, O_r, O_p, O_b, \dots\}$$

O_s – strategia rozwoju systemu informatycznego

O_z – zarządzenia, rozporządzenia i wytyczne regulujące kwestie związane z utworzeniem, funkcjonowaniem i rozwojem systemu informatycznego.

O_r – regulaminy dotyczącego zasad użytkowania systemu.

O_p – procedury ochronne i procedury awaryjne.

O_b – dokument polityki bezpieczeństwa.

M – zbiór metainformacji

$$M = \{Me, Mp, Mm\}$$

Me – metainformacje w postaci elektronicznej.

Mp – metainformacje w postaci papierowej.

Mm – metainformacje zgromadzone
w pamięci osób.

R – relacje pomiędzy elementami systemu informatycznego

$$R = \{Rp-p, Rp-t, Rt-i, Rp-i\}$$

Rp-p – relacje pomiędzy personelem systemu.

Rp-t – relacje pomiędzy personelem systemu a urządzeniami technologii informatycznej.

Rt-I – relacje pomiędzy urządzeniami a zasobami informacyjnymi.

Rp-I – relacje pomiędzy personelem a zasobami informacyjnymi

N – infrastruktura i otoczenie systemu informatycznego

$$N = \{Nw, Nz, Nw-z\}$$

Nw – infrastruktura wewnętrzna przedsiębiorstwa

Nz – infrastruktura zewnętrzna przedsiębiorstwa

Nw-z – relacja pomiędzy infrastrukturą wewnętrzną a zewnętrzną przedsiębiorstwa

Architektury systemów rozproszonych

Dotyczy oprogramowania przeznaczanego do wykonania na więcej niż jednym procesorze

Cele

- Poznać główne wady i zalety architektur systemów rozproszonych.
- Poznać różne podejścia do opracowania architektur klient-serwer.
- Zrozumieć różnice między architekturą klient-serwer a architekturą obiektów rozproszonych.
- Zrozumieć pojęcie pośrednika zleceń obiektowych i znać zasady standardów CORBA.

Zawartość

- Architektury wieloprocessowe
- Architektury klient-serwer
- Architektury obiektów rozproszonych
- CORBA

Systemy rozproszone

- Niemal wszystkie współczesne systemy komputerowe są systemami rozproszonymi.
- W systemie rozproszonym przetwarzanie informacji jest wykonywane na kilku komputerach, a nie przydzielone do jednej maszyny.
- Inżynieria systemów rozproszonych ma wiele wspólnego z inżynierią każdego innego rodzaju oprogramowania, istnieją jednak specyficzne zagadnienia, które należy wziąć pod uwagę, projektując takie systemy.

Trzy najważniejsze rodzaje systemów

- **Systemy osobiste**, które nie są rozproszone i są przeznaczone do pracy na komputerze osobistym lub stacji roboczej. Przykładami takich systemów są m.in. procesory tekstów, arkusze kalkulacyjne, systemy graficzne itd..
- **Systemy wbudowane**, które działają na jednym procesorze lub na zintegrowanej grupie procesorów. Przykładami takich systemów są m.in. systemy sterowania sprzętem domowym i systemy sterujące przyrządami.
- **Systemy rozproszone**, w których oprogramowanie systemowe działa na luźno zintegrowanej grupie współpracujących procesorów połączonych siecią. Przykładami takich systemów są m.in. systemy bankomatów, systemy rezerwacji, systemy pracy grupowej itd.

Istotne cechy systemu rozproszonego

- Współdzielenie zasobów
- Otwartość
- Współbieżność
- Skalowalność
- Odporność na awarie
- Przezroczystość

Wady systemów rozproszonych

- Złożoność
- Zarządzanie zabezpieczeniem
- Trudności z zarządzaniem i pielęgnacją
- Nieprzewidywalność czasu reakcji

Zagadnienia projektowania systemów rozproszonych

Zagadnienia projektowe

Opis

Identyfikacja
należy więc
zasobów
wykorzystywać

Zasoby systemu rozproszonego są rozmieszczone na różnych komputerach, opracować schemat nazewnictwa aby użytkownicy mogli znajdować i potrzebne im zasoby. Przykładem takiego schematu jest URL (Uniform Resource jednolity lokalizator zasobów) służący do identyfikacji stron WWW. Jeśli nie znaczącego i powszechnie rozumianego schematu, to wiele zasobów będzie dla użytkowników systemu.

Locator-

zastosuje się

niedostępnych

Komunikacja
komunikacyjnego

Powszechna dostępność Sieci i efektywna implementacja jej protokołu TCP/IP oznacza, że jest to najskuteczniejszy sposób komunikacji komputerów w większości systemów rozproszonych. Tam, gdzie występują szczególne efektywnościowe, niezawodnościowe itp., można skorzystać z innych mechanizmów komunikacji.

wypadku

wymagania

mechanizmów

Jakość usług
efektywność, dostępność i nie-

Jakość usług oferowanych przez system odzwierciedla jego

Architektury systemów rozproszonych

- Architektury klient-serwer
 - W tym podejściu system jest postrzegany jako zbiór usług oferowanych klientom, którzy z nich korzystają. W takich systemach odmiennie traktuje się serwery i klientów.
- Architektury obiektów rozproszonych
 - W tym podejściu nie istnieje rozróżnienie między klientami i serwerami. System jest postrzegany jako zbiór komunikujących się obiektów, których położenie jest nieistotne. Nie ma różnicy między dostawcą i użytkownikiem usług.

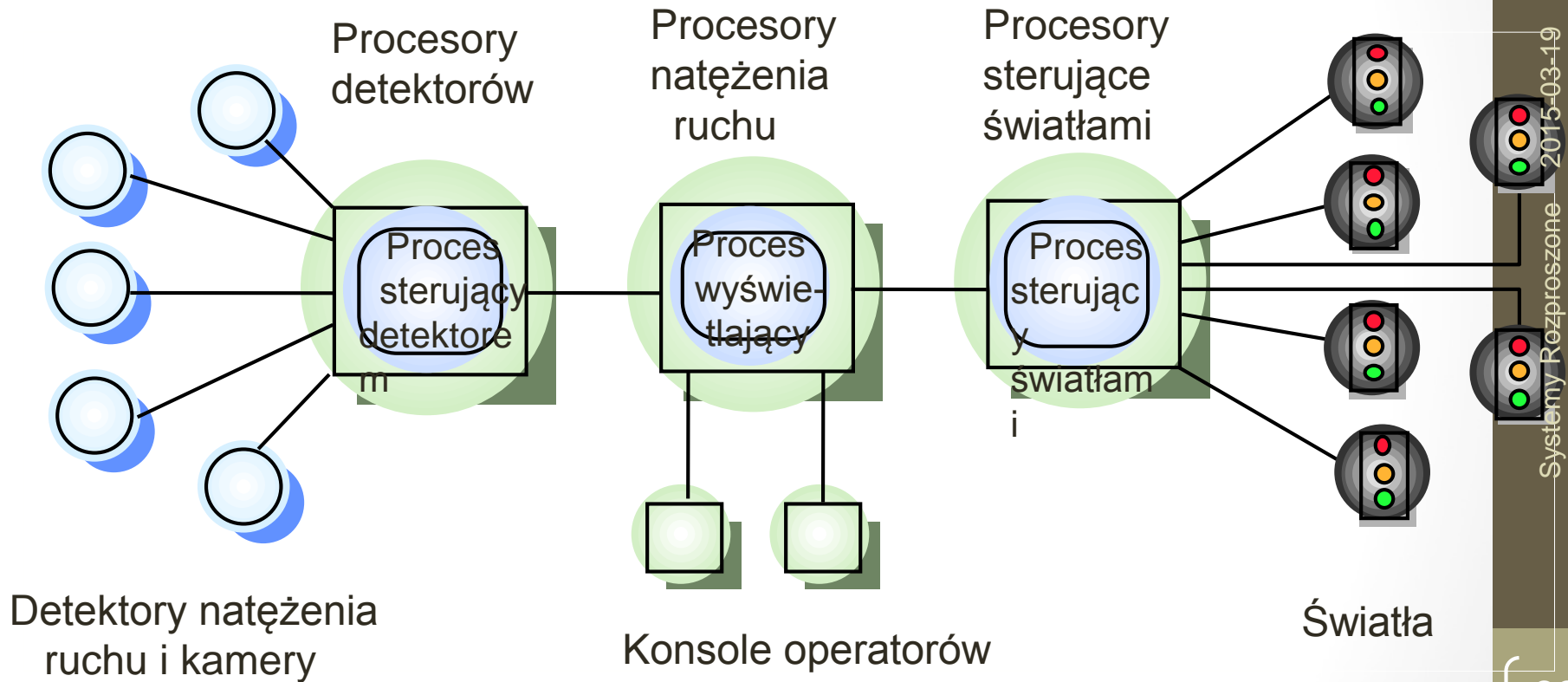
Śródprogramy

- Różne komponenty systemu rozproszonego mogą być zaimplementowane za pomocą rozmaitych języków programowania i działać na zupełnie innych rodzajach procesorów.
- Modele danych, reprezentacja informacji i protokoły komunikacyjne mogą być całkowicie odmienne.
- System rozproszony musi zatem zawierać oprogramowanie, które będzie zarządzać tymi różnorodnymi częściami oraz zapewni możliwość komunikacji i wymiany danych.
- Do takiego oprogramowania odnosi się podejście **śródprogramu** (**middleware**), które znajduje się w środku różnych rozproszonych komponentów systemu.

Architektury wieloprocessowe

- Najprostszym modelem systemu rozproszonego jest system wieloprocessorowy.
- Z logicznego punktu widzenia procesory zajmujące się zbieraniem informacji, podejmowaniem decyzji i sterowaniem efektorami mogłyby działać na jednym procesorze sterowanym przez moduł szeregujący. Użycie wielu procesorów poprawia efektywność i odporność systemu.
- Przydział procesów do procesorów może być zadany z góry (tak zwykle jest w systemach krytycznych) albo sterowany przez dyspozytora, który decyduje o przyznaniu procesora procesowi.

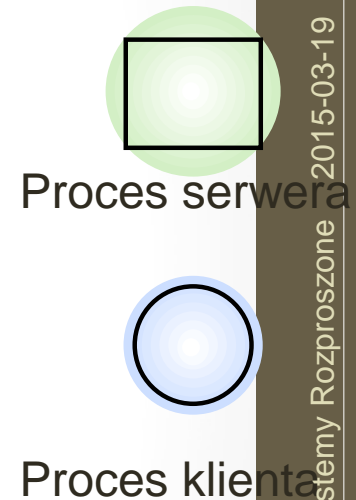
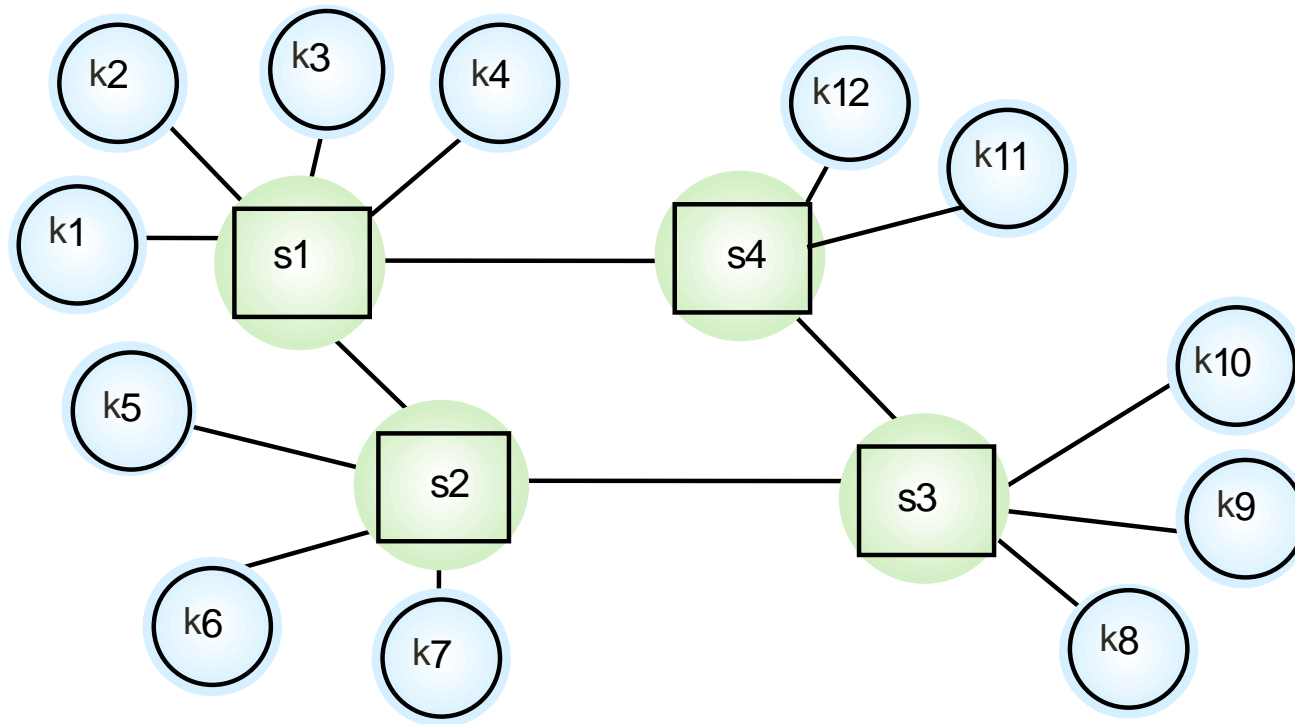
System wieloprocessorowy do kierowania ruchem



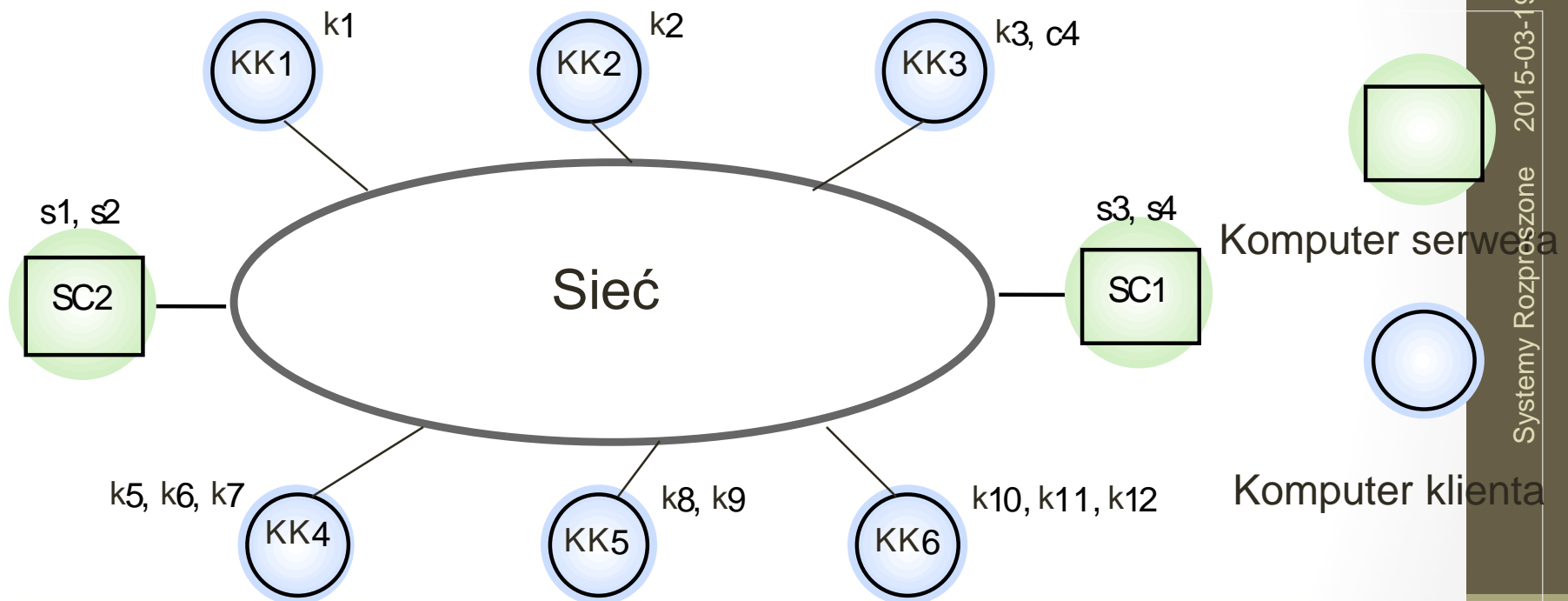
Architektury klient-serwer

- W takiej architekturze program użytkowy jest modelowany jako zbiór usług oferowanych przez serwery i zbiór klientów, którzy z tych usług korzystają.
- Klienci muszą znać dostępne serwery, ale zwykle nie muszą wiedzieć o istnieniu innych klientów.
- Klienci i serwery są oddzielnymi procesami.
- Procesy i procesory systemu nie muszą być wzajemnie jednoznacznie przyporządkowane.

Logiczny model rozproszonej architektury klient-serwer



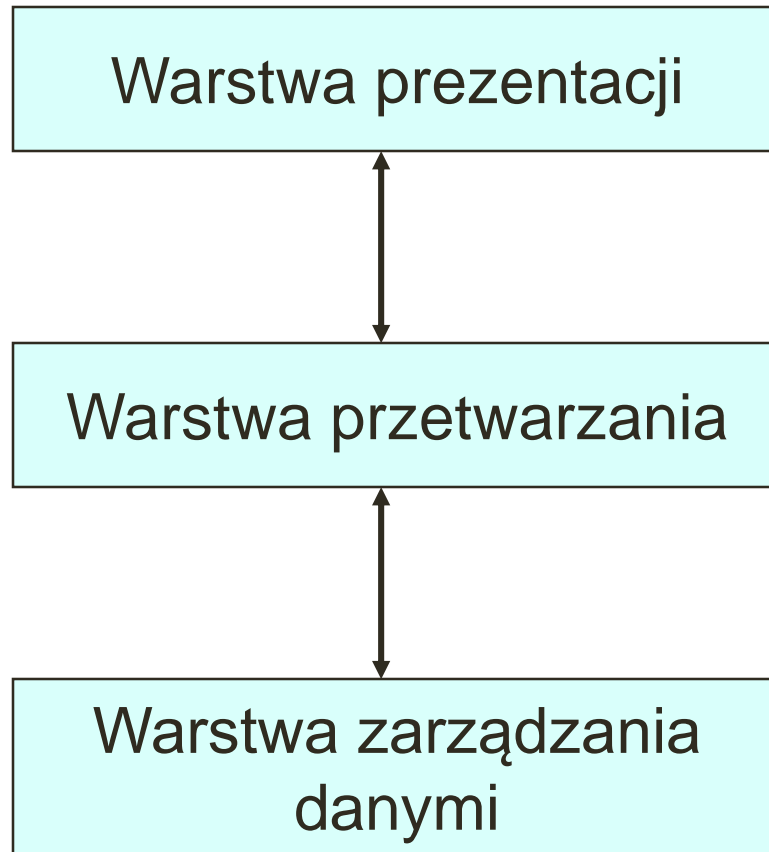
Komputery w sieci klient-serwer



Projektowanie systemu klient-serwer z pomocą architektury warstwowej

- Warstwa prezentacji dotyczy przedstawiania informacji użytkownikowi i całego kontaktu z użytkownikiem.
- W warstwie przetwarzania implementuje się logikę programu użytkowego.
- Warstwa zarządzania danymi jest odpowiedzialna za wszystkie operacje na bazie danych

Warstwy programu użytkowego



Dwa rodzaje architektury klient-serwer

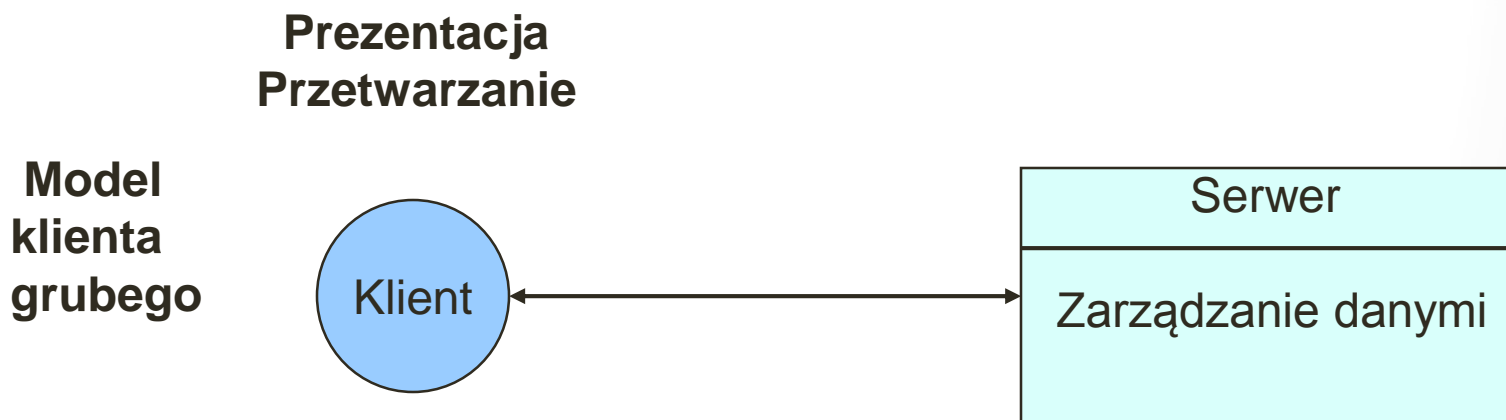
- ***Model klienta cienkiego***

- W tym modelu całość przetwarzania i zarządzania danymi ma miejsce na serwerze. Jedynym zadaniem klienta jest uruchomienie oprogramowania prezentacyjnego.

- ***Model klienta grubego***

- W tym modelu serwer odpowiada jedynie za zarządzanie danymi. Oprogramowanie u klienta implementuje logikę programu użytkowego i kontakt z serwerem.

Klient cienki i klient gruby



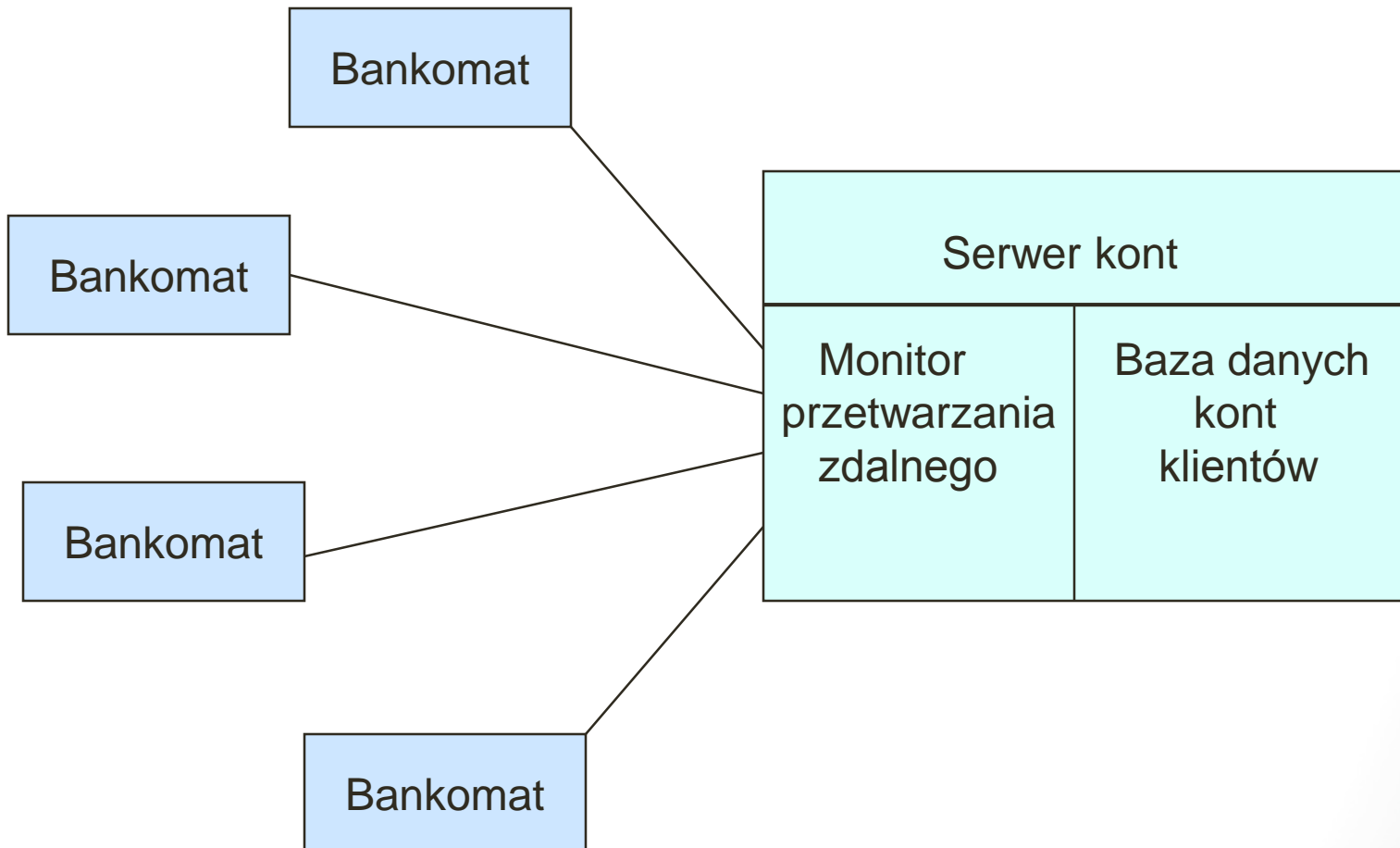
Model klienta cienkiego

- Architektura dwuwarstwowa z klientem cienkim jest najprostszym rozwiązaniem, które można wykorzystać w scentralizowanych systemach odziedziczonych ewoluujących w kierunku architektur klient-serwer.
- Interfejs użytkowy działa jako serwer i obsługuje przetwarzanie użytkowe oraz zarządzanie danymi.
- Model klienta cienkiego może być również zaimplementowany tam, gdzie klienci są raczej prostymi urządzeniami sieciowymi, a nie komputerami osobistymi albo stacjami roboczymi.
- Na takim urządzeniu sieciowym działa przeglądarka Sieci oraz interfejs użytkownika realizowany przez ten system.
- Najważniejszą wadą modelu klienta cienkiego jest duże obciążenie przetwarzaniem zarówno sieci, jak i serwera.

Model klienta grubego

- Korzysta się z dostępnej mocy obliczeniowej klienta przekazując mu zarówno przetwarzanie związane z logiką programu użytkowego, jak i prezentację.
- Serwer jest zasadniczo serwerem transakcji, który zarządza transakcjami w bazie danych.
- Dobrze znanym przykładem architektury tego typu są systemy bankomatów. Bankomat jest tam klientem, a serwerem jest komputer główny obsługujący bazę danych kont klientów.
- W modelu klienta grubego przetwarzanie jest bardziej efektywne niż w wypadku modelu klienta cienkiego, zarządzanie systemem jest natomiast trudniejsze w tym pierwszym modelu.

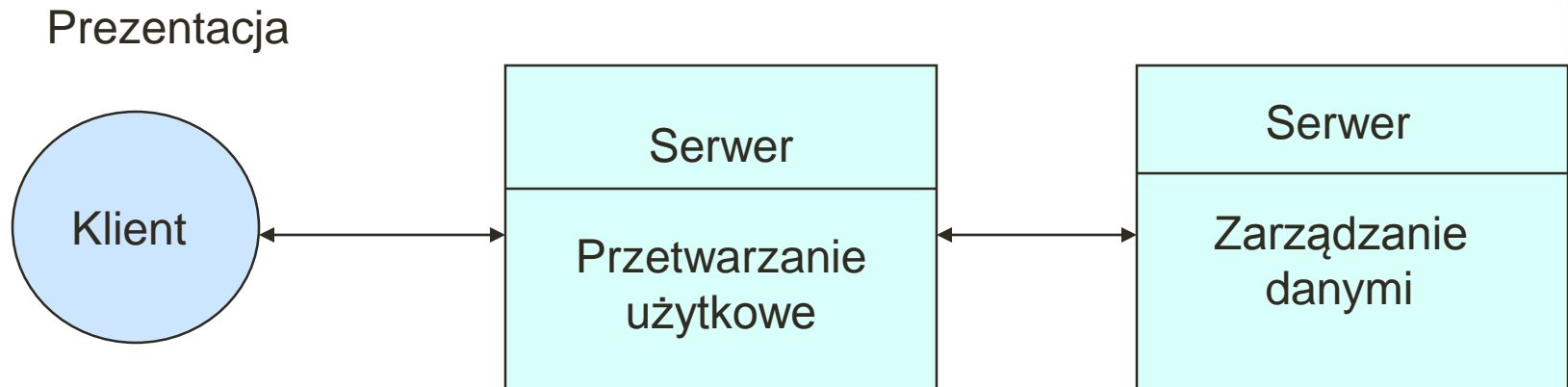
System klient-serwer do obsługi bankomatów



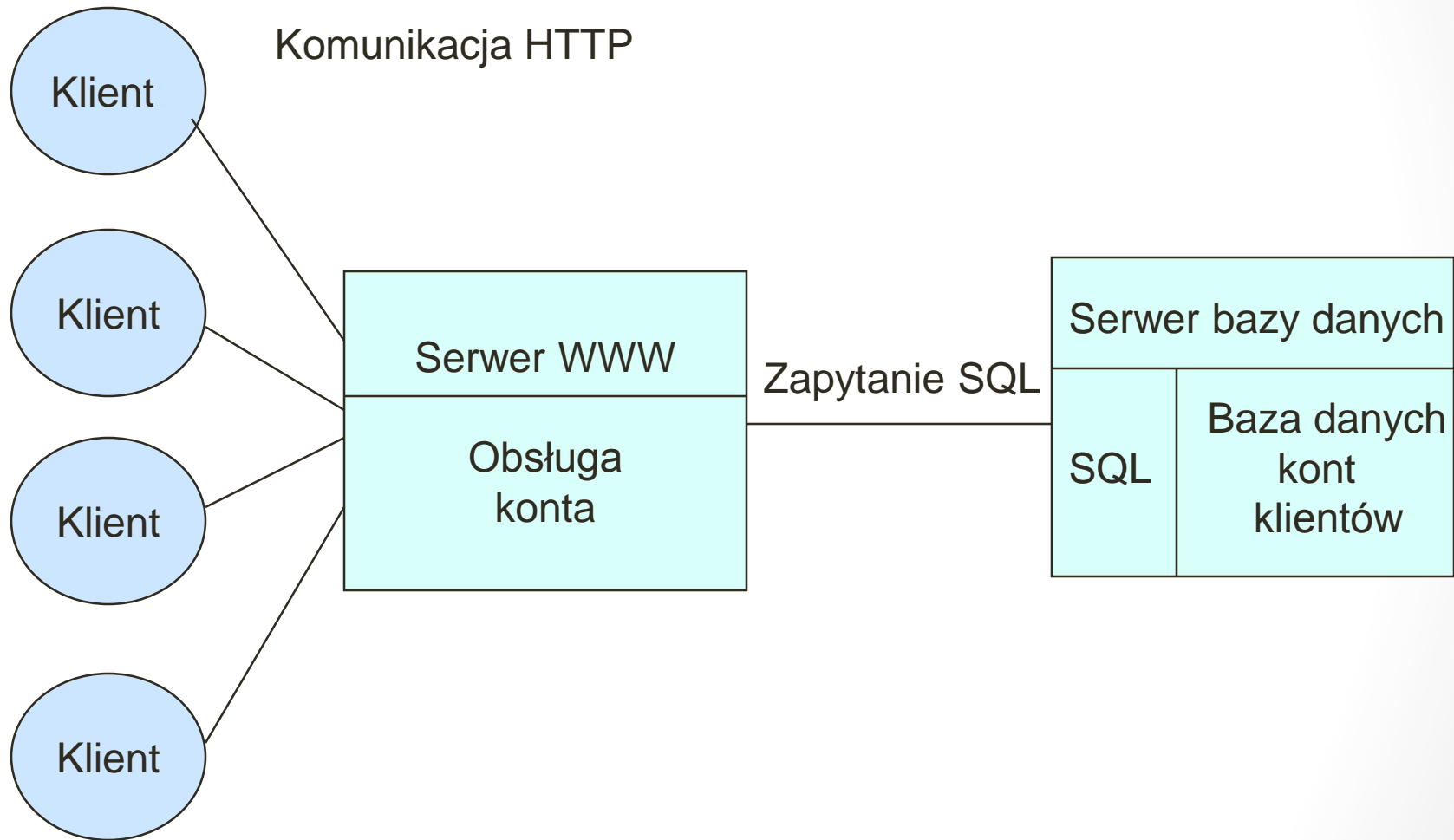
Uwagi o architekturze warstwowej

- W tej architekturze prezentacja, przetwarzanie użytkowe i zarządzanie danymi są logicznie oddzielonymi procesami.
- Niekoniecznie potrzebne są trzy systemy komputerowe włączone do sieci.
- Jeden komputer serwera może obsługiwać zarówno przetwarzanie użytkowe, jak i zarządzanie danymi programu użytkowego jako dwa oddzielne logicznie serwery.
- Gdy jednak oczekiwania wzrosną, można będzie dość łatwo wyodrębnić przetwarzanie użytkowe i zarządzanie danymi, po czym uruchomić je na oddzielnych procesorach.

Architektura trójwarstwowa klient-serwer



Architektura trójwarstwowa system bankowego w Sieci



Zastosowania różnych architektur warstwowych klient-serwer

Architektura

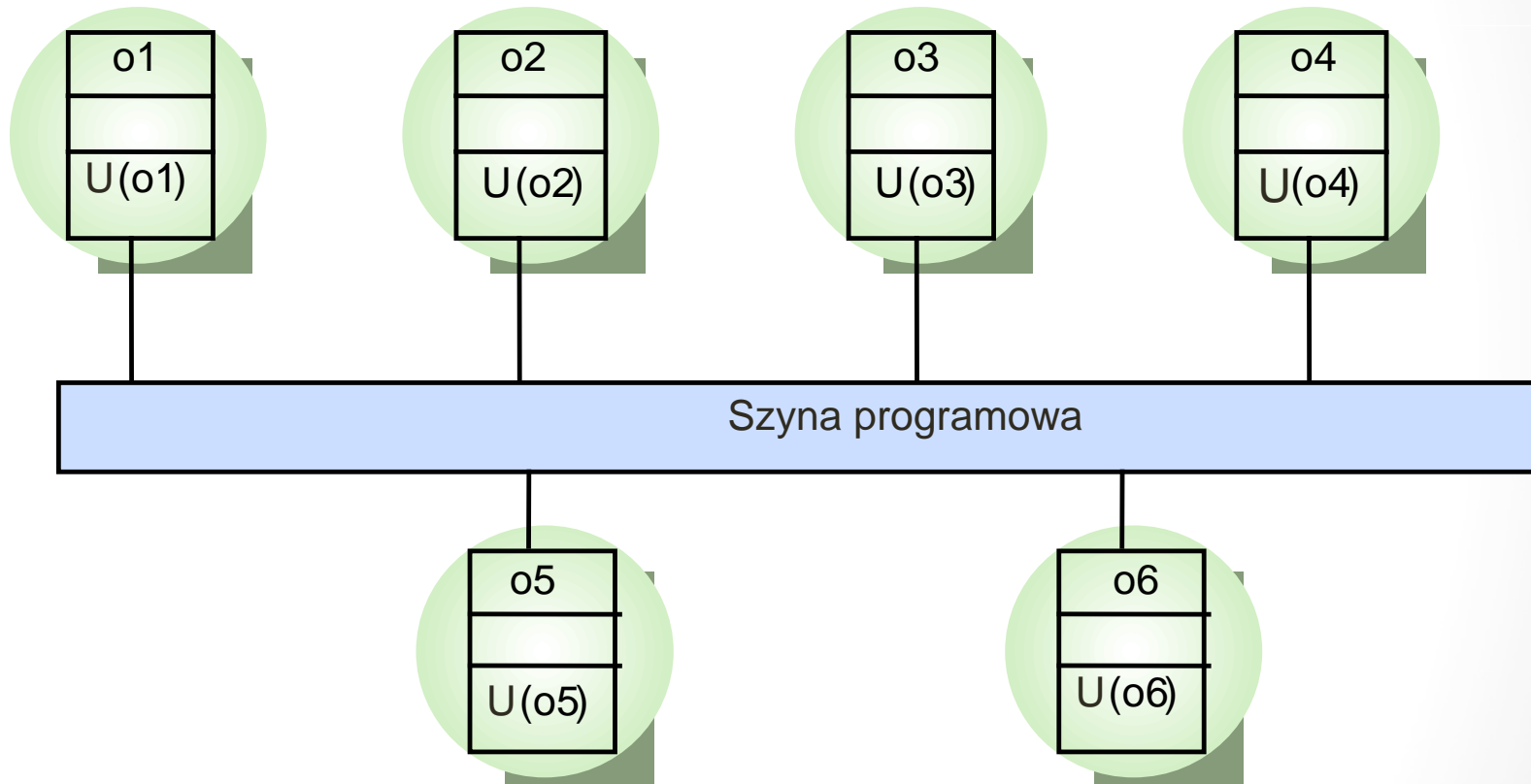
Zastosowania

| | |
|--|--|
| Architektura dwuwarstwowa klient-serwer z klientami cienkimi | <p>Systemy odziedziczone, w których oddzielenie przetwarzania użytkowego od zarządzania danymi jest niepraktyczne</p> <p>Programy użytkowe wykonujące dużo obliczeń, ale w małym stopniu (albo wcale) zarządzające danymi, np. kompilatory</p> <p>Programy użytkowe dużo korzystające z danych (przeglądanie i zapytywanie), ale wykonujące mało (albo wcale) obliczeń użytkowych</p> |
| Architektura dwuwarstwowa klient-serwer z klientami grubymi | <p>Programy użytkowe, w których obliczenia są wykonywane u klienta przez COTS (komponenty z półki), np. Microsoft Excel</p> <p>Programy użytkowe wymagające złożonego obliczeniowo przetwarzania danych, np. przedstawiania graficznego danych</p> <p>Programy użytkowe ze względnie stabilną funkcjonalnością oferowaną użytkownikowi stosowane w środowisku ze starannie ustalonym zarządzaniem systemem</p> |
| Architektura trójwarstwowa lub wielowarstwowa klient-serwer | <p>Ogromne programy użytkowe z setkami lub tysiącami użytkowników</p> <p>Programy użytkowe, w których zarówno dane, jak i programy są płynne</p> <p>Programy użytkowe, w których integruje się dane z wielu źródeł</p> |

Charakterystyka architektur systemów rozproszonych

- W modelu systemu rozproszonego klient-serwer, klienci odróżniają się od serwerów.
- Klienci otrzymują usługi od serwerów, ale nie od innych klientów.
- Serwery mogą działać jako klienci korzystający z usług innych serwerów, ale nie mogą żądać usług od klientów.
- Klienci muszą znać usługi oferowane przez specyficzne serwery i wiedzieć, jak się z tymi serwerami porozumieć.
- Ten model sprawdza się w wielu zastosowaniach. Ogranicza jednak swobodę projektantów systemu, którzy muszą zdecydować, gdzie mają być oferowane usługi.

Architektury *obiektów* rozproszonych: usunięcie rozróżnienia na klientów i serwery



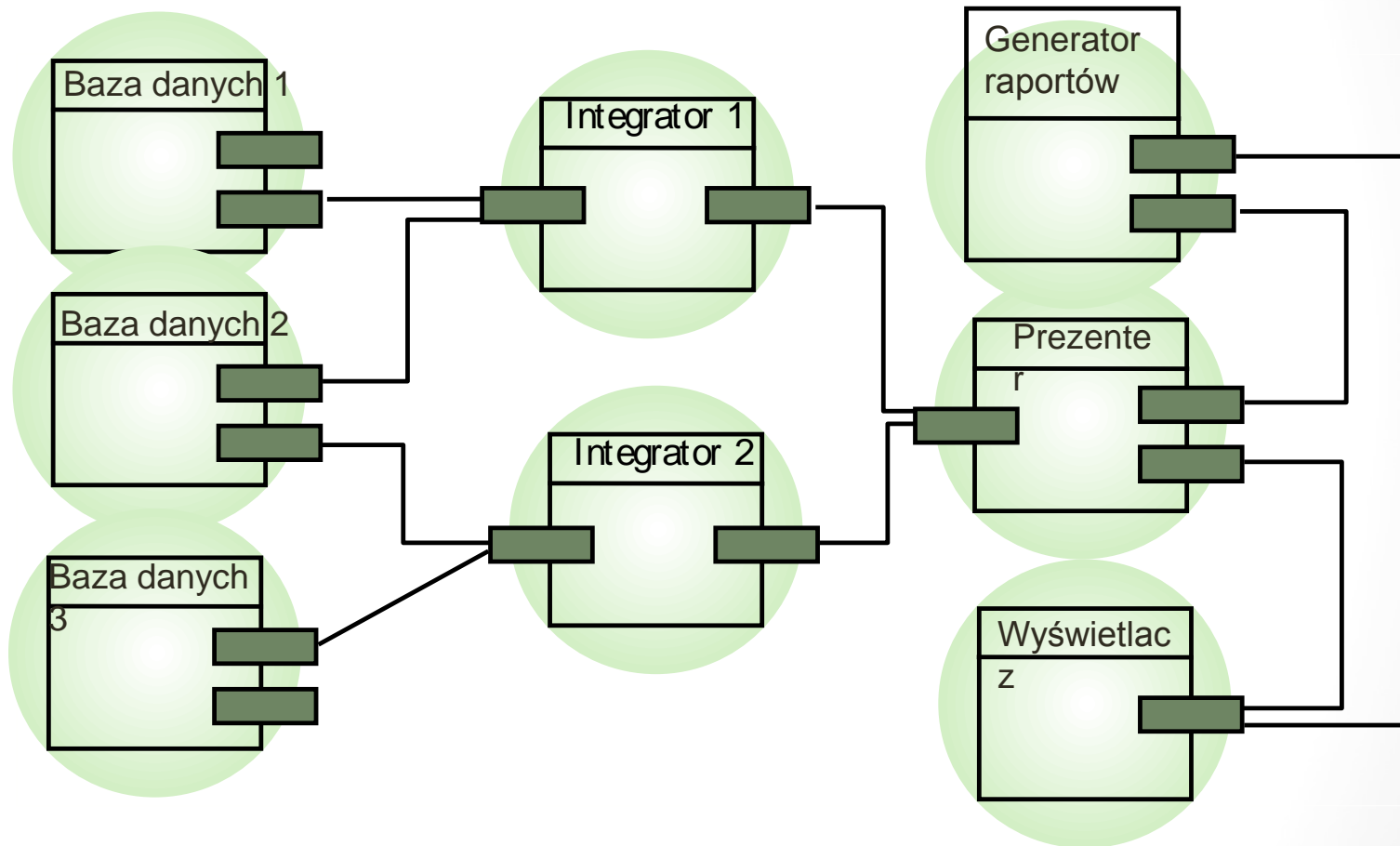
Zalety architektury obiektów rozproszonych

- Umożliwia projektantowi odłożenie w czasie decyzji, gdzie i jak oferować usługi.
- Jest architekturą bardzo otwartych systemów, która umożliwia dodawanie nowych zasobów.
- System jest bardzo elastyczny i skalowalny. Do obsługi rozmaitych obciążeń można utworzyć różne egzemplarze systemu z tym samym zbiorem usług oferowanych przez odrębne lub powielone obiekty.
- W miarę potrzeby można dynamicznie zmieniać konfigurację systemu przez migrację obiektów w sieci.

Wykorzystanie architektury obiektów rozproszonych

- Jako model logiczny, który pomaga w ustaleniu struktury i organizacji systemu. W tym wypadku trzeba jednak zastanowić się, jak zapewnić funkcjonalność użytkową jedynie w kategoriach usług i ich kombinacji. Następnie należy wypracować sposób oferowania tych usług przez kilka obiektów rozproszonych.
- Jako elastyczne podejście do implementacji systemów klient-serwer. W tym wypadku model logiczny systemu odpowiada architekturze klient-serwer, ale zarówno klienci, jak i serwery mają postać obiektów rozproszonych porozumiewających się za pośrednictwem szyny programowej.

Architektura obiektów rozproszonych w systemu wyszukiwania danych



Zalety architektury obiektów rozproszonych w systemie wyszukiwania danych

- Taka architektura umożliwia zwiększenie liczby wykorzystywanych baz danych bez przerywania pracy systemu.
- Każda baza danych to po prostu kolejny obiekt rozproszony.
- Obiekty bazy danych mogą oferować uproszczony interfejs, który umożliwia kontrolę dostępu do danych.
- Odczytywane bazy danych mogą znajdować się na różnych maszynach.
- Taka architektura umożliwia eksplorację nowych rodzajów związków przez dodanie nowych obiektów integratorów.

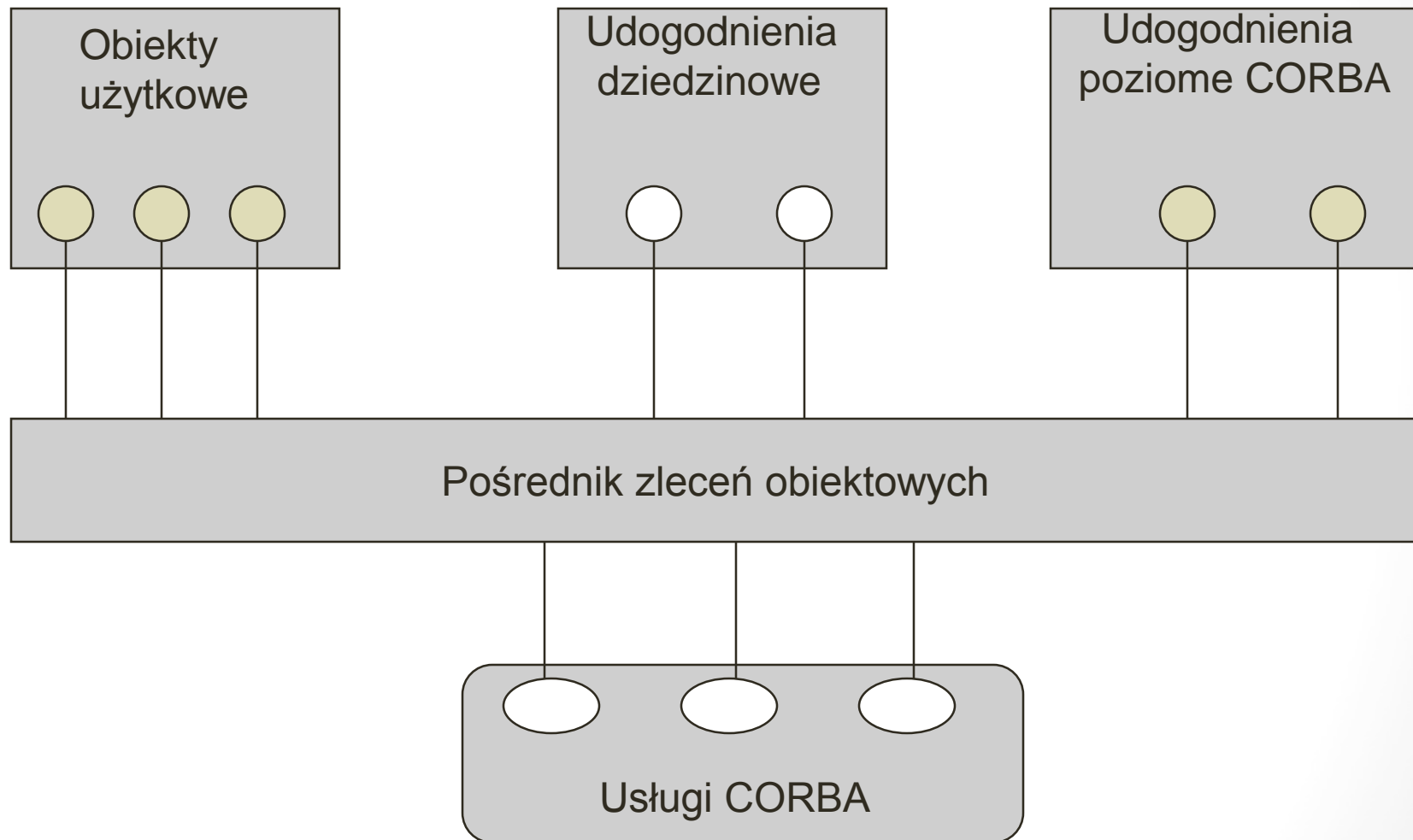
CORBA i DCOM jako standardy pośredników zleceń obiektowych

- CORBA (Common Object Request Broker Architecture) jest zbiorem standardów dla środowisk zdefiniowanym przez OMG (Object Management Group).
- Implementacje CORBA są dostępne dla Unixa i systemów operacyjnych firmy Microsoft.
- Z kolei DCOM (Distributed Component Object Model) jest standardem, który opracowano i zaimplementowano w firmie Microsoft i zintegrowano z systemami operacyjnymi jej produkcji.
- Model rozproszonych obliczeń w DCOM jest mniej ogólny niż CORBA.

Komponenty rozproszonego programu użytkowego wg. OMG

- Obiekty użytkowe zaprojektowane i zaimplementowane dla tego programu użytkowego.
- Obiekty standardowe zdefiniowane przez OMG na użytek specyficznej dziedziny.
- Główne usługi CORBA związane z podstawowymi aspektami obliczeń rozproszonych, takie jak katalogi, zarządzanie zabezpieczeniami itd.
- Poziome udogodnienia CORBA, takie jak ułatwienia interfejsu użytkownika, zarządzanie systemem itp.

Struktura rozproszonego programu użytkowego opartego na CORBA



Standardy CORBA obejmują:

- Model obiektów użytkowych, w którym obiekty CORBA obudowują stan i mają dobrze określony i niezależny od języka interfejs opisany w IDL (Interface Definition Language – język opisu interfejsów).
- Pośrednik zleceń obiektowych (Object Request Broker, ORB), który obsługuje żądania obiektów.
- Zbiór ogólnych usług obiektowych, które prawdopodobnie będą potrzebne w wielu rozproszonych programach użytkowych.
- Zbiór wspólnych komponentów zbudowanych na bazie tych podstawowych usług. Te komponenty mogą się przydać w rozmaitych programach użytkowych.

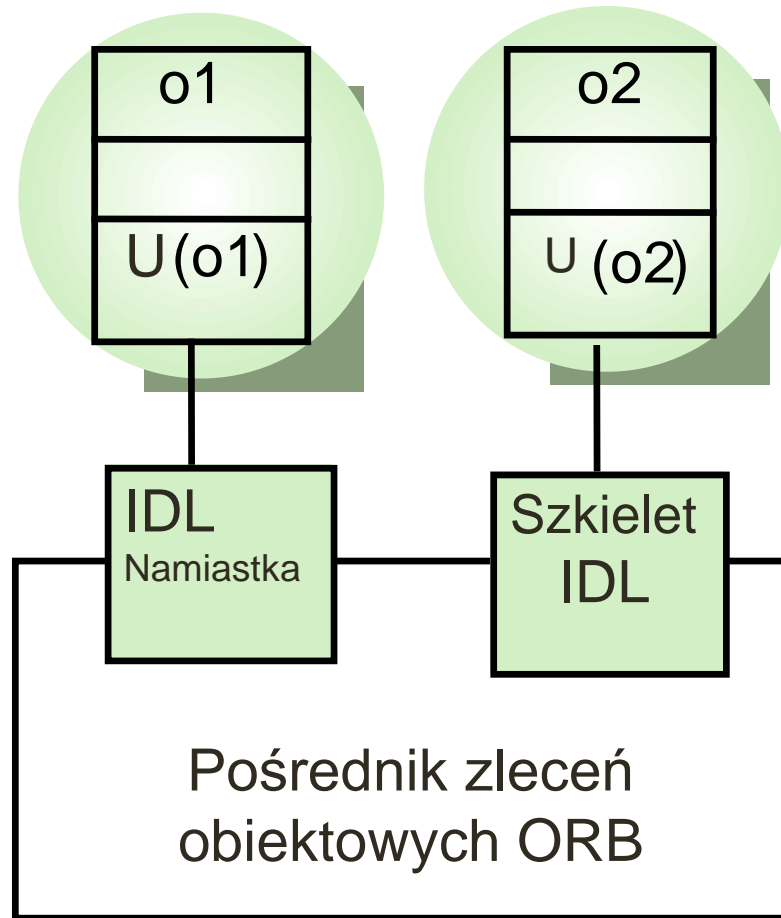
Model obiektowy CORBA

- Obiekt jest obudową atrybutów i usług, jak to jest zwykle w wypadku obiektów.
- Obiekty CORBA muszą mieć jednak oddzielną definicję interfejsu, w której określa się publiczne atrybuty i metody obiektu.
- Interfejs obiektów CORBA ustala się za pomocą standardowego, niezależnego od języka programowania języka opisu interfejsów IDL (ang. Interface Description Language).
- Gdy obiekt pragnie skorzystać z usług innego obiektu, dostaje się do nich przez interfejs IDL.
- Obiekty CORBA mają unikatowe identyfikatory zwane IOR (ang. Interoperable Object Reference) czyli odniesienia do obiektów umożliwiające współpracę.

Pośrednik zleceń obiektowych (ORB)

- Obiekt wywołujący ma do dyspozycji namiastkę IDL, która zawiera definicję interfejsu obiektu oferującego żądaną usługę.
- Osoba programująca umieszcza wywołania tej namiastki w swojej implementacji wszędzie tam, gdzie jest potrzebna ta usługa.
- IDL jest nadzbiorem C++, jeśli więc programuje się w tym języku, to korzystanie z namiastki jest łatwe. Jest również dość łatwe w C i Javie.
- Zdefiniowano również przekształcenia IDL na inne języki, na przykład COBOL i Ada. W wypadku tych języków zwykle jest jednak konieczne wspomaganie narzędzi łączących z IDL.

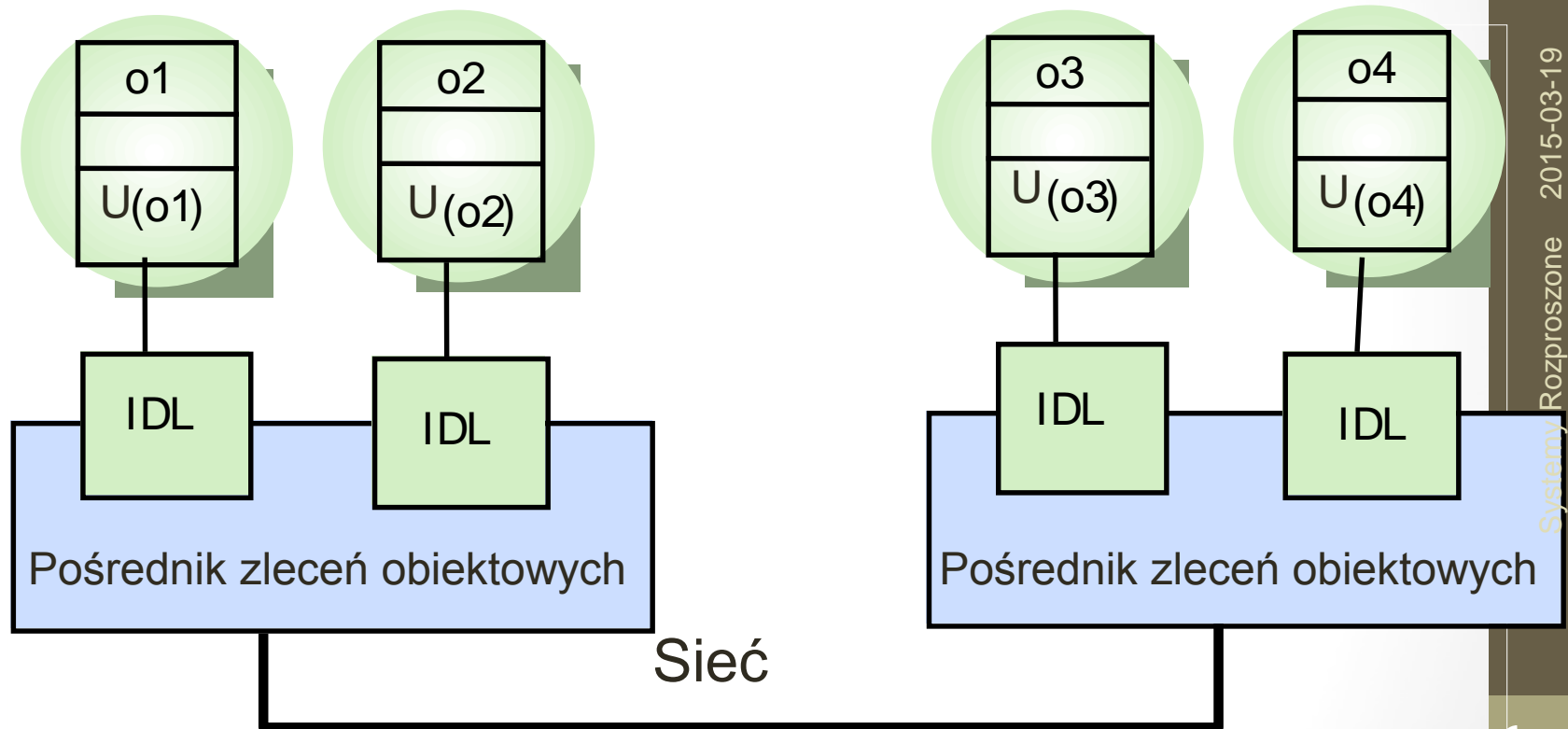
Komunikacja obiektów za pośrednictwem ORB



Komunikacja między pośrednikami ORB

- Pośrednicy ORB często muszą się porozumieć.
- Implementacja CORBA realizuje komunikacje między pośrednikami ORB przez dostęp do opisu interfejsów w IDL oraz standardowy w OMG protokół Generic Inter-ORB Protocol (GIOP) czyli ogólny protokół komunikacji między ORB).
- W tym protokole zdefiniowano standardowe komunikaty, które mogą być przesyłane przez pośredników ORB i służą do implementacji zdalnych wywołań obiektów oraz przekazywania informacji.
- Gdy połączy się go z niskopoziomowymi protokołami sieci TCP/IP, GIOP umożliwia komunikację ORB za pośrednictwem Sieci.

Komunikacja między różnymi egzemplarzami ORB



Usługi CORBA jako udogodnienia w budowie systemów rozproszonych

- *Usługa katalogowa i usługa handlowa*, dzięki którym obiekty mogą znajdować się i odwoływać do innych obiektów sieci.
- *Usługi powiadamiania*, dzięki którym obiekty mogą powiadomić inne obiekty o zajściu pewnego zdarzenia.
- *Usługi transakcyjne* do obsługi niepodzielnych transakcji i ich wycofywania w wypadku niepowodzenia.

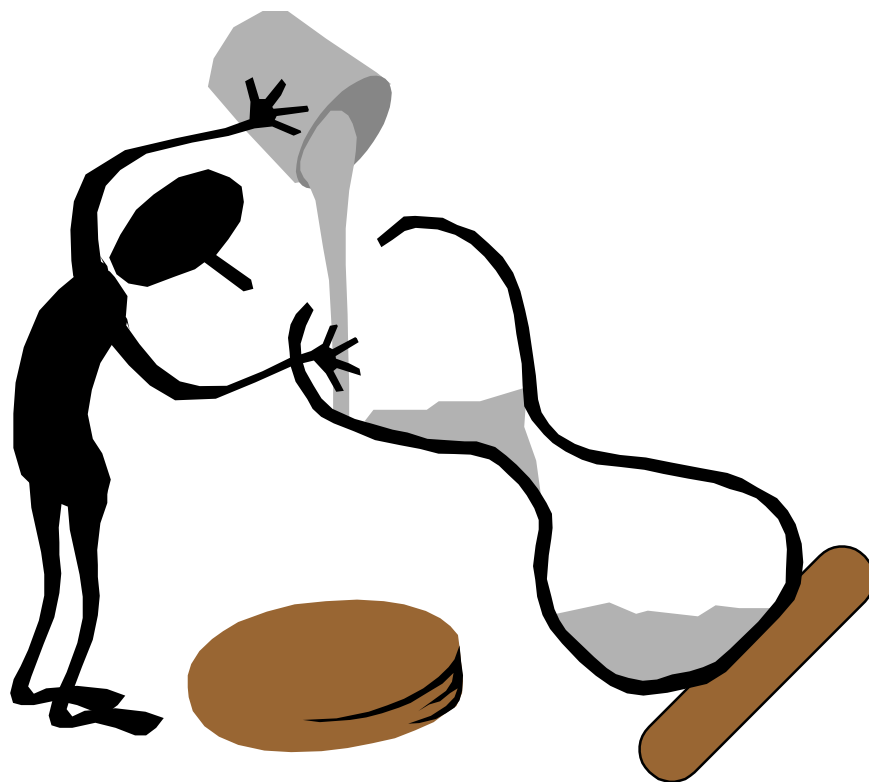
Główne tezy

- Obecnie niemal wszystkie nowe wielkie systemy są rozproszone. Ich oprogramowanie systemowe działa na luźno zintegrowanych grupach procesorów.
- Systemy rozproszone mogą charakteryzować się współdzieleniem zasobów, otwartością, współbieżnością, skalowalnością, odpornością na awarie i przezroczystością.
- Systemy klient-serwer to systemy rozproszone, których modelem jest zbiór usług oferowanych procesom klientów przez serwery.
- W systemach klient-serwer interfejs użytkownika działa zwykle u klienta, a zarządzanie danymi jest zawsze wykonywane przez współdzielony serwer. Funkcjonalność użytkowa może być zaimplementowana na komputerze klienta lub na serwerze.

Główne tezy

- W architekturze obiektów rozproszonych nie ma rozróżnienia na klientów i serwery. Obiekty oferują ogólne usługi, które mogą być wywoływane przez inne obiekty. To podejście można wykorzystać do implementacji systemów klient-serwer.
- Systemy obiektów rozproszonych muszą korzystać ze śródprogramów do obsługi komunikacji obiektów oraz dodawania i usuwania obiektów z systemu. Na poziomie projektowym można postrzegać te śródprogramy jako szynę programową, do której podłącza się obiekty.
- CORBA jest zbiorem standardów dla śródprogramów do obsługi architektur obiektów rozproszonych. Obejmuje definicje modelu obiektowego, pośrednika zleceń obiektowych i wspólnych usług. Są już dostępne rozmaite implementacje usług CORBA.

dziękuję za uwagę



dr inż. Jacek Czerniak
jczerniak@ukw.edu.pl