Projektowanie architektoniczne

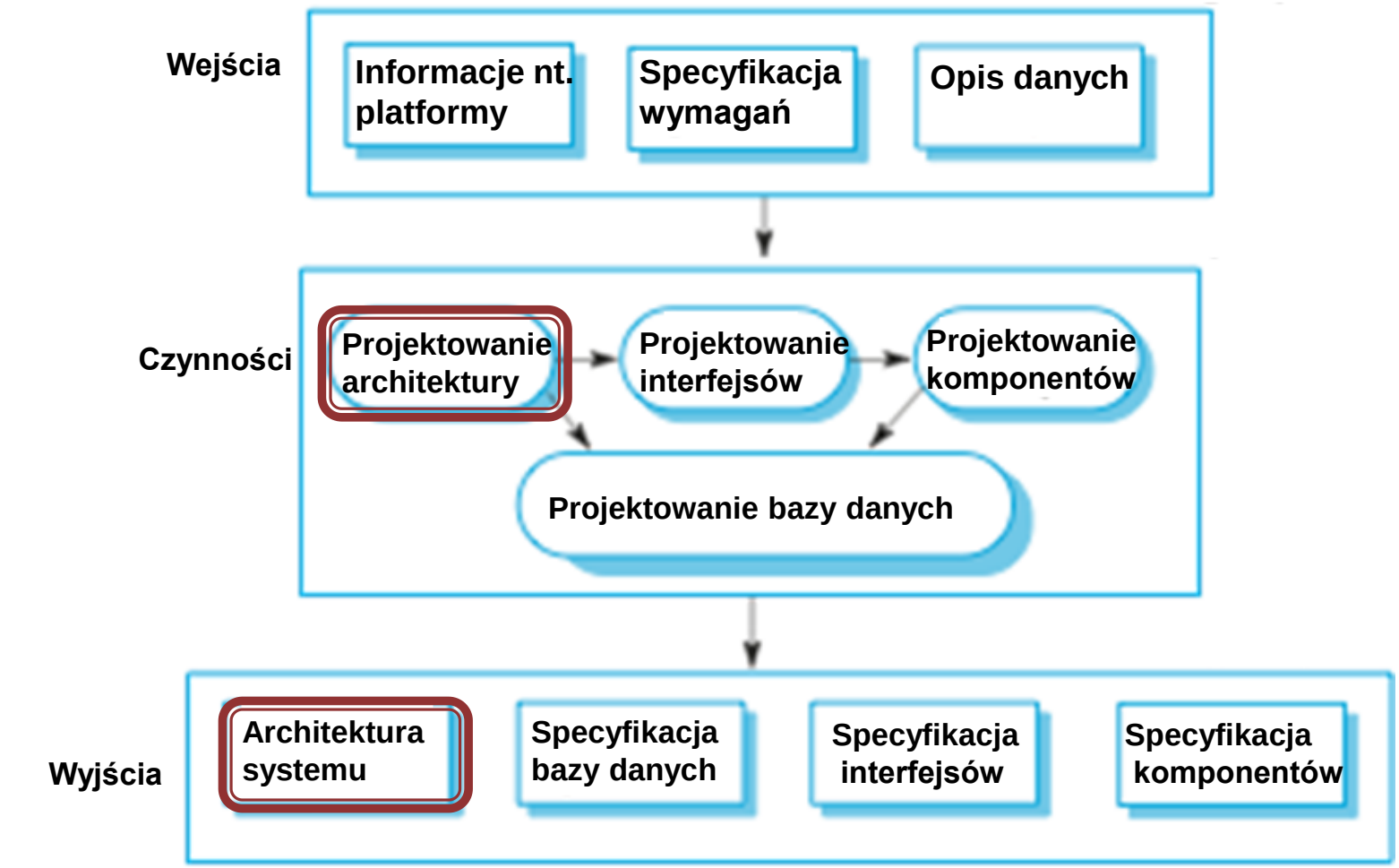
Zagadnienia

- Znaczenie architektury oprogramowania
- Decyzje architektoniczne
- Perspektywy architektoniczne
- Wzorce architektoniczne
- Ogólna architektura aplikacji

Architektura oprogramowania

- Projektowanie architektoniczne
  - Proces projektowania, którego celem jest identyfikacja podsystemów, na które należy podzielić system, oraz szkieletu pozwalającego na kontrolę oraz wzajemną komunikację podsystemów.
- Architektura oprogramowania
  - Rezultat projektowania architektonicznego

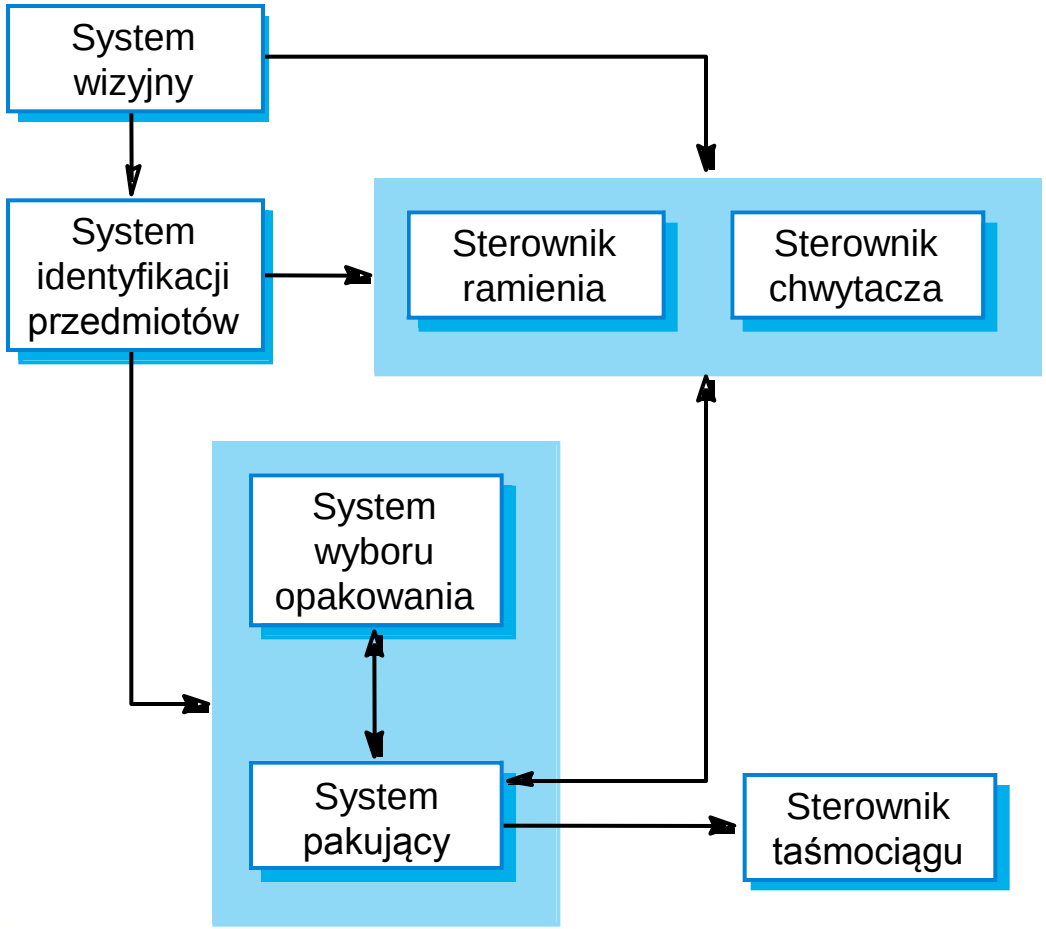
Ogólny model procesu projektowania oprogramowania



Projektowanie architektoniczne

- Wczesny etap procesu projektowania systemu.
- Stanowi powiązanie pomiędzy specyfikacją a procesem projektowania.
- Często wykonywane równolegle z czynnościami specyfikacji oprogramowania.
- Dotyczy identyfikacji głównych komponentów systemu oraz zasad ich wzajemnej komunikacji.

Architektura systemu kontrolującego pracę robota pakującego



Abstrakcje architektoniczne

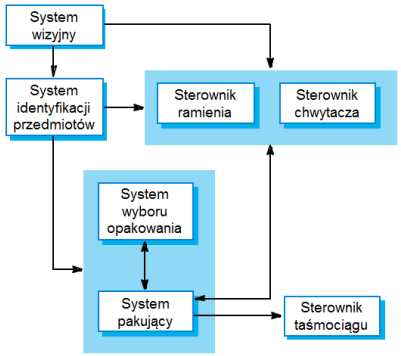
- **Architektura małej skali (ang. Architecture in the small)**
  - Architektura pojedynczych programów. Sposób w jaki pojedynczy program dekomponowany jest do poziomu mniejszych elementów (komponentów).
- **Architektura wielkoskalowa (ang. Architecture in the large)**
  - Dotyczy architektury systemów korporacyjnych, które składają się z systemów, aplikacji, komponentów. Systemy korporacyjne są rozproszone w sposób dopuszczający sytuację , w której poszczególne jego elementy mogą być zarządzane i stanowić własność różnych organizacji.

Zalety jawnej architektury

- Porozumienie stron
  - Architektura może zostać wykorzystana jako element dyskusji pomiędzy zainteresowanymi stronami.
- Analiza systemu
  - W znaczeniu analizy mającej dać odpowiedź na pytanie czy system będzie w stanie zrealizować wymagania niefunkcjonalne.
- Wielokrotne użycie w wielkiej skali
  - Sama architektura może być podstawą wielokrotnego użycia
  - Koncepcja architektury linii produkcyjnej.

Reprezentowanie (dokumentowanie) architektury

- Najczęściej
  - proste, nieformalne diagramy blokowe ukazujące jednostki systemu oraz relacje pomiędzy nimi.
  - Abstrakcyjne - nie pozwalają na reprezentację natury powiązań pomiędzy komponentami oraz widocznych (zewnętrznie) cech podsystemów.
  - Użyteczne dla celów porozumienia pomiędzy stronami oraz planowania projektu



Wykorzystanie modeli architektonicznych

- Ułatwienie dyskusji na temat projektu systemu
  - Punkt odniesienia w dyskusjach, pozwala na pominięcie skomplikowanych szczegółów.
- Sposób dokumentowania sposobu, w jaki została zaprojektowana architektura systemu
  - Opracowanie kompletnego modelu systemu pokazującego oddzielne komponenty systemu, interfejsy pomiędzy nimi oraz powiązania.

Decyzje architektoniczne

- Czy istnieje generyczna architektura aplikacji, którą można wykorzystać?
- Czy i w jaki sposób system będzie rozpraszany?
- Jaki styl architektoniczny będzie najodpowiedniejszy?
- Jakie podejście do strukturalizacji systemu należy wykorzystać?
- W jaki sposób system będzie dekomponowany do poziomu komponentów?
- Jaką strategię kontroli wykorzystać?
- Jaką metodę oceny architektury wykorzystać?
- Jak dokumentować architekturę?

Architektura a oczekiwane cechy systemu

1. Efektywność
   - Gruboziarnistość, minimalizacja komunikacji między modułami.
2. Pewność (ang. safety)
   - Lokalizacja krytycznych elementów w małej liczbie komponentów.
3. Bezpieczeństwo (ang. security)
   - Warstwowość, lokalizacja krytycznych zasobów w warstwach wewnętrznych.
4. Dostępność
   - Nadmiarowe komponenty, mechanizmy tolerowania awarii.
5. Zdatność do pielęgnacji
   - Drobnoziarnistość, wymienne komponenty.

Perspektywy architektoniczne

- Punkty widzenia (perspektywy) użyteczne w projektowaniu oraz dokumentowaniu architektury
- Notacja wykorzystywana do opisu modeli architektonicznych?
- Każdy z modeli architektonicznych prezentuje jedną perspektywę systemu.
  - Sposób dekompozycji na moduły
  - Sposób interakcji procesów (w czasie wykonania)
  - Sposób podziału komponentów systemu pomiędzy węzły sieci.
- Zwykle potrzebne są różne punkty widzenia.

Model 4 + 1 (perspektyw) architektury oprogramowania

1. Perspektywa logiczna (ang. logical view)
   - Kluczowe pojęcia w systemie reprezentowane w postaci obiektów lub klas obiektów.
2. Perspektywa procesów (ang. process view)
   - System reprezentowany z punktu widzenia procesów oddziałujących na siebie w czasie wykonania.
3. Perspektywa projektowa (ang. development view)
   - Sposób podziału oprogramowania dla celów implementacyjnych.
4. Perspektywa fizyczna (ang. physical view)
   - Sposób rozmieszczenia komponentów systemu w kontekście sprzętu (węzłów sieci, procesorów).

- Powiązane z użyciem perspektywy scenariuszy (przypadków) użycia (+1)

Wzorce architektoniczne

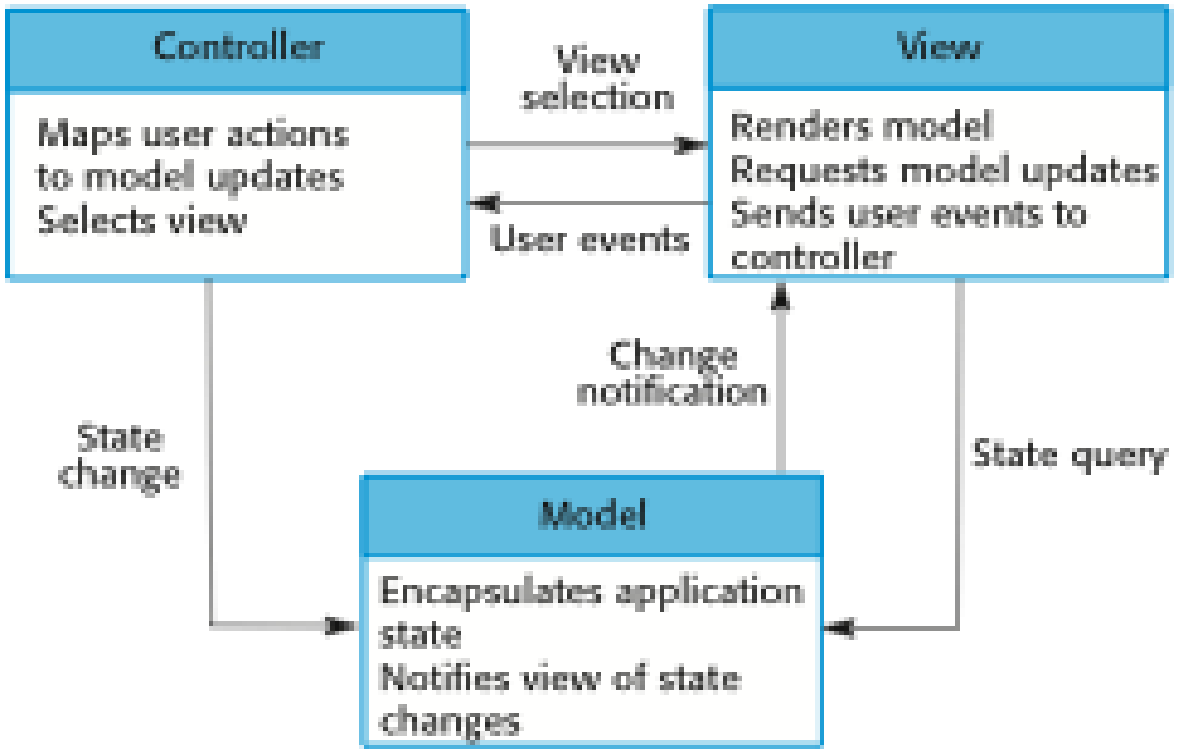
- Wzorce są sposobem reprezentowania, współdzielenia i wielokrotnego użycia **wiedzy**.
- Wzorzec (styl) architektoniczny to stylizowany opis dobrej praktyki projektowej, która została wypróbowana i przetestowana w różnych środowiskach.
- Opis wzorca powinien określać jego stosowalność (kiedy warto a kiedy nie)
- Wzorzec może być reprezentowany w postaci tabelarycznej oraz graficznej.

Wzorzec Model-Widok-Kontroler

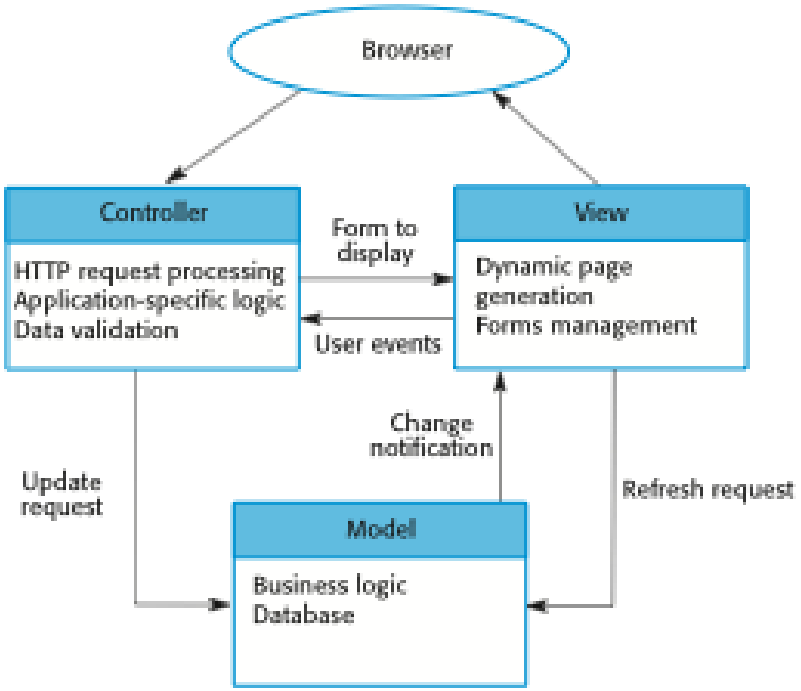
Model-View-Controller (MVC)

Nazwa	MVC (Model-View-Controller)
Opis	Oddziela prezentację oraz interakcję od danych systemu. System podzielony jest na trzy, oddziaływujące na siebie, logiczne komponenty: Model – to komponent zarządzający danymi oraz operacjami na nich wykonywanymi. Widok – to komponent definiujący oraz zarządzający sposobem prezentacji danych użytkownikom. Kontroler – to komponent zarządzający interakcją użytkownika z aplikacją (np. kliknięcie myszą, naciśnięcie przycisku, itp.), który przekazuje interakcję tak do Widoku, jak i Modelu.
Przykład	Następny slajd pokazuje architekturę aplikacji webowej zorganizowanej z wykorzystaniem wzorca MVC.
Stosowalność	Wiele widoków oraz mechanizmów interakcji powiązanych z danymi. Możliwe zmiany w wymaganiach dotyczących sposobu interakcji z danymi oraz metod ich prezentacji.
Zalety	Pozwala na niezależne zmiany w obrębie danych systemu oraz w obrębie ich reprezentacji. Ułatwia prezentowanie tych samych danych na różne sposoby oraz pozwala na zmianę wszystkich widoków w odpowiedzi na modyfikację danych (np. wykonaną za pośrednictwem jednego z widoków).
Wady	W przypadku gdy dane oraz model interakcji użytkownika z aplikacją jest prosty może zwiększać złożoność kodu.

Organizacja wzorca MVC



Architektura aplikacji webowej z wykorzystaniem wzorca MVC



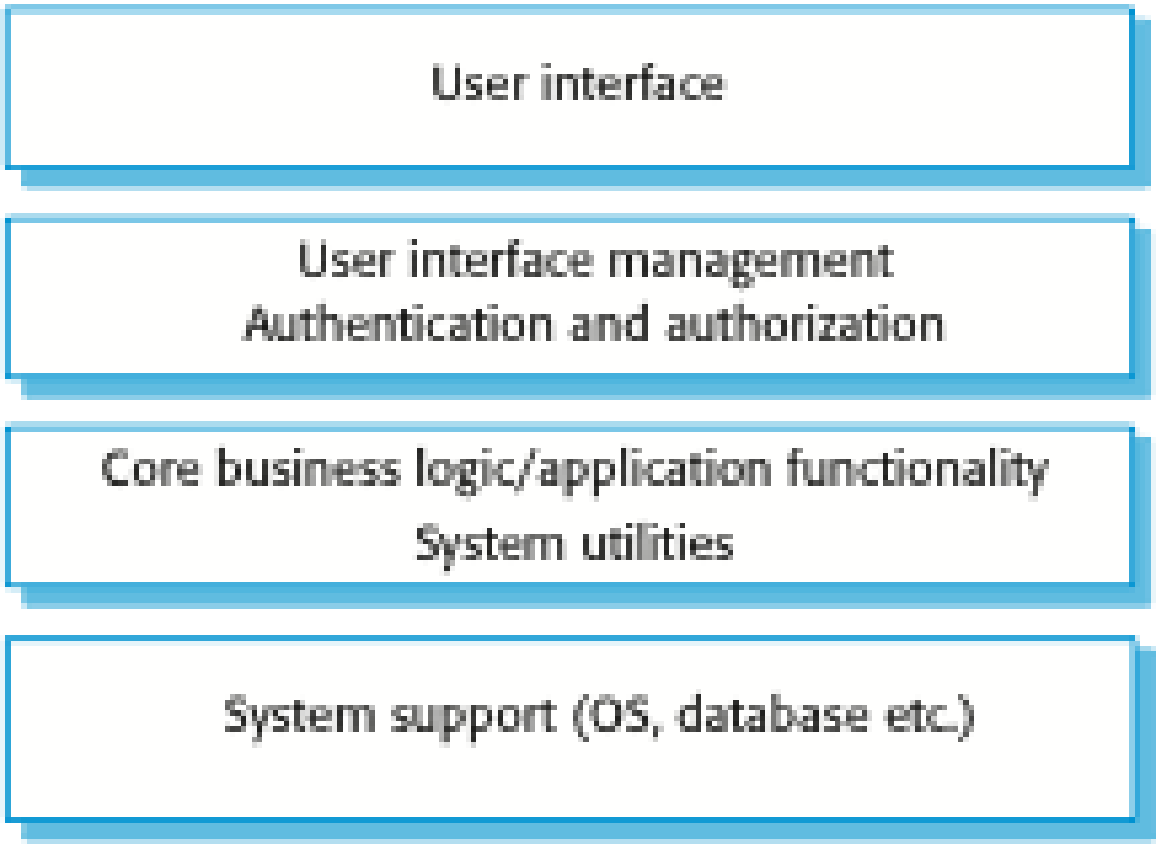
Architektura warstwowa Layered architecture

- Wykorzystywana do modelowania interfejsów podsystemów.
- Organizuje system w zbiór warstw (albo abstrakcyjnych maszyn). Każda z nich udostępnia zestaw usług.
- Ułatwia przyrostową realizację podsystemów w różnych warstwach. Zmiana interfejsu warstwy ma wpływ tylko na warstwy przylegające.
- Często powoduje sztuczną strukturalizację systemu.

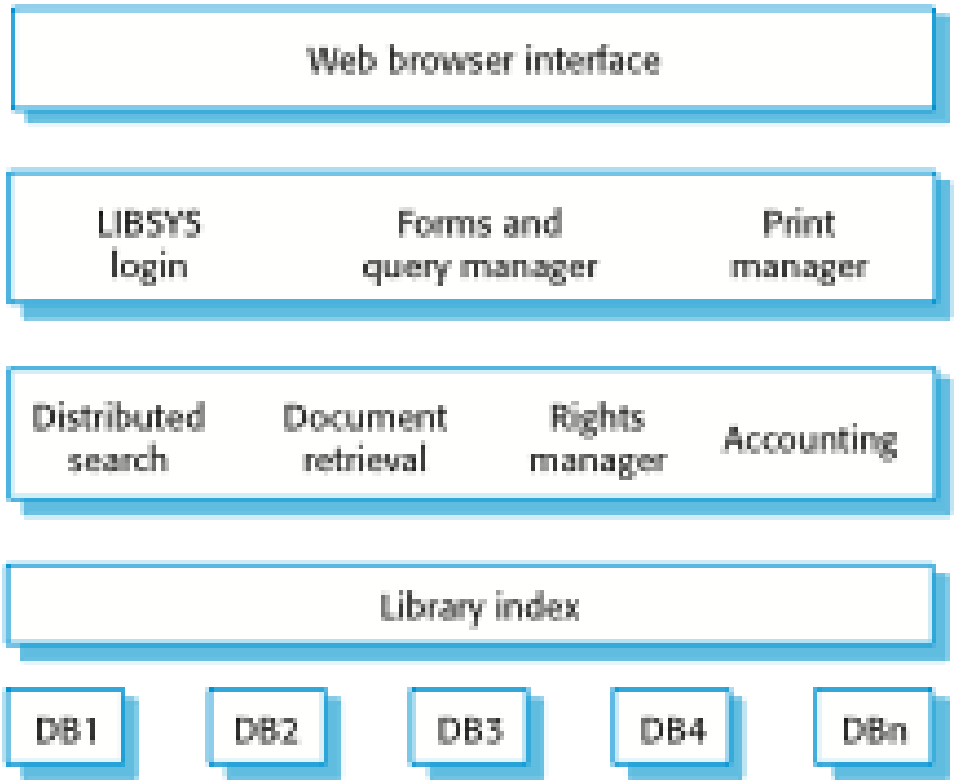
Wzorzec architektury warstwowej

Nazwa	Layered Architecture
Opis	Organizuje system w warstwy. Powiązana funkcjonalność znajduje się w określonej warstwie. Warstwa udostępnia usługi warstwie znajdującej się bezpośrednio nad nią. Warstwa położona najniżej reprezentuje podstawowe usługi, najczęściej wykorzystywane w systemie.
Przykład	Warstwowy model systemu umożliwiający współdzielenie dokumentów chronionych prawami autorskimi, przechowywanymi w różnych bibliotekach.
Stosowalność	Budowanie nowych elementów na „czubku” istniejących systemów. Rozwój jest podzielony pomiędzy odrębne zespoły i każdy z zespołów odpowiedzialny jest za określoną warstwę. Istnieje wymaganie wielopoziomowych zabezpieczeń.
Zalety	Pozwala na podmianę całej warstwy tak długo, jak interfejs pozostaje bez zmian. Nadmiarowe udogodnienia mogą być wprowadzane do każdej warstwy (np. uwierzytelnianie) w celu zwiększenia pewności systemu.
Wady	W praktyce pełna realizacja założeń wzorca jest trudna i może prowadzić do problemów z wydajnością (potrzeba przetwarzania żądania usługi przez każdą warstwę).

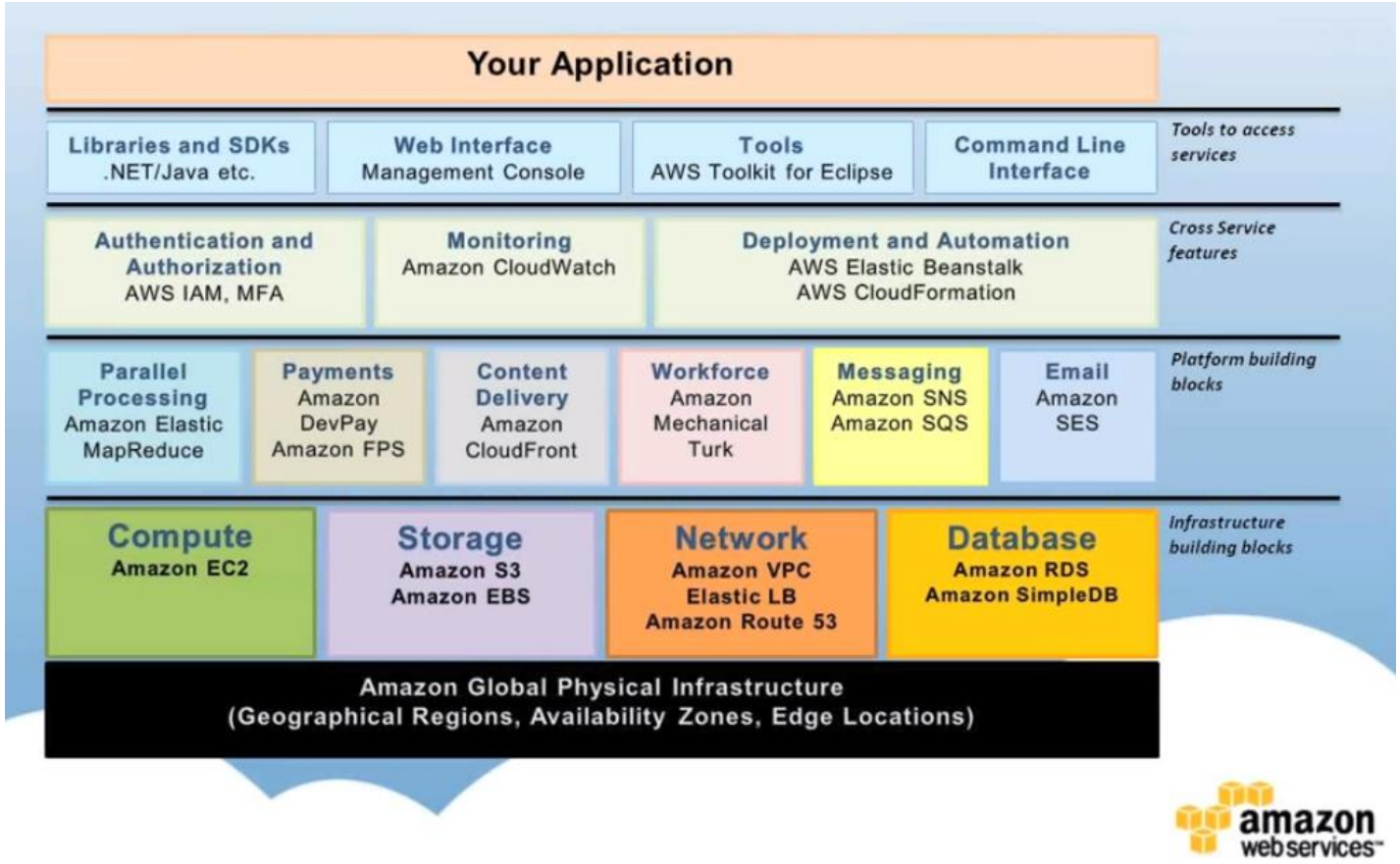
Generyczna architektura warstwowa



Architektura systemu współdzielenia zasobów bibliotecznych



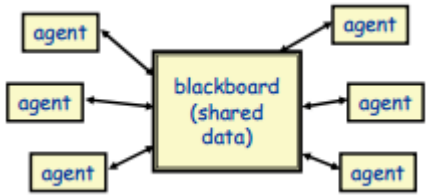
Warstwowa infrastruktura



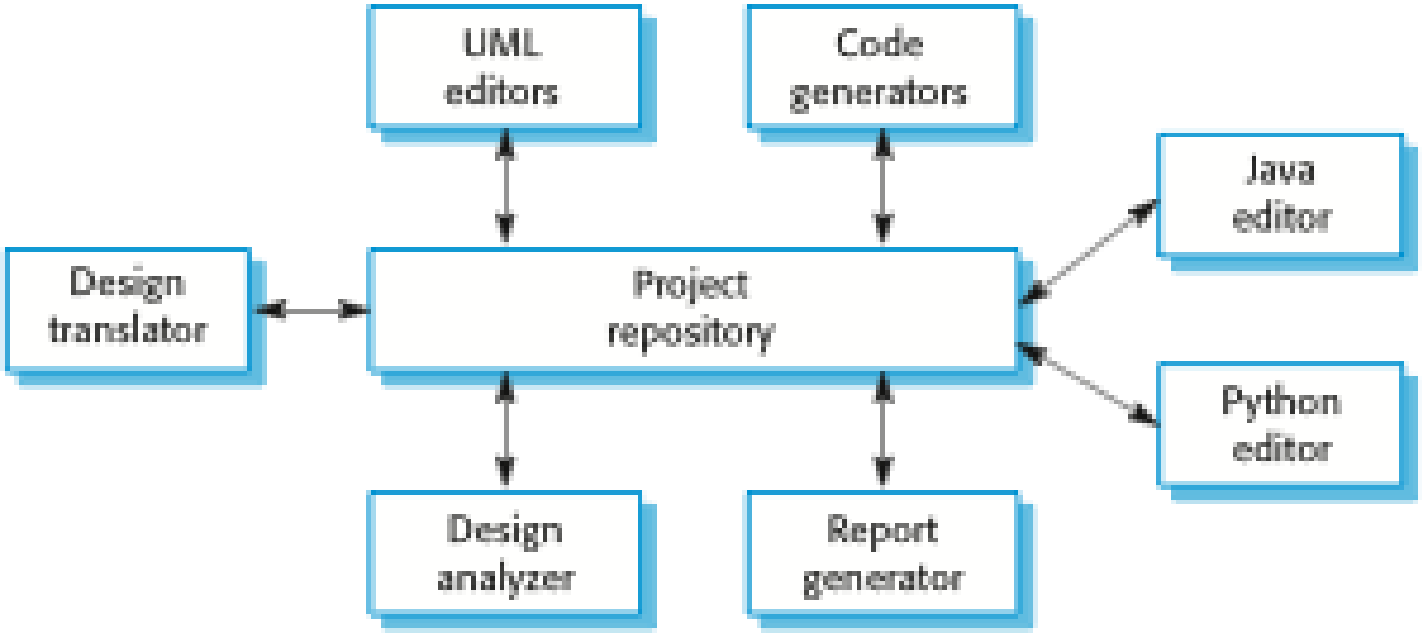
amazon web services

Wzorzec Repozytorium (Blackboard)

Nazwa	Repository
Opis	Zarządzanie wszystkimi danymi systemu odbywa się za pośrednictwem centralnego repozytorium. Repozytorium dostępne jest dla wszystkich komponentów systemu. Komponenty współdziałają ze sobą tylko za pośrednictwem repozytorium.
Przykład	Następny slajd prezentuje przykład IDE, którego komponenty wykorzystują repozytorium informacji o projekcie. Każde narzędzie wchodzące w skład IDE generuje informacje, które mogą być wykorzystane przez pozostałe.
Stosowalność	System, w którym generowane są duże ilości danych, które muszą być przechowywane przez długi czas. Systemy sterowane danymi, w których wprowadzenie danych wyzwala akcje (np. uruchamia narzędzie).
Zalety	Komponenty systemu mogą być niezależne – nie musza wiedzieć o swoim istnieniu. Zmiany wprowadzone przez jeden z komponentów mogą być propagowane do innych. Dane mogą być zarządzane w spójny sposób (np. kopie zapasowe wykonywane w tym samym czasie).
Wady	Repozytorium jest pojedynczym punktem awarii (single point of failure) – problem z repozytorium wpływa na cały system. Organizowanie całej komunikacji za pośrednictwem repozytorium może być nieefektywne. Rozproszenie repozytorium na kilka komputerów może być trudne.



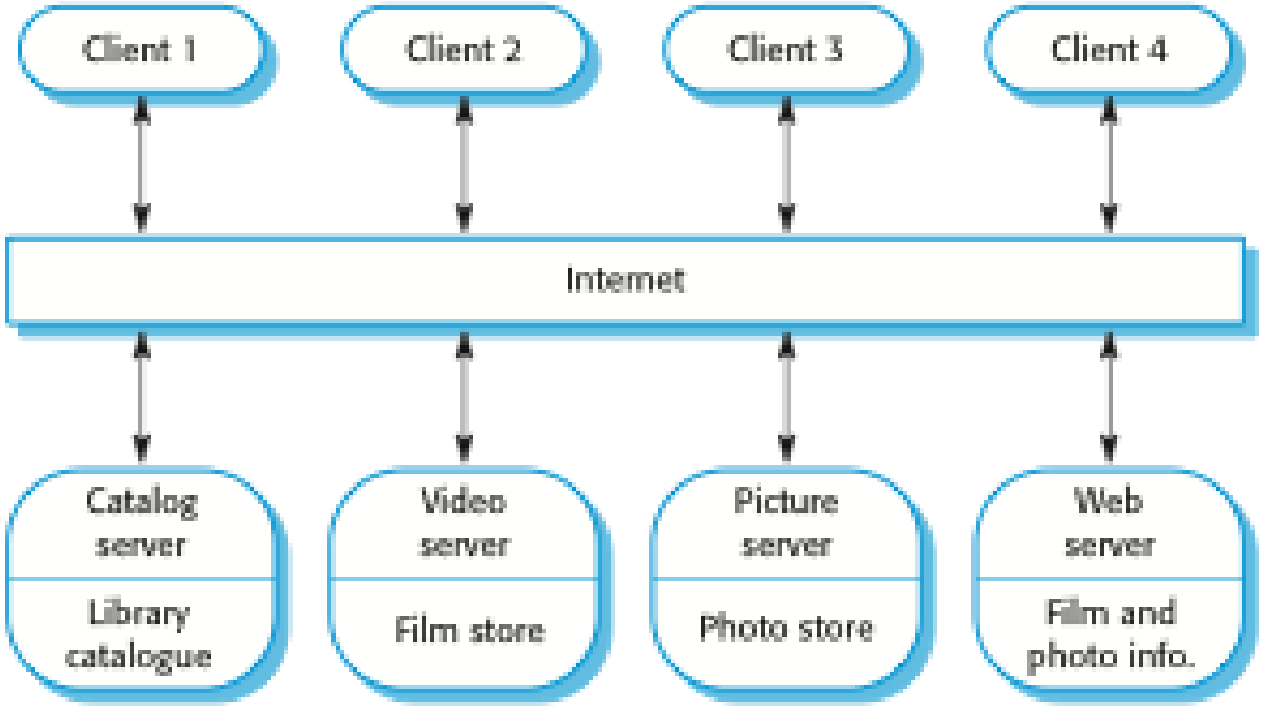
Architektura Repozytorium dla IDE



Wzorzec klient-serwer client–server

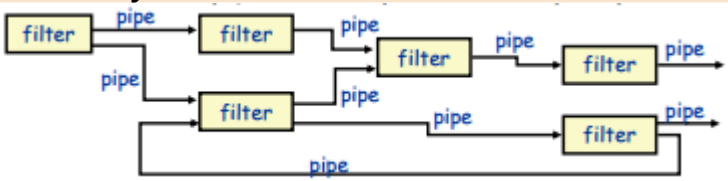
Nazwa	Client-server
Opis	W architekturze klient-serwer funkcjonalność systemu reprezentowana jest przez usługi. Każda usługa udostępniana jest przez odrębny serwer. Klienci są użytkownikami usług i, aby je wykorzystać, uzyskują dostęp do serwerów.
Przykład	Następny slajd prezentuje przykad wypożyczalni filmów wykorzystującej architekturę klient-serwer.
Stosowalność	Dane przechowywane we współdzielonej bazie danych muszą być dostępne z dla różnych lokalizacji. Ponieważ serwery mogą podlegać replikacji, możliwe jest stosowanie w warunkach zmiennego obciążenia.
Zalety	Serwery mogą być rozmieszczone w sieci komputerowej. Ogólne funkcjonalności (np. usługa drukowania) mogą być dostępne dla wszystkich klientów i nie muszą być implementowane przez wszystkie usługi systemu.
Wady	Każda usługa jest pojedynczym punktem awarii. Wydajność może być nieprzewidywalna – zależna jest nie tylko od systemu ale również od sieci. Możliwe problemy z zarządzaniem jeżeli właścicielami serwery są różne organizacje.

Architektura klient-serwer wypożyczalni filmów

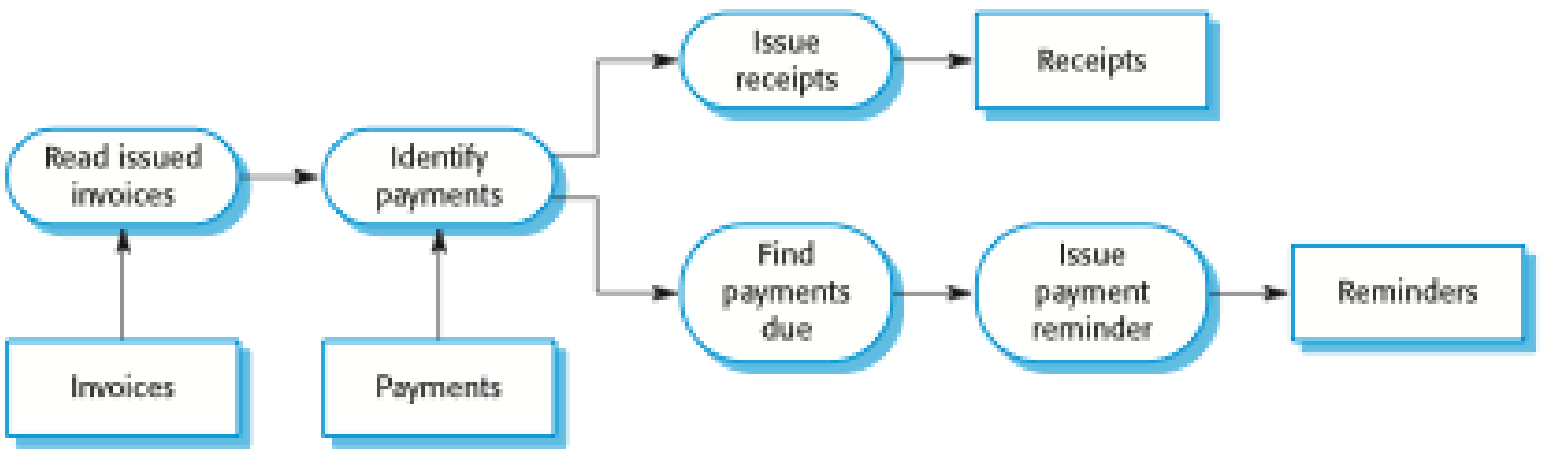


Wzorzec Potoku pipe and filter

Nazwa	Pipe and filter
Opis	Przetwarzanie danych w systemie zorganizowane jest w taki sposób, że każdy komponent przetwarzający dane (filtr) jest niezależny i wykonuje jeden typ transformacji. Dane podczas przetwarzania przepływają od jednego komponentu do kolejnego (jak w potoku, rurze).
Przykład	Następny slajd prezentuje przykład systemu potokowego do przetwarzania faktur.
Stosowalność	Szerokie zastosowanie w systemach przetwarzania danych (tak wsadowych jak i transakcyjnych), gdzie generację danych wyjściowych można oprzeć na serii niezależnych kroków przetwarzania danych wejściowych.
Zalety	Zrozumiałość, łatwość modyfikowania i wielokrotnego użycia. Styl przepływu prac (workflow) odpowiadający strukturze wielu rzeczywistych procesów biznesowych. Ewolucja systemu polega na dodaniu transformacji. Może być zaimplementowana tak sekwencyjnie jak i współbieżnie.
Wady	Komunikujące się transformacje musza mieć uzgodniony format danych. Każda transformacja musi parsować dane wejściowe z- oraz formatować dane wejściowe do- uzgodnionego formatu. Powoduje to zwiększenie obciążenia systemu i może oznaczać, że nie jest możliwe ponowne wykorzystanie transformacji, które używają niezgodnych struktur danych.



Przykład architektury potokowej



Architektura zdarzeniowa

- Model systemu typu bodziec-reakcja, którego działanie opiera się na reakcjach na niezdeterminowane czasowo (asynchroniczne) zdarzenia.
- System składa się z luźno powiązanych komponentów (lub usług).
- Komponenty mogą emitować zdarzenia (agenci, emitenci, wydawcy) oraz je konsumować (konsument, subskrybent).

Ogólne architektury aplikacji

- Systemy są zazwyczaj projektowane w celu zaspokojenia potrzeb określonych organizacji.
- Można jednak znaleźć wiele wspólnych mechanizmów tak niezależnych od typu biznesu jak i zależnych od konkretnej dziedzin.
- Architektura ogólna (generyczna) to architektura danego typu oprogramowania.
  - Wymaga konfiguracji i adaptacji w celu utworzenia systemu spełniającego specyficzne wymagania.

Wykorzystanie ogólnej architektury

1. Startowy punkt do procesu projektowania architektury.
2. Lista kontrolna projektu.
3. Metoda organizacji pracy zespołu.
4. Sposób oceny komponentów wielokrotnego użycia.
5. Słownictwo wykorzystywane w dyskusji na temat typów aplikacji.

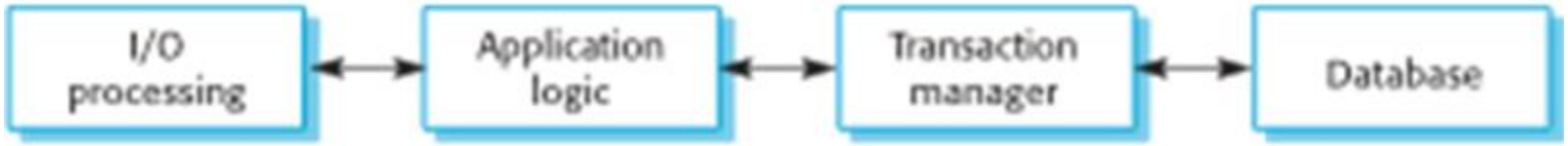
Typy aplikacji

- Systemy przetwarzania danych (wsadów) (ang. Data processing systems)
  - Systemy sterowane danymi, przetwarzające dane w postaci wsadu (batch) bez jawnej interakcji z użytkownikiem w trakcie działania.
- Systemy transakcyjne (ang. Transaction processing applications)
  - Systemy zorientowane na dane, przetwarzające zlecenia użytkowników i aktualizujące informacje w bazie danych systemu.
- Systemy zdarzeniowe (ang. Event processing systems)
  - Aplikacje sterowane zdarzeniami, w których akcje systemu zależą od interpretacji zdarzeń pojawiających się w środowisku działania systemu.
- Systemy przetwarzania językowego (ang. Language processing systems)
  - Aplikacje, w których intencje użytkownika specyfikowane są za pomocą języka formalnego. System przetwarza a następnie interpretuje wyrażenia języka.

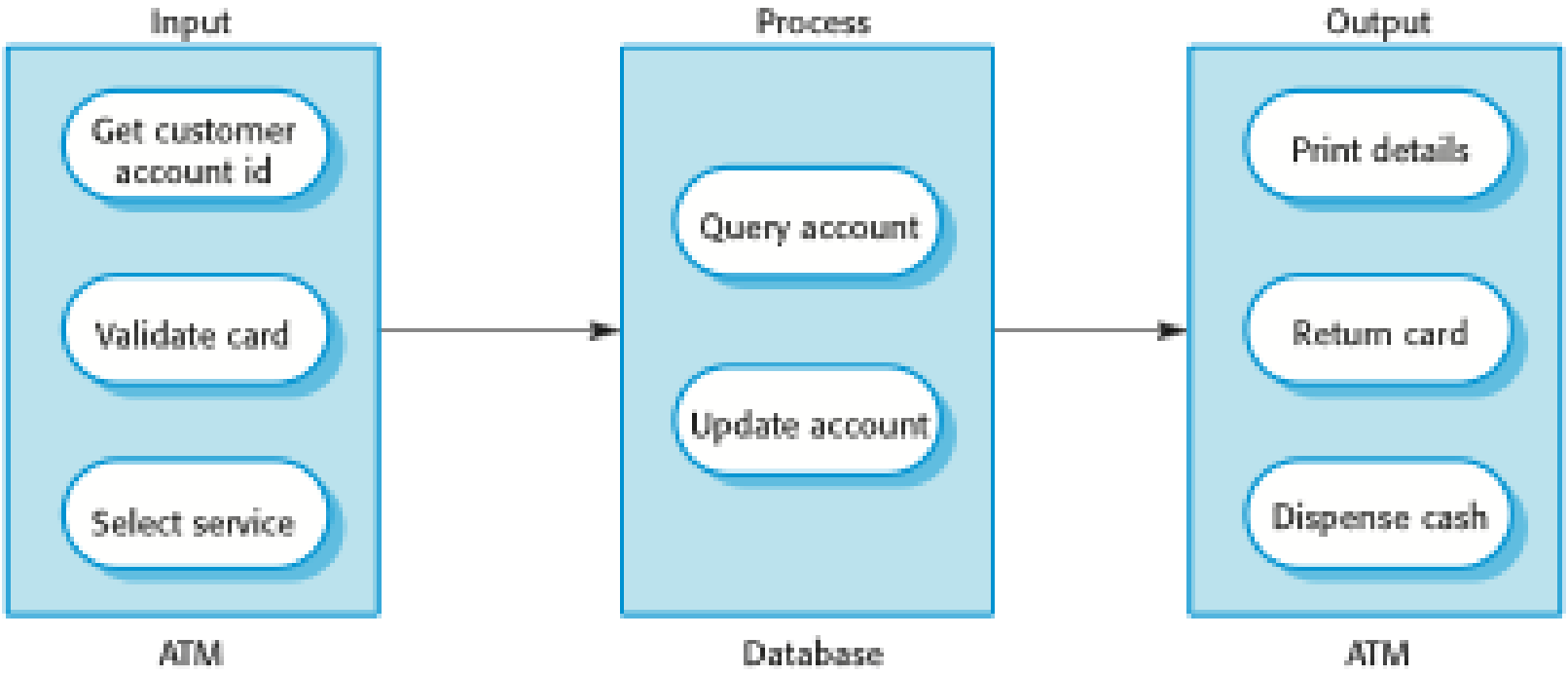
Systemy transakcyjne

- Przetwarzają żądania (ang. requests) użytkowników i aktualizujące informacje w bazie danych systemu.
- Transakcja (z perspektywy użytkownika):
  - Każda spójna sekwencja operacji zaspokajająca potrzebę;
  - Np. znajdź daty i godziny lotów z Londynu do Paryża.

- Zlecenia użytkowników są asynchroniczne. Usługa przyjmuje zlecenie i wykorzystuje tzw. zarządcę transakcji do jej przetworzenia.

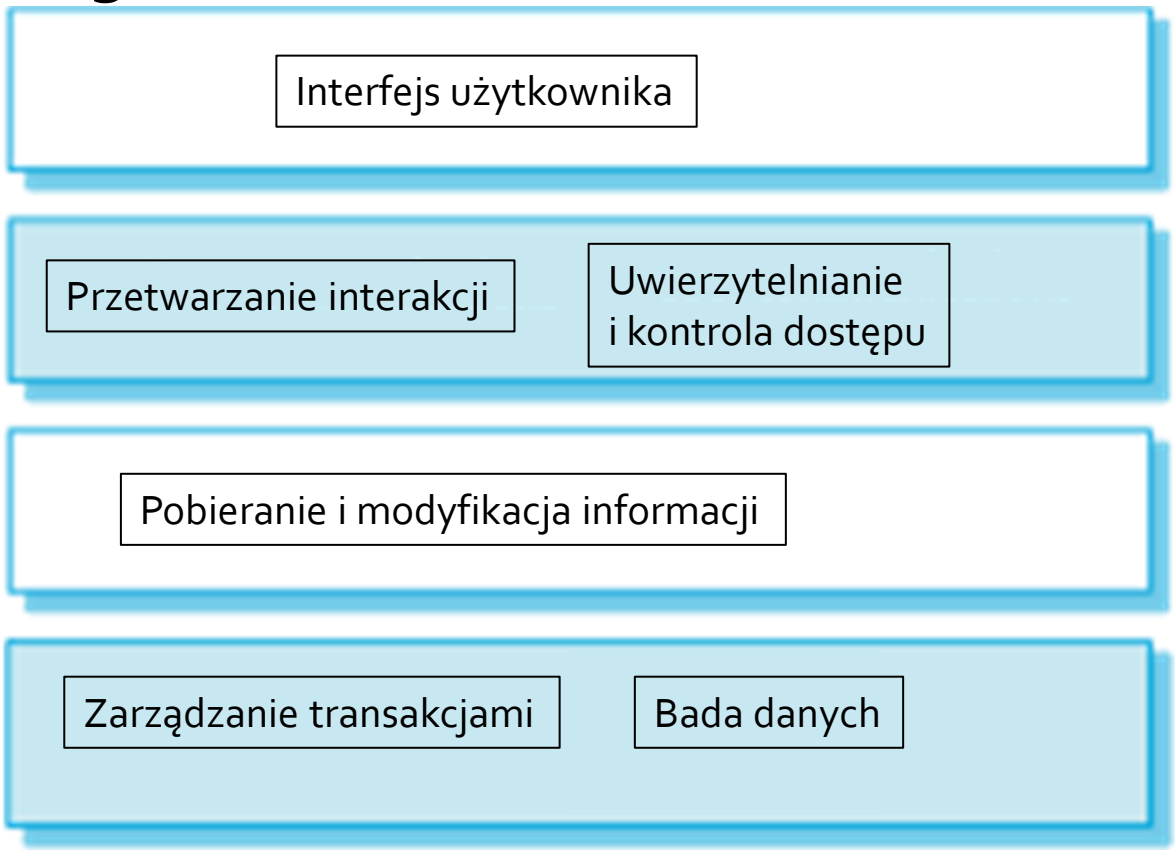


Architektura bankomatu

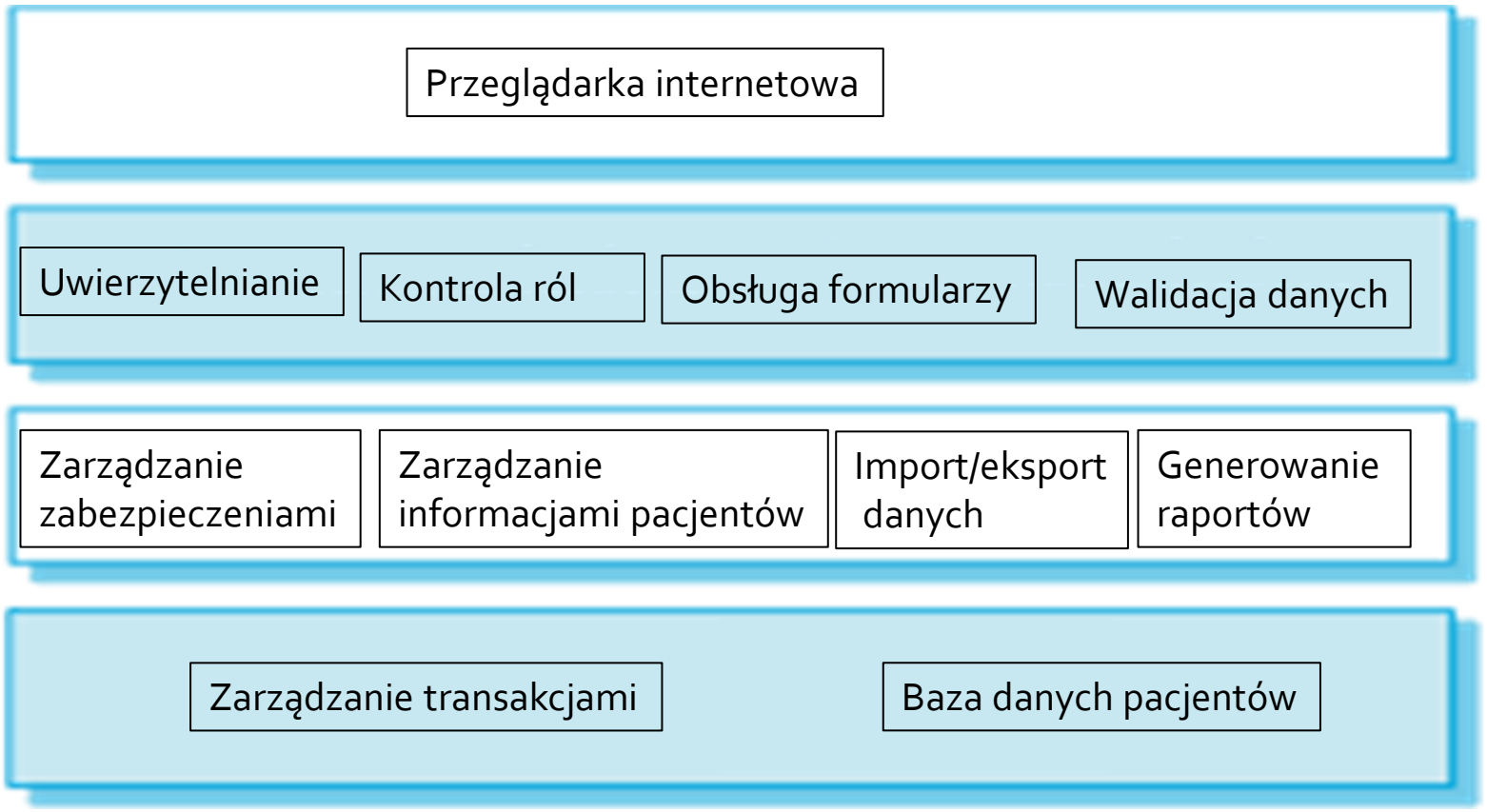


Architektura systemów informacyjnych

- Systemy informacyjne posiadają ogólną architekturę, która może być zorganizowana w postaci modelu warstwowego.



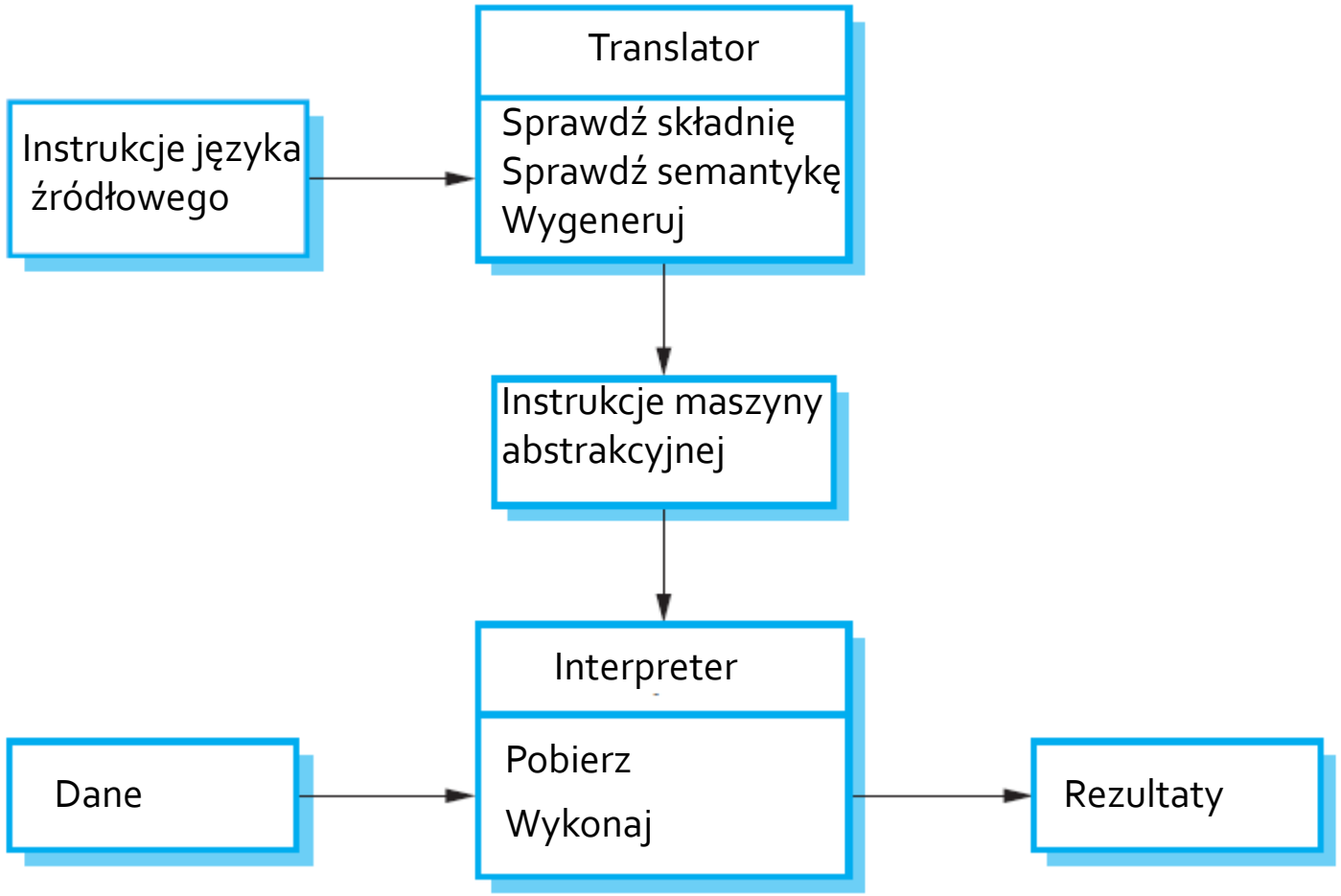
Przykładowa architektura systemu centrum medycznego do zarządzania przyjęciami pacjentów



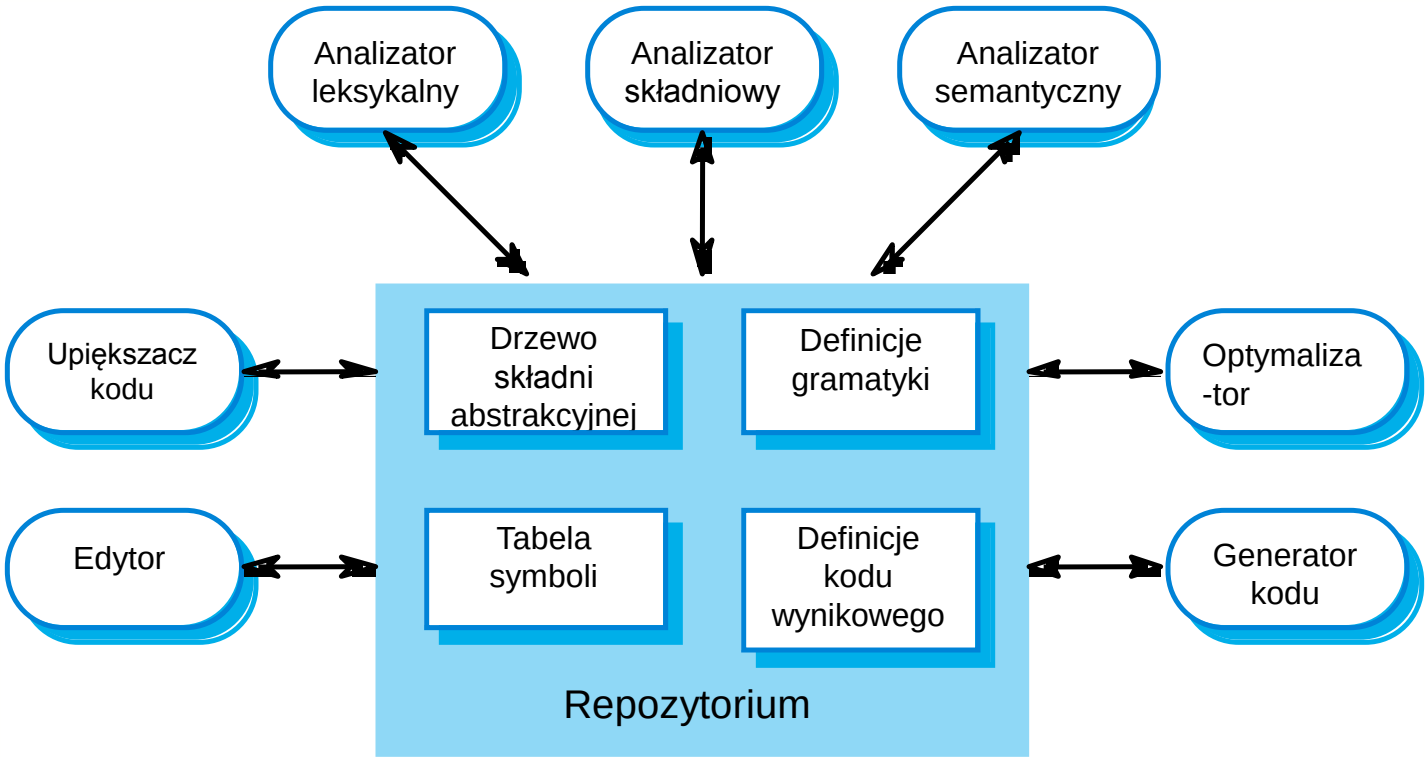
Systemy przetwarzania językowego

- Przyjmuje wyrażenia języka naturalnego lub sztucznego i generuje inna reprezentację tych wyrażeń.
- Może dodatkowo zawierać interpreter działający pod dyktando instrukcji opisanych w przetwarzanym języku.
- Wykorzystywane w sytuacjach, w których najprostszym rozwiązaniem problemu jest opisanie algorytmu jego rozwiązania i/lub opisanie danych systemu.

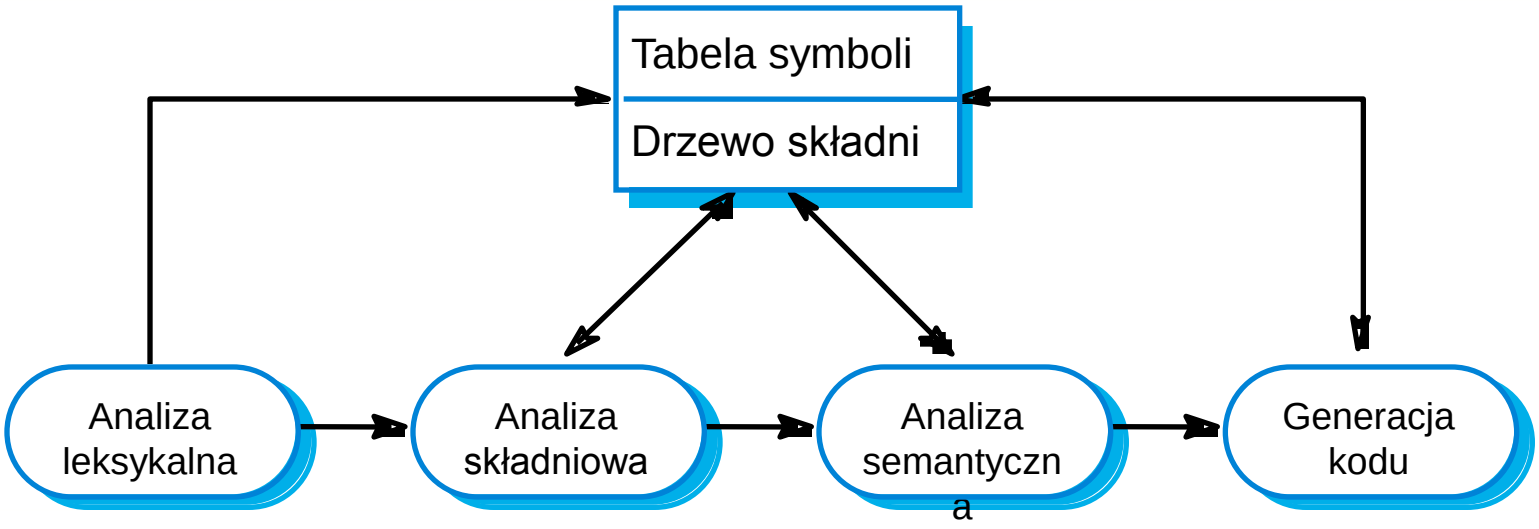
Architektura systemu przetwarzania językowego



Model architektury kompilatora - repozytorium



Model potokowy kompilatora



Podsumowanie (1)

- Architektura oprogramowania jest opisem organizacji systemu oprogramowania.
- Decyzje architektonicznie dotyczą wyboru typu systemu, sposobu jego rozproszenia oraz wykorzystywanego stylu (wzorca) architektonicznego.
- Architektura może być dokumentowana z poziomu różnych punktów widzenia (np. perspektywa: koncepcyjna, logiczna, procesów czy sposobu wdrażania).
- Wzorce (style) architektoniczne to metoda wielokrotnego użycia wiedzy o ogólnych systemach architektonicznych. Styl opisuje koncepcję architektoniczną, wskazuje zastosowanie oraz przedstawia zalety i wady wykorzystania.

Podsumowanie (2)

- Modele architektoniczne systemów pomagają w zrozumieniu oraz porównywaniu aplikacji. Wspierają również ocenę projektu oraz wielkoskalowych komponentów wielokrotnego użycia.
- Systemy transakcyjne ot interaktywne systemy pozwalające na zdalny i równoległy dostęp do danych oraz ich modyfikacje w bazie danych.
- Systemy przetwarzania językowego wykorzystywane są do translacji tekstu zapisanego z użyciem jednego języka, na inny język. W ich skład wchodzi translator oraz abstrakcyjna maszyna pozwalająca na wykonanie wygenerowanych instrukcji języka docelowego.