Warszawska Wyższa Szkoła Informatyki

Seminarium Dyplomowe
Semestr 7

Zajęcia nr 5

Architektura systemu i wybór narzędzi do realizacji projektu

Mgr inż. Jerzy Stankiewicz

ZAJĘCIA NR 1 - ROZLICZENIE

- ***Przygotować harmonogram prac*** *z wykorzystaniem MS Project (za okres październik 2018 : 31-03-2018r.)*
- ***Utworzyć dokument pracy dyplomowej*** *(nazwisko imię v1.docx) ze stroną tytułową, proponowanymi rozdziałami (z wygenerowanym spisem treści)*
- ***Literatura*** *(na końcu dokumentu) – przedstawić propozycje literatury z dziedzin:*
  - *Projektowania systemów informatycznych*
  - *Modelowania systemów*
  - *Projektowania baz danych*
  - *Języków programowania*
  - *Dziedziny tematycznej pracy dyplomowej (normy prawne, dzienniki ustaw itp.)*
  - *Strony internetowe (ćwiczenia, opisy, przegląd produktów rynkowych o podobnej tematyce itp.)*
- ***Opracować rozdział wstępny*** *w zakresie: temat pracy, cel i zakres pracy, wprowadzenie do problemu (ogólne)*

- ***Opracować część analityczną pracy dyplomowej*** *w zakresie:*
  - *szczegółowy opis problemu*

Opracować część analityczną pracy dyplomowej
w zakresie:

- *Wymagania funkcjonalne systemu*
- *Wymagania pozafunkcjonalne systemu*
- *Użytkownicy systemy i dostępne im funkcje*

Zamodelować system z wykorzystaniem:

statycznych i dynamicznych diagramów UML

Zakres tematyczny

- Prezentacja koncepcji rozwiązań
- Techniki komputerowego wspomagania projektowania rozwiązań
- Architektura systemu
- Wybór narzędzi do realizacji projektu

Elementy systemu komputerowego

- użytkownicy (ludzie, maszyny, inne komputery)
- programy użytkowe (kompilatory, edytory, systemy baz danych, gry)
- system operacyjny
- sprzęt (procesor, pamięć, urządzenia wejścia–wyjścia)

Architektura

- ***Architektura*** [gr.] sztuka projektowania i wznoszenia budowli mających oprócz wartości użytkowych także artystyczne. (…) Dzieło architektury winno odpowiadać zamierzonej funkcji, technice, wymaganiom ekonomicznym i estetycznym. (…) (http://encyklopedia.pwn.pl/)

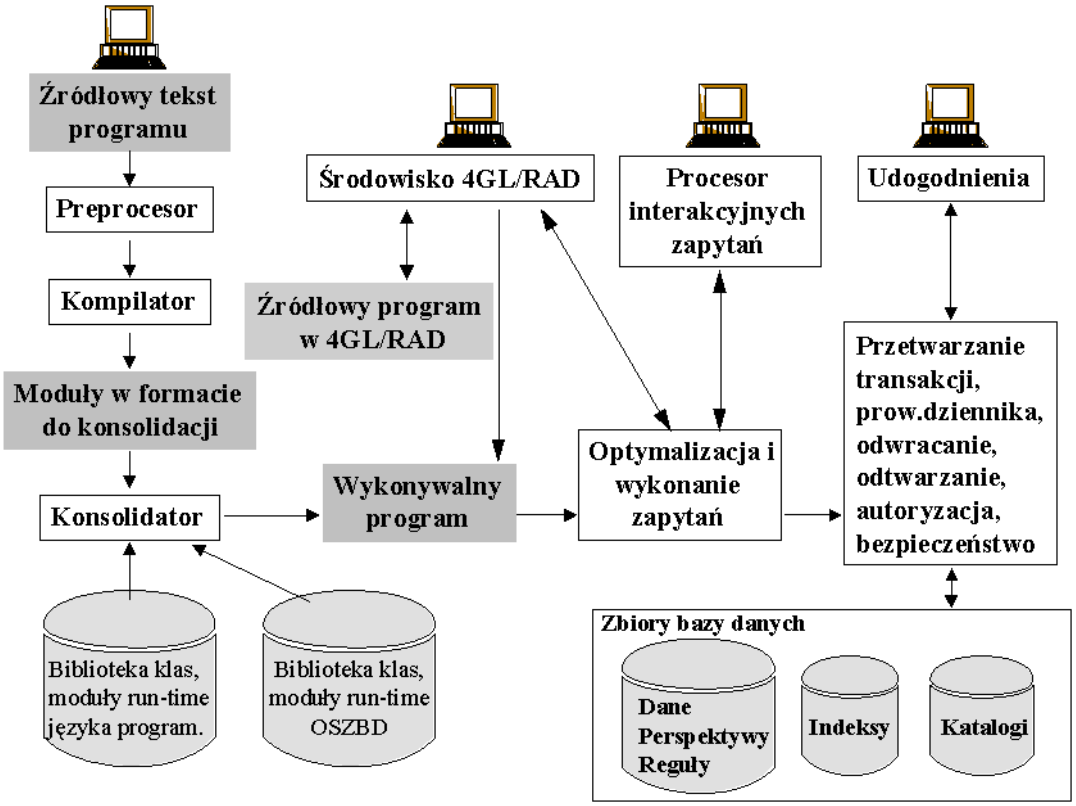
Architektura

- ***Architekt*** musi stworzyć pewien projekt, który ma określone wartości użytkowe i artystyczne. Projekt powinien zapewnić spełnienie wymagań funkcjonalnych, nałożonych przez zamawiających. Projekt architektoniczny powinien być zgodny ze sztuką budowlaną (techniką) oraz zapewniał realizowalność w sensie ekonomicznym. Istotnym elementem architektury są walory estetyczne (jakościowe) obiektu, który na jej podstawie powstanie.
- ***Architekt*** musi zaprojektować budynek, który zadowoli zamawiającego i jednocześnie będzie możliwy do zrealizowania przez wykonawców. Projekt architektoniczny powinien zapewnić wykonawcom (majstrom, murarzom, kierownikowi budowy) wystarczające informacje, aby byli w stanie na jej podstawie zbudować budynek. Projekt architektoniczny jest na tyle ważnym elementem procesu budowlanego, że nikt nie wyobraża sobie zbudowania choćby najmniejszego domu bez niego. Posiadanie projektu architektonicznego jest wymogiem niezbędnym do uzyskania pozwolenia na budowę (budynek stworzony bez planów może stanowić poważne zagrożenie dla jego użytkowników).
- ***Efektem prac architekta*** jest zestaw rysunków pokazujących strukturę budynku na różnym poziomie szczegółowości.
- ***Architektura pozwala przekazać wykonawcom wytyczne dotyczące sposobu wykonania ich dzieła. Żaden wykonawca nie może sobie pozwolić na rozpoczęcie prac nad swoim dziełem bez planów architektonicznych***.

Architektura systemów informatycznych

Architektura (*architecture*) Ogólna, ramowa budowa systemu komputerowego lub oprogramowania, określająca składowe, powiązania pomiędzy składowymi, wzajemne interakcje oraz przepływ informacji. Zwykle w literaturze tym terminem określa się kombinację własności strukturalnych (np. fizycznych komponentów) oraz funkcjonalnych (wewnętrznych i zewnętrznych funkcji systemu).

Niżej zaprezentowano przykładową architekturę obiektowego systemu zarządzania bazą danych.



Architektura oprogramowania

- Jest ***modelem pokazującym strukturę i dynamikę*** działania całego systemu oprogramowania.
- Podstawą działalności architekta jest ***podejmowanie decyzji technicznych i przekazywanie ich wykonawcom systemu w formie modelu***. Model powinien być czytelny dla wykonawców.
- Podstawowym zadaniem architekta jest ***przekształcenie modelu wymagań zamawiającego w model architektoniczny systemu***. Bardzo istotne jest zapisanie tego modelu w notacji znanej wszystkim wykonawcom systemu. Oni odpowiadają za realizację pomysłów architekta. Ta notacja powinna być spójna z notacją używaną później do zaimplementowania systemu, a także jednoznacznie wynikać z notacji używanej do zapisania wymagań.

Architektura oprogramowania

- Poprzez architekturę rozumie się *sposób (styl) konstruowania statycznej struktury systemu informatycznego i samą jego strukturę*.
- **Architektura systemu informatycznego**, czyli to, jak jest on zbudowany, może być rozpatrywana na różnych poziomach szczegółowości i przy uwzględnieniu różnych jej aspektów.
- ***Architektura sprzętowa*** pokazuje na jakim sprzęcie i w jakiej konfiguracji jest (lub będzie) zaimplementowany system.
- ***Architektura konceptualna*** opisuje system w postaci warstw reprezentujących różne poziomy abstrakcji w postrzeganiu go przez zewnętrznych obserwatorów.
- ***Architektura logiczna*** przedstawia podział systemu na logicznie wydzielone części, realizujące odpowiednie funkcje.

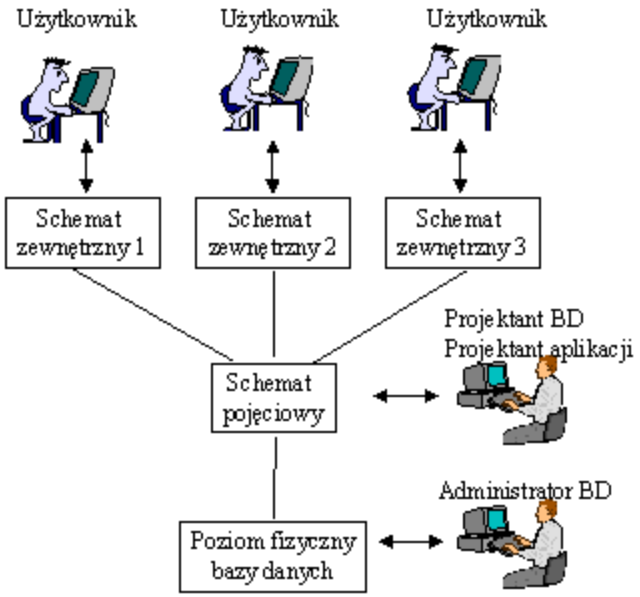
Architektura ANSI/SPARC trójwarstwowa

- Twórcy SI starają się tak zaprojektować architekturę, aby odseparować silnie zależne od technologii i narzędzi programistycznych interfejs użytkownika i bazę danych od logicznych i pojęciowych, odnoszących się bezpośrednio do rozwiązywanego problemu, zagadnień.
- Architektura trójwarstwowa, której standard został wprowadzony w 1978 przez komitet ANSI/SPARC, proponuje podział systemu na trzy poziomy:
  - ***Schemat wewnętrzny*** - jego fizyczną implementację
  - ***Schemat pojęciowy*** - abstrakcyjny model wycinka rzeczywistości (biznesu, firmy) odzwierciedlanej przez system
  - ***Schemat zewnętrzny*** - oraz poziom zewnętrzny, reprezentujący sposoby, w jakie system jest postrzegany z zewnątrz

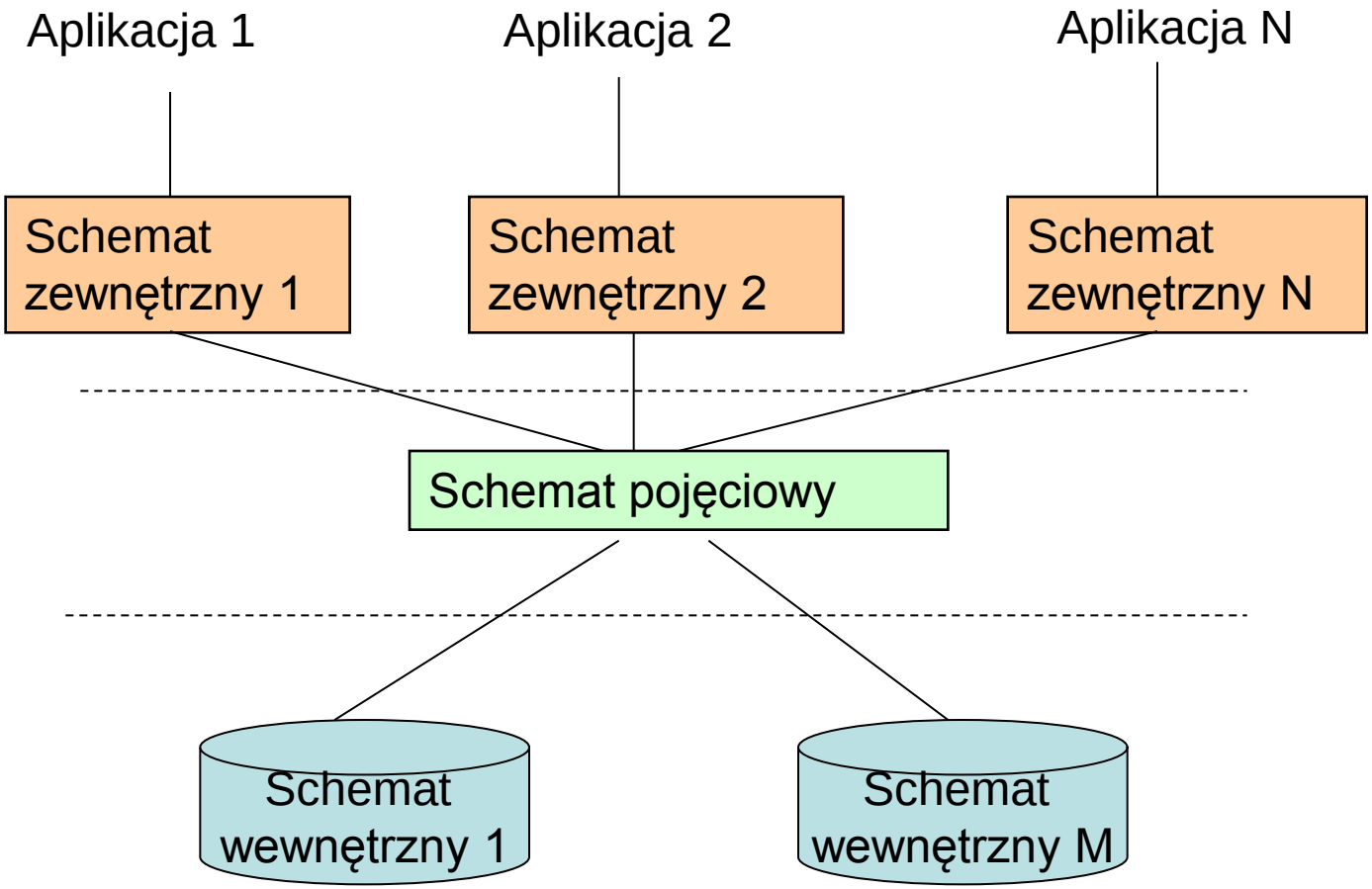
Schemat zewnętrzny

Schemat pojęciowy

Schemat wewnętrzny

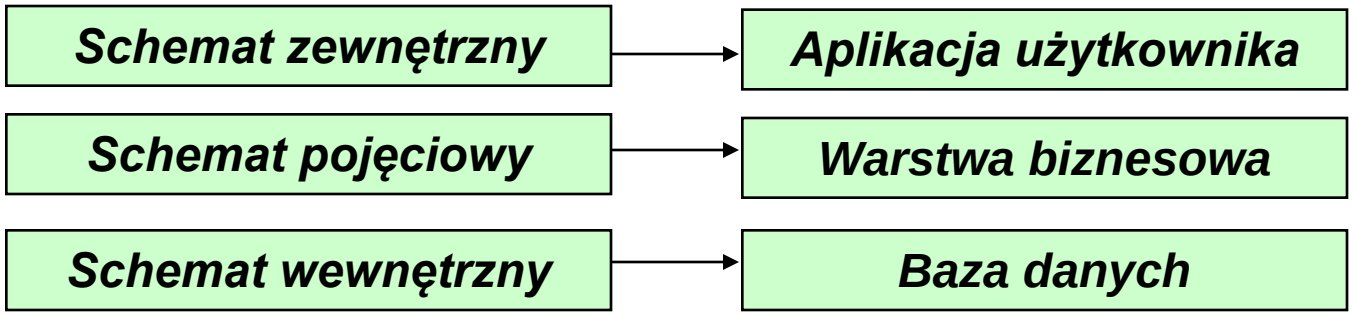


Trójwarstwowa architektura systemu informatycznego



Architektura systemów informatycznych w przedsiębiorstwach

Architektura trzywarstwowa Architektura klient-serwer, która jest podzielona na trzy warstwy: interfejs użytkownika, logikę przetwarzania (reguły biznesu, logikę biznesu) oraz bazę danych. Warstwy te są zaprojektowane i istnieją niezależnie, co ma duże znaczenie dla pielęgnacyjności całego systemu ze względu na możliwość zmian w dowolnej warstwie bez konieczności interwencji w pozostałych warstwach. Często warstwy są zrealizowane na odrębnych platformach: interfejs na MS Windows, logika przetwarzania na serwerze aplikacji i baza danych na serwerze bazy danych. Środkowa warstwa może składać się z wielu warstw, co jest określane jako „architektura wielowarstwowa".



Uwaga: W przypadku architektury ***SI stosowanych w przedsiębiorstwach*** te trzy warstwy: schemat wewnętrzny, konceptualny i zewnętrzny interpretowane są jako: odpowiednio

- **warstwa bazy danych,**
- **warstwa reguł i strategii biznesowych (warstwa biznesowa systemu)**
- **oraz warstwa aplikacji (a właściwie jej interfejsu)**

Architektura oprogramowania

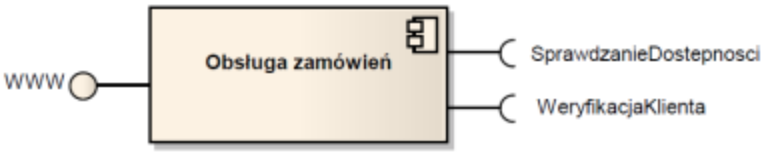
- Projekt architektoniczny systemu oprogramowania jest wyartykułowanym zbiorem decyzji podjętych przez architekta
- Decyzje podejmowane są na ***podstawie wymagań sformułowanych przez zamawiającego oraz na podstawie wiedzy*** o stanie sztuki (znajomości technologii wytwarzania oprogramowania)
- W projekcie architektonicznym ***pokazana jest struktura i dynamika systemu, który ma być zrealizowany***.
- Projekt architektoniczny uwzględnia ***uwarunkowania ekonomiczne i technologiczne***, dając podstawy do ewentualnej zmiany lub rozszerzenia wymagań zamawiającego
- Projekt architektoniczny jest ***zapisany w języku graficznym zrozumiałym dla wykonawców*** (programistów i projektantów systemu oprogramowania)

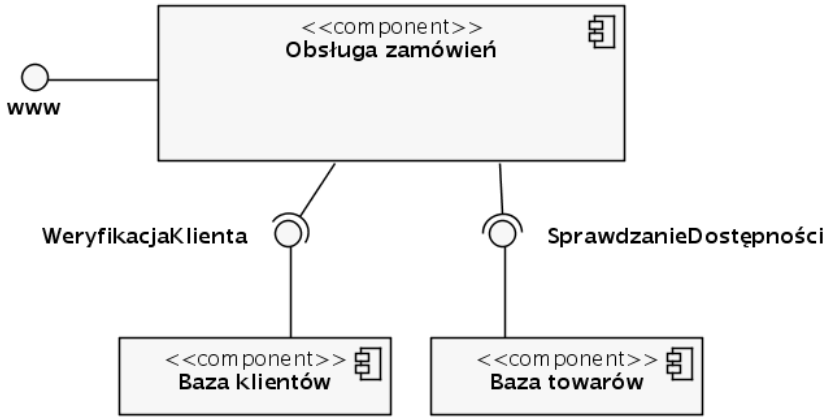
Notacja UML

- Język UML dostarcza kilku notacji, które są przydatne podczas tworzenia modeli architektonicznych.
- Najpowszechniej używane są ***diagramy komponentów i diagramy wdrożenia.*** Służą do wyrażenia struktury logicznej i fizycznej systemu.
- Na ***diagramach komponentów*** można przedstawić jednostki logiczne (komponenty) oraz punkty, poprzez które komponenty się ze sobą porozumiewają (interfejsy i porty).
- ***Diagramy wdrożenia*** służą do przedstawienia jednostek fizycznych systemu. Struktura fizyczna podzielona jest na węzły reprezentujące np. rzeczywiste maszyny (komputery) w realizowanym systemie. Na tych maszynach instalowane są odpowiednie pliki będące fizycznymi elementami oprogramowania. Węzły na diagramach komponentów są połączone odpowiednimi połączeniami wskazującymi na strukturę fizycznej sieci lokalnej lub rozległej.
- Do wyrażenia ***struktury systemu i jego składników*** często przydają się ***diagramy składowych i diagramy klas***. Na diagramach możemy pokazać dekompozycję większych elementów (np. komponentów) na elementy składowe.
- ***Dynamikę systemu*** architekt może zaprezentować za pomocą **diagramów sekwencji. Na takich diagramach pokazujemy działanie systemu jako ciąg komunikatów wymienianych między jego elementami** (komponentami, interfejsami itp.). W wielu przypadkach mogą być również pomocne diagramy opisu interakcji.

Notacja UML – diagram komponentów (1)

Diagramy komponentów przedstawiają (pokazują) podział systemów programowych na mniejsze podsystemy.





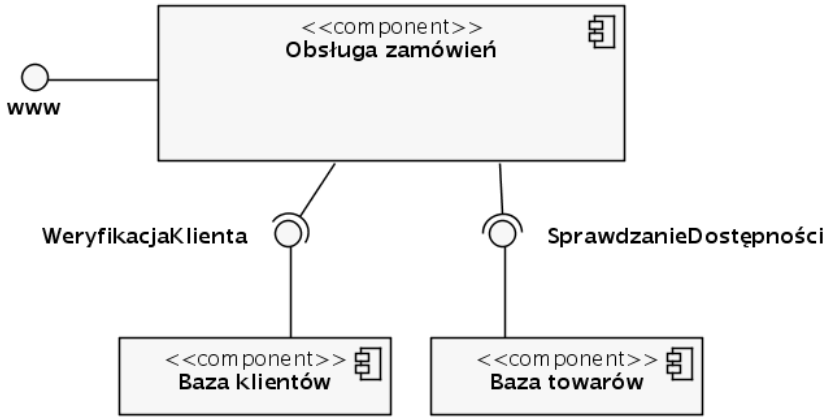
Komponent - reprezentuje jeden fizyczny moduł kodu.

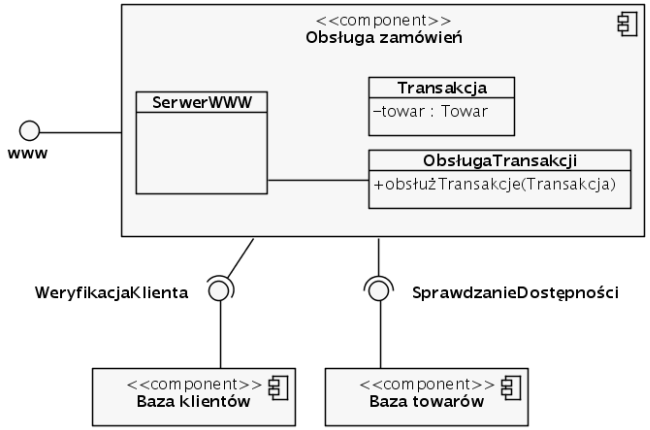
- Komponent udostępnia zestaw interfejsów, może też wymagać pewnych interfejsów do funkcjonowania.
- **Komponent** to wymienny, wykonywalny fragment systemu o hermetyzowanych szczegółach implementacyjnych (np. podprogram) . Komponenty z natury służą do ponownego wykorzystania poprzez połączenie ich z innymi komponentami, zwykle poprzez ich skonfigurowanie, bez potrzeby rekompilacji.
- Funkcjonalność oferowana przez komponent jest dostępna przez interfejsy, które implementuje. Z drugiej strony, komponent może wymagać pewnych interfejsów, które muszą być dostarczone przez inne komponenty.
- Diagram komponentów służy do pokazania związków pomiędzy komponentami i interfejsami

Notacja UML – diagram komponentów (2)

WIDOKI

- ***Widok czarnej skrzynki*** - nie są pokazane żadne szczegóły wewnętrzne danego komponentu.
- ***Widok białej skrzynki*** - pokazujemy szczegóły wewnętrznej budowy komponentu, czyli jego części składowe

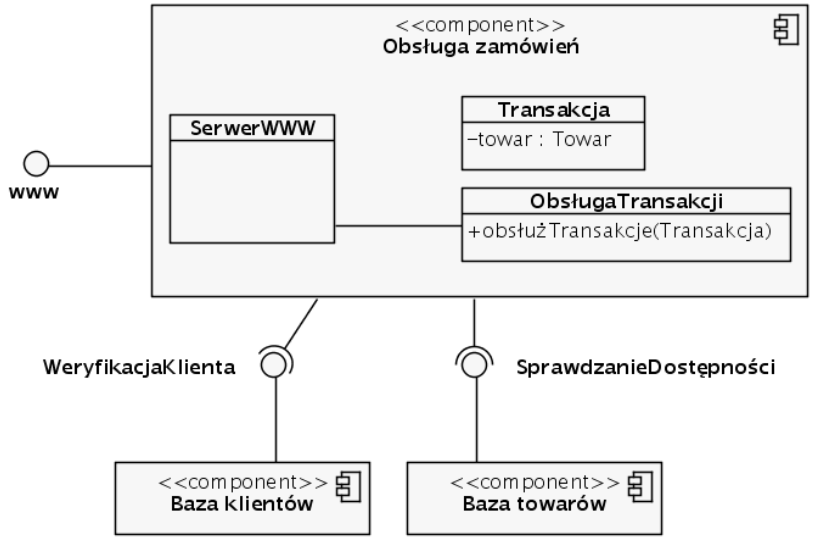




Notacja UML – diagram komponentów (3)

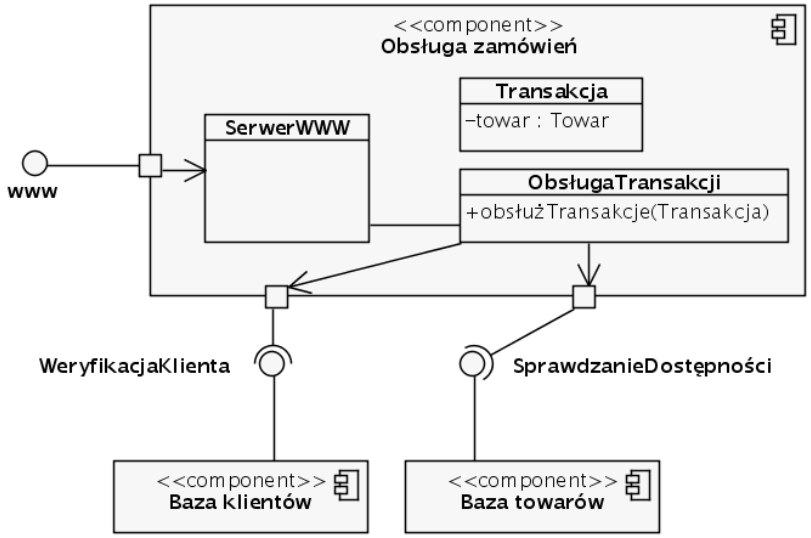
WIDOKI

- ***Widok czarnej skrzynki*** - nie są pokazane żadne szczegóły wewnętrzne danego komponentu.
- ***Widok białej skrzynki*** - pokazujemy szczegóły wewnętrznej budowy komponentu, czyli jego części składowe



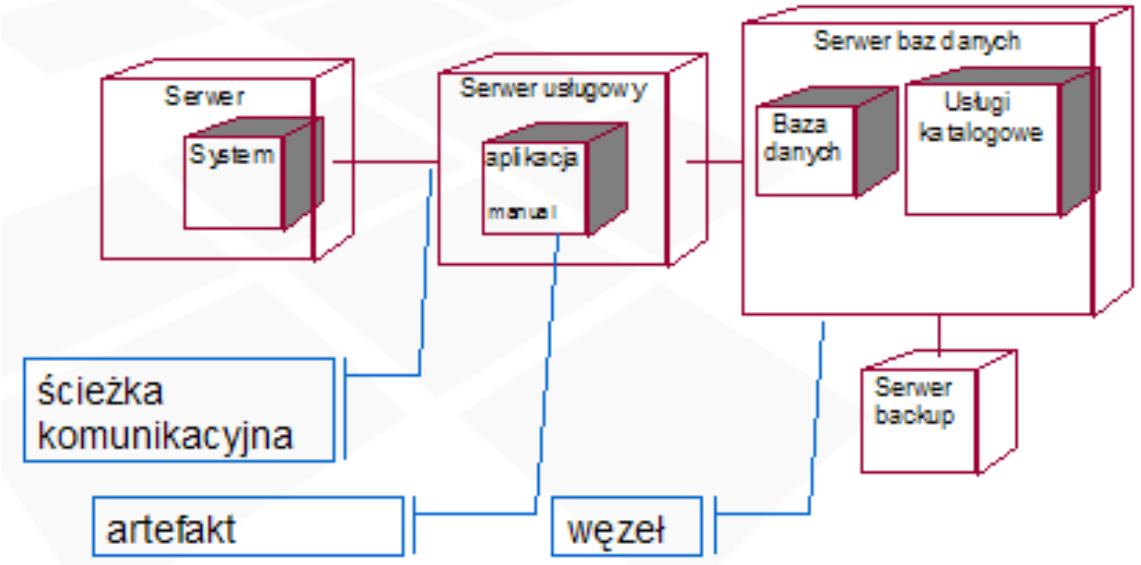
PORTY

Porty - pozwalają łączyć interfejsy wewnętrzne z odpowiedzialnymi za nie fragmentami wewnętrznymi komponentu.



Notacja UML – diagram wdrożenia

Diagramy wdrożenia przedstawiają powiązania miedzy oprogramowaniem (artefaktami) i sprzętem (węzłami). Są stosowane przy modelowaniu duzych systemów



Diagramy - uwarunkowania

- Diagramy tworzone przez architekta są bardziej techniczne – przeznaczone do czytania i stosowania przez wykwalifikowanych projektantów i programistów.
- Elementy opisu używane na diagramach architektonicznych wynikają bezpośrednio z zastosowanej technologii oprogramowania. Zawierają niektóre rozwiązania czytelne jedynie dla osób znających te technologie.
- ***Wszystkie elementy modelu architektonicznego powinny być ściśle powiązane z elementami modelu wymagań***. Jest to warunek niezbędny do zapewnienia zgodności budowanego systemu z wymaganiami zamawiającego.

Struktura – komponenty, interfejsy, klasy, węzły

- Fundamentalną decyzją architekta jest ***podział systemu na logiczne jednostki funkcjonalne***. Stanowią one podstawę dalszych prac nad architekturą.
- **Logiczne jednostki funkcjonalne**, na jakie podzielimy system nazywamy **komponentami**.
- **Komponent** reprezentuje pewną „skrzynkę” zawierającą w sobie elementy realizujące pewną funkcjonalność. Ma cechy pakietu, gdyż zamyka w sobie inne elementy. Dla komponentu określamy pewien sposób zachowania. Komponent potrafi się komunikować z innymi komponentami i dostarczać im usług. **Komponent jest tak naprawdę rodzajem klasy, tylko na wyższym poziomie abstrakcji.**

Komponenty (1)

- Podział na komponenty jest bardzo ważną decyzją architektoniczną. Powinna być ona podjęta na podstawie dobrze określonych przesłanek.
- Pierwszą przesłanką jest ***konieczność zapewnienia, aby komponent realizował dobrze zasadę abstrakcji***. Oznacza to, że komponent powinien grupować w sobie elementy realizujące pewien zwarty podsystem. **Zwartość** (cohesion) komponentów jest bardzo istotną cechą określającą jakość architektury. ***Dobry komponent powinien odpowiadać jakiemuś zbiorowi ściśle związanych ze sobą pojęć (klas) ze środowiska lub elementów wykonawczych (klas), realizujących pewne ściśle związane ze sobą przypadki użycia***.
- Ze zwartością komponentów wiąże się druga przesłanka podziału na komponenty – realizacja zasady ***zamykania informacji***. Oznacza ona, że podsystem realizowany przez komponent powinien być ukryty dla „świata zewnętrznego”. Jednocześnie komunikacja z pozostałymi komponentami powinna się odbywać przez jak najwęższy styk. W ten sposób ***więzy (coupling) między komponentami są stosunkowo słabe***. Większość komunikacji odbywa się wewnątrz komponentów. Komunikacja między komponentami odbywa się tylko wtedy, kiedy jest to niezbędne.

Komponenty (2)

- ***Dobra architektura definiuje podział systemu na komponenty o wysokiej zwartości mającej jednocześnie słabe więzy z innymi komponentami***. Komponenty wymieniają między sobą dane tylko w kluczowych momentach działania poszczególnych przypadków użycia systemu. Cała „*brudna robota*” jest wykonywana wewnątrz poszczególnych komponentów. Jest ona jednak ukryta dla komponentów pozostałych.
- Komponenty nawzajem oczekują od siebie jedynie efektów swojej pracy. Każdy komponent ma zwartą strukturę wewnętrznych elementów (np. „mniejszych komponentów lub klas). Wykonując jakieś czynności, składniki komponentu mogą poprosić o „pomoc” inne komponenty. Mogą to zrobić jedynie „drogą oficjalną”, zwracając się do drugiego komponentu. Nie mogą natomiast „pójść na skróty” i poprosić o pomoc jakiś składnik drugiego komponentu. Komponent po otrzymaniu „oficjalnej prośby” od innego komponentu, zleca wykonanie zadania swoim elementom składowym. Elementy te następnie wykonują jakąś, często bardzo skomplikowaną, czynność związaną z przetwarzaniem danych. Na koniec komponent zwraca komponentowi, który go o to prosił, rezultat przetwarzania.

Zalety podziału systemu na komponenty

- ***Poprawia zrozumienie konstrukcji systemu***. Realizacja zasad abstrakcji i zamykania informacji umożliwia koncentrację na istotnych aspektach systemu.
- ***Ułatwia pracę grupową***, gdyż umożliwia podział pracy. Grupy realizujące swoje komponenty muszą dokładnie znać ich strukturę, ale poza tym wystarcza im znajomość specyfikacji powiązań z innymi komponentami.
- ***Zasada elastyczności systemu***. Architektura podzielona na dobrze wydzielone komponenty może być łatwo rozszerzana o nowe składniki, a także łatwiej poddaje się zmianom. Tego typu architektury wykazują dużą odporność na zmiany konfiguracji sprzętowej systemu – łatwo jest przydzielać komponenty do różnych maszyn fizycznych.
- ***Zapobiega „makaronizmom" w kodzie***. Do innych komponentów trzeba się zwracać jedynie drogą oficjalną – poprzez powiązania między komponentami. Jeśli potrzebne są dodatkowe powiązania – architekt dokonuje analizy zmian i zmienia odpowiednio architekturę.
- ***Ułatwia testowanie ze względu na wyraźnie określone źródła błędów*** (komponenty). Należy zwrócić uwagę, że liczba źródeł błędów jest znacznie ograniczona. W szczególności eliminowane są bardzo trudne do wykrycia błędy wynikające z ukrytych zależności między odległymi fragmentami kodu systemu.
- ***Ułatwia ponowne wykorzystanie (reuse) kodu***. Dobrze zaprojektowane (spójne i zamykające informację) i dobrze napisane komponenty są doskonałymi jednostkami możliwymi do ponownego użycia.

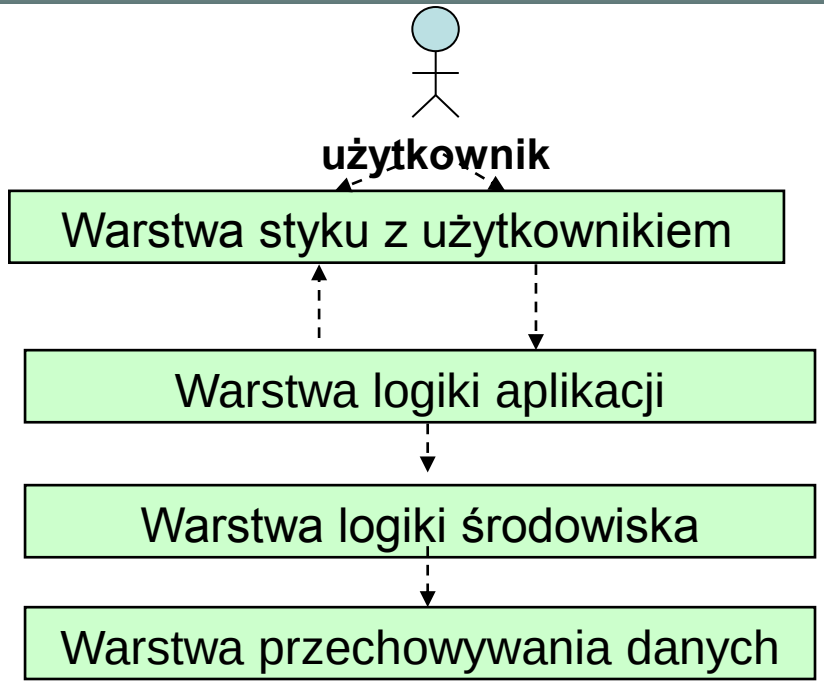
Komponenty

- W języku UML **komponenty** traktowane są podobnie jak klasy – są one pewnym **specjalnym rodzajem klas**.
- Podobnie jak klasa, ***komponent może mieć atrybuty i operacje***.
- Podstawową cechą komponentów jest udostępnianie oraz korzystanie z usług innych komponentów. Funkcjonalność systemu podzielonego na komponenty jest realizowana poprzez ***współpracę wielu komponentów***. Oznacza to, że komponenty są od siebie nawzajem zależne. Architekt, tworząc model architektury na poziomie komponentów, powinien pokazać nie tylko komponenty, ale również odpowiednie relacje (zależności).
- Dobrą praktyką jest stosowanie rozwiązań sprawdzonych już we wcześniejszych projektach.
- Szkielet architektoniczny (wzorzec architektoniczny) jest pewnego rodzaju szablonem, który opisuje strukturę systemu opartego na wybranej technologii czy też wybranej konfiguracji sprzętowej systemu. Może się składać z jednego lub kilku diagramów komponentów.

Model warstwowy

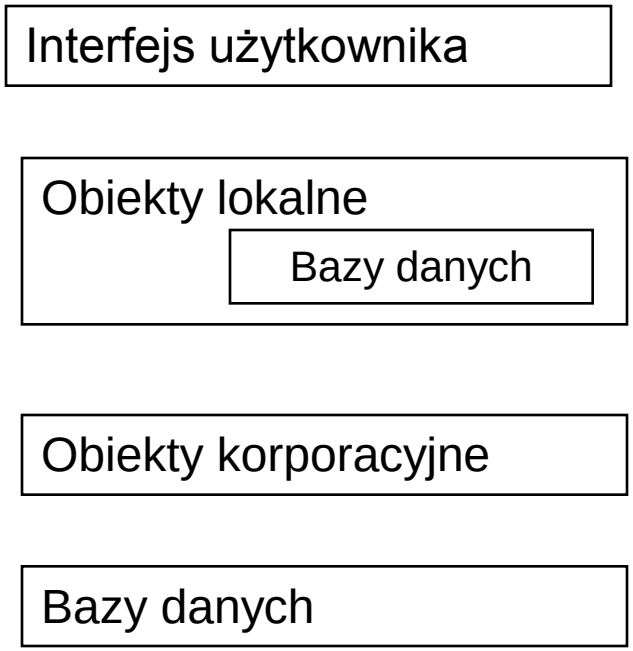
- Znaczna część współczesnych wzorców architektury oparta jest na modelu warstwowym. W modelu tym system dzielony jest na klika warstw (najczęściej trzy lub cztery). Warstwy oznaczają fragmenty systemu o jednakowej „odległości” od użytkownika.
- Warstwa najwyższa jest ***warstwą styku z użytkownikiem***. Umieszczone są w niej komponenty odpowiedzialne za bezpośrednie porozumiewanie się z użytkownikiem. Znajduje się tu zatem menu, okna dialogowe, okna komunikatów itp.
- W następnej warstwie (***warstwa aplikacji***) znajdują się komponenty odpowiedzialne za logikę działania aplikacji. To one powinny wiedzieć w jaki sposób ma się zachować system we współpracy z użytkownikiem. Realizują wszystkie scenariusze przypadków użycia systemu.
- Trzecia ***warstwa szkieletu zawiera logikę środowiska***. W tej warstwie umieszczane są komponent odpowiadające pojęciom środowiska. Tutaj wykonywane są przez system czynności związane z realizacją procesów biznesowych czy też operacje definiowane przez analityków w modelu klas na poziomie wymagań.
- Najniższa warstwa jest ***warstwą przechowywania danych***. W tej warstwie wszystkie dane, które są przetwarzane w warstwach wyższych, umieszczane są w sposób trwały. Najczęściej na tym poziomie znajdują się komponenty bazy danych.

Architektura czterowarstwowa



- Warstwy na diagramie są od siebie zależne – komunikują się ze sobą.
- Zasadą jest, że komunikacja następuje tylko między warstwami sąsiadującymi. Przykładowo warstwa styku z użytkownikiem nigdy nie komunikuje się bezpośrednio z warstwą przechowywania danych.
- Bardzo ważny w szkielecie jest również kierunek komunikacji. Warstwa logiki środowiska nie jest np. zależna i nie uruchamia komunikacji z warstwą logiki aplikacji, natomiast istnieje zależność odwrotna. Wynika to z tego, że czynności na poziomie logiki środowiska nie wymagają uruchamiania jakiejś funkcjonalności na poziomie aplikacji. Warstwa logiki aplikacji musi się natomiast komunikować zarówno z logiką środowiska, jak i ze stykiem z użytkownikiem.

Metodyka The SELECT – architektura czterowarstwowa



Warstwa interfejs użytkownika składa się z ***obiektów interfejsu – okien, menu, przycisków, czytników kart*** itp. Wydzielony on został w osobną warstwę ze względu na dużą zależność od używanej technologii, bibliotek i środowiska. Z tych samych powodów wydzielona została też warstwa bazy danych. Pozostałe warstwy, tworzące logiczną strukturę systemu, to obiekty lokalne i obiekty korporacyjne.

Lokalne obiekty biznesowe – abstrakcyjne rzeczy, ludzi i pojęć z dziedziny problemu (biznesu), występujących lokalnie w danym systemie.

Korporacyjne obiekty biznesowe – obiekty wspólne dla kilku projektów, występujące w wielu systemach. Istnieją one niezależnie od pojedynczego systemu.

Lokalne obiekty biznesowe

- **Lokalne obiekty biznesowe** są odzwierciedleniem ***wymagań oraz reguł biznesowych*** dotyczących ***konkretnego systemu***, wobec czego muszą być izolowane od zależnego od technologii interfejsu.
- Obiekty takie są specyficzne dla danego wycinka działalności organizacji, np. gdy jeden SI służy do obsługi kont, a drugi obsługuje hipoteki, to dla pierwszego ***lokalnym obiektem biznesowym*** będzie ***rachunek***, a dla drugiego ***księga wieczysta***.
- Obiekty lokalne występują tylko w granicach jednego systemu. Mogą być one zarówno obiektami aplikacji, jak i obiektami sterującymi.
- Obiekty tej warstwy często określa się mianem ***obiektów pojęciowych***, ponieważ reprezentują ***pojęcia*** dotyczące kształtu (architektury) pojedynczych procesów biznesowych.
- Bardzo często obiekty te wymagają przechowywania pewnych informacji w fizycznej bazie danych. Lokalna baza danych będzie wobec tego częścią tej warstwy.

Korporacyjne obiekty biznesowe

- **Korporacyjne obiekty biznesowe** zawierają ***funkcjonalność i dane kluczowe dla wielu systemów oraz projektów w ramach firmy*** (organizacji) (są one wykorzystywane przez kilka niezależnych systemów, działających w środowisku firmy).
- Przykładem obiektu korporacyjnego może być ***klient***, występujący zarówno w ***systemie obsługi rachunków, jak i w systemie hipotetycznym***.
- Z punktu widzenia procesów biznesowych, ***obiekty korporacyjne występują w ramach kilku procesów, podczas gdy lokalne obiekty biznesowe są używane tylko w granicach pojedynczych procesów***.
- Obiekty tej warstwy również określa się mianem obiektów pojęciowych, gdyż reprezentują pojęcia dotyczące kształtu (architektury) wspólnych części procesów biznesowych.
- Korporacyjne obiekty biznesowe mogą być zarówno obiektami aplikacji, jak i obiektami sterującymi.

Baza danych

- Czwarta warstwa systemu to baza danych, ***przechowująca niezbędne dla systemu dane***.
- Ponieważ najczęściej przechowywania będą wymagały same obiekty, najlepszym rozwiązaniem jest posiadanie obiektowej bazy danych, umożliwiającej bezpośrednie i automatyczne odwzorowanie obiektów systemu (zarówno korporacyjnych, jak i lokalnych) w tę bazę.
- W przypadku ***baz relacyjnych wymagane jest*** natomiast przeprowadzenie odpowiedniego ***przekształcenia zbioru obiektów w tabele relacyjne***.

Zalety architektury czterowarstwowej

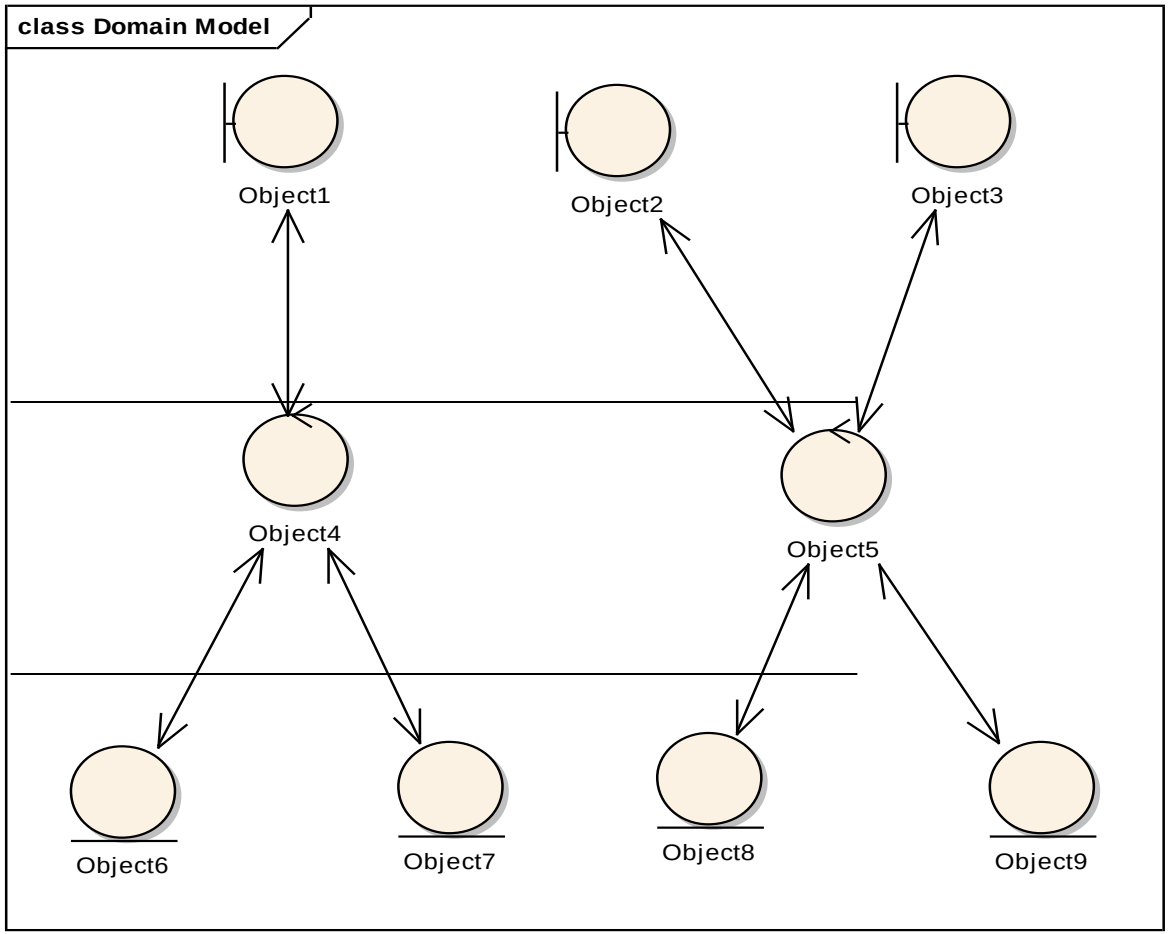
- Podział architektury na cztery warstwy powoduje, że ***chroni się biznesową strukturę systemu, reprezentowaną przez obiekty lokalne i korporacyjne, przed częstymi zmianami jakie zachodzą w interfejsie z użytkownikiem oraz w ściśle związanej z technologią w warstwie bazy danych***.
- Podział obiektów na korporacyjne i lokalne umożliwia lepsze wykorzystanie obiektów w różnych systemach.
- Posiadanie dopracowanego zbioru obiektów korporacyjnych znacznie przyspiesza budowanie nowych systemów, gdyż mogą być one „składane” z istniejących już obiektów.
- W trakcie projektowania i implementacji, stopień uniezależnienia się od zmian interfejsu zostanie zwiększony poprzez podział warstwy obiektów lokalnych oraz korporacyjnych na obiekty aplikacji, reprezentujące bierne elementy architektury, mające najczęściej odzwierciedlenie w bazie danych, i obiekty sterujące, zarządzające zadaniami systemu i pośredniczące w komunikacji między obiektami interfejsu a obiektami aplikacji. Spowoduje to, że wiedza o logicznej strukturze systemu będzie skupiona w obiektach sterujących, natomiast obiekty interfejsu będą klientami tychże, nie wiedząc nic o strukturze logicznej systemu poniżej nich.

Rozdzielenie obiektów interfejsu od obiektów aplikacji (przechowujących

Obiekty interfejsu

Obiekty sterujące

Obiekty aplikacji



Technologia klient/serwer (1)

- **Architektura klient-serwer -** Architektura sprzętu lub oprogramowania, w której występuje podział na część zlecającą pewne usługi (czyli klienta) oraz część wykonującą te usługi (czyli serwer).
- Architektura czterowarstwowa jest szczególnie przydatna, gdy chce się zastosować technologię klient/serwer. Podział aplikacji (systemu) automatycznie dokonuje się na dowolnej granicy między warstwami, w zależności od tego, czy chce się otrzymać w wyniku system o budowie typu gruby klient (*fat client*) z rozbudowaną aplikacją klienta, czy też decyduje się na gruby serwer (*fat server*), gdzie większość operacji wykonywanych jest przez serwer.
- W pierwszym przypadku linia podziału może przebiegać między obiektami korporacyjnymi a korporacyjną bazą danych

  (***klient***: interfejs, obiekty lokalne, baza lokalna, obiekty korporacyjne;
  serwer: korporacyjna baza danych).
- W drugim przypadku po stronie klienta można umieścić jedynie interfejs, pozostałe warstwy implementując na serwerze.
- Inne możliwe podziały to np. między obiektami lokalnymi a obiektami korporacyjnymi – po stronie klienta umieszcza się interfejs użytkownika oraz warstwę lokalnych obiektów biznesowych, natomiast po stronie serwera warstwę obiektów korporacyjnych i warstwę bazy danych.

Technologia klient/serwer (2)

- Inna bardziej subtelna konfiguracja to:
  - interfejs i obiekty sterujące umieszcza się w ***aplikacji klienta***,
  - lokalne obiekty aplikacji, obiekty korporacyjne i bazę danych umieszczając ***w serwerze***.
    Wprowadzenie serwerów lokalnych, działających w ramach jednego systemu i serwerów korporacyjnych, przechowujących dane wspólne dla całego przedsiębiorstwa, daje dodatkowe możliwości podziału.
- Również takie techniki jak DCOM i CORBA, znajdują zastosowanie w systemie o architekturze czterowarstwowej. Jeśli aplikacja o tej architekturze ma być serwerem OLE, to można łatwo to uzyskać poprzez udostępnienie metod obiektów lokalnych innym aplikacjom – wówczas mogą one, omijając interfejs, korzystać z praktycznie całej funkcjonalności systemu. Można również zaimplementować wnętrze systemu w ten sposób, aby poszczególne warstwy (jako oddzielne aplikacje) komunikowały się między sobą za pomocą tych mechanizmów.

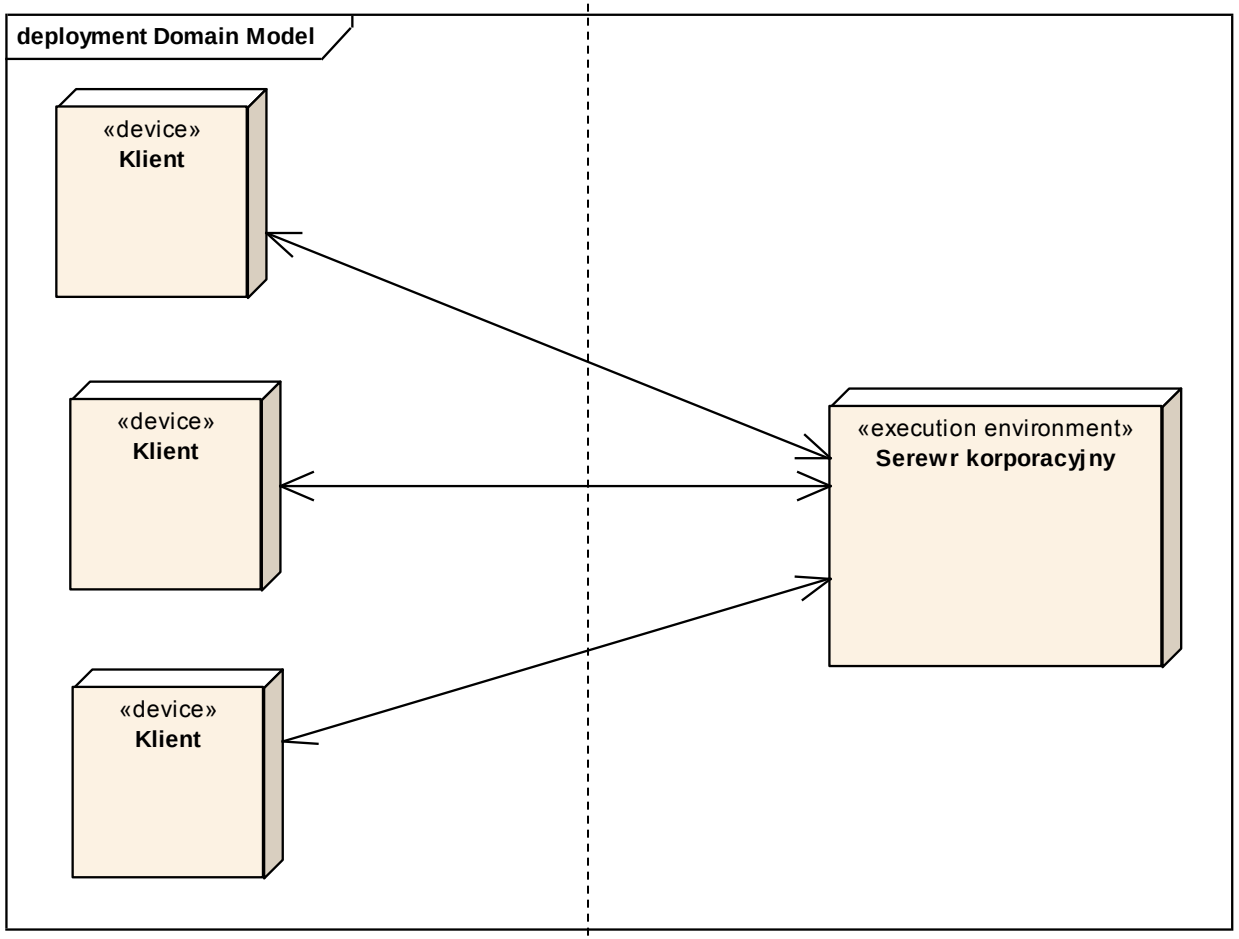
Trzy wersje klient/serwer:

- wszystkie aplikacje wykonywane są przez serwer a wyniki wyświetlane na ekranie klienta
- serwer dostarcza danych dla aplikacji uruchamianych na komputerze klienta
- wszystkie komputery współpracują ze sobą jak równy z równym (peer-to-peer), korzystając wzajemnie ze swoich zasobów

Przykładowy podział aplikacji w technologii klient/serwer

Obiekty interfejsu,
Obiekty lokalne,
Lokalne bazy danych

Obiekty korporacyjne,
korporacyjna baza danych

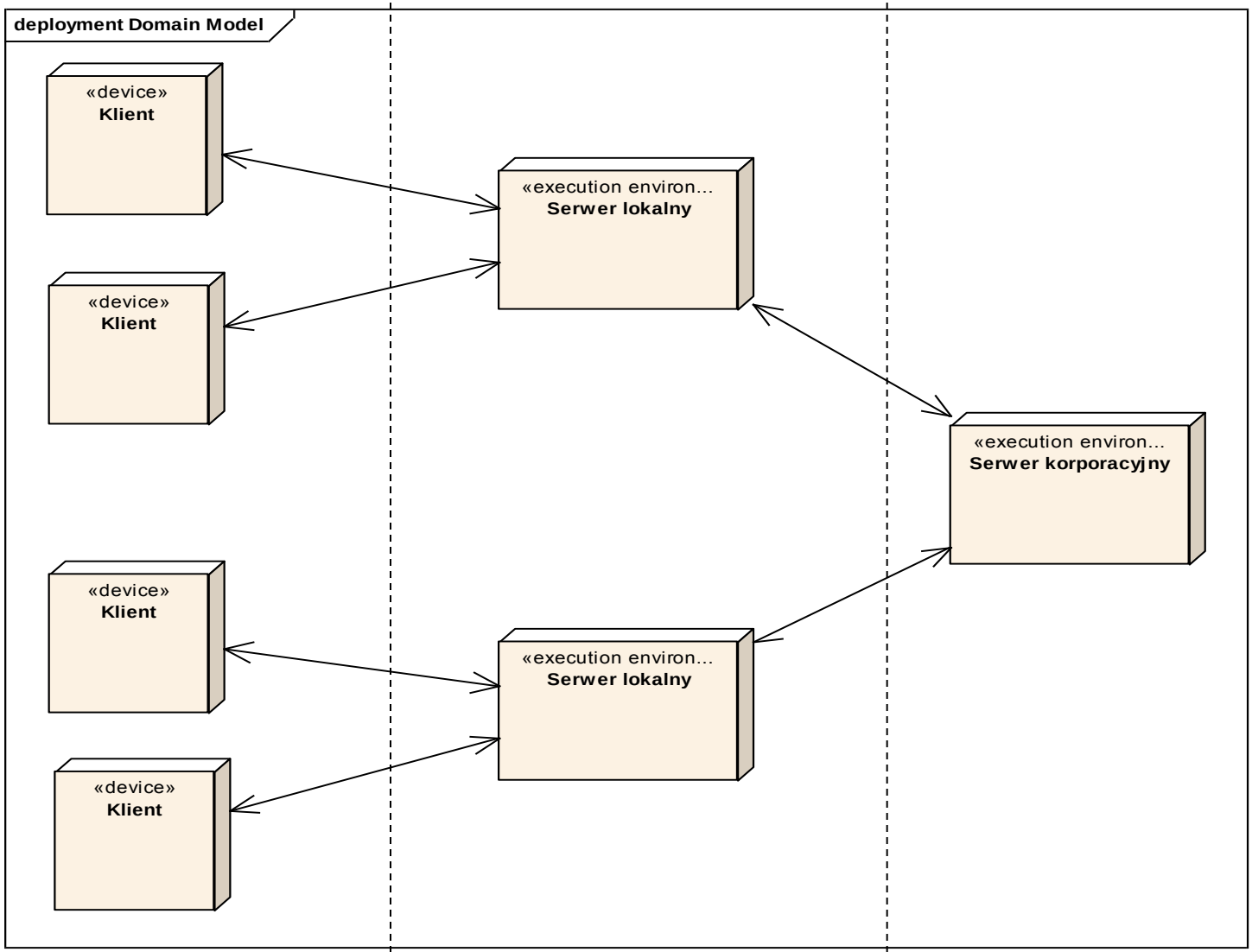


Przykładowy podział aplikacji w technologii klient/serwer

Obiekty interfejsu

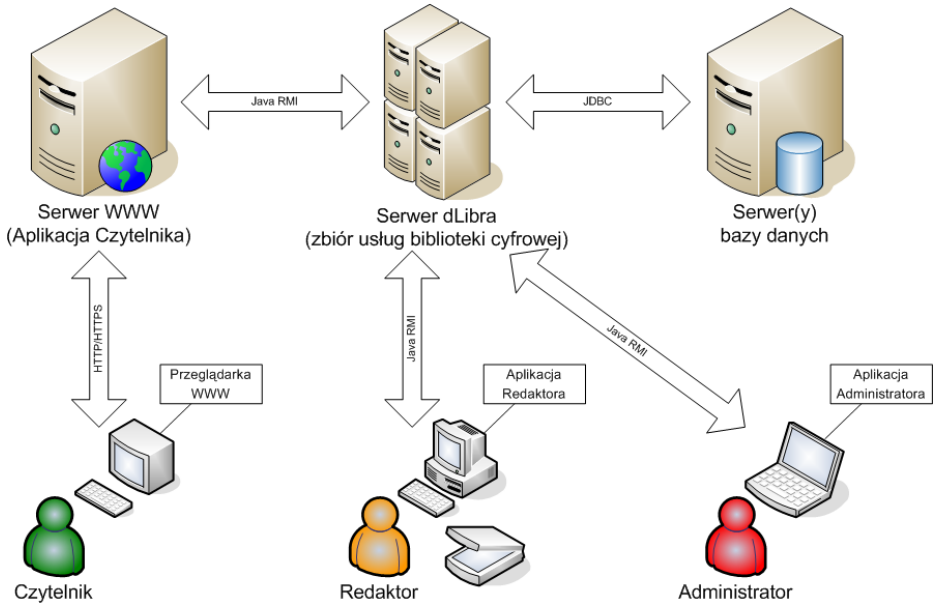
Obiekty lokalne, Lokalne bazy danych

Obiekty korporacyjne, korporacyjna baza danych

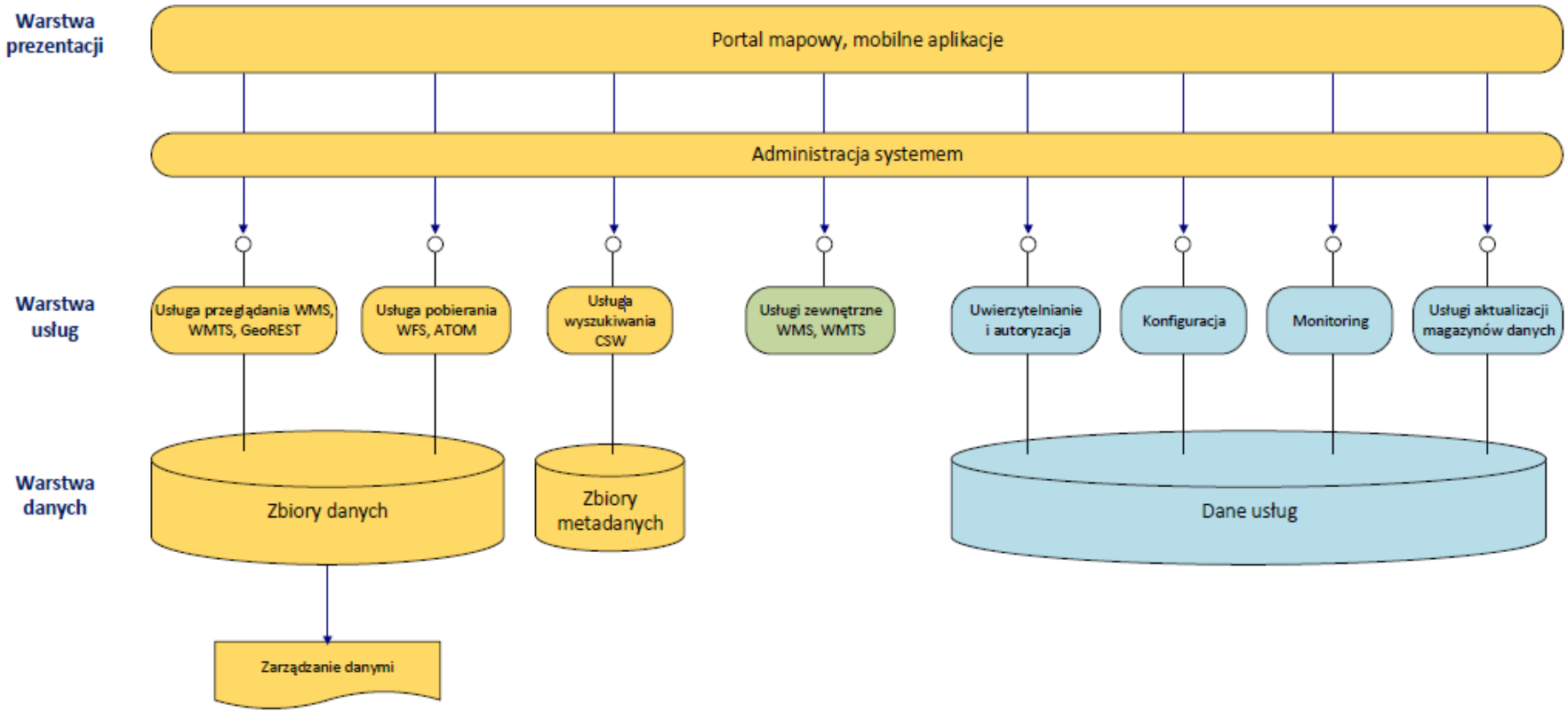


Przykłady

- System *X* jest systemem wielowarstwowym, wykorzystującym do połączeń pomiędzy poszczególnymi komponentami (warstwami) sieć komputerową.
- Poniższy rysunek schematycznie przedstawia architekturę systemu. Taka architektura umożliwia rozmieszczenie poszczególnych komponentów systemu (czyli serwera:
  - aplikacji do zarządzania biblioteką,
  - aplikacji WWW,
  - bazy danych i archiwum publikacji) na różnych komputerach.
- Ważne jest jedynie, aby były one połączone między sobą siecią komputerową umożliwiającą komunikację przy użyciu protokołu TCP/IP. Oczywiście, możliwe jest zainstalowanie i uruchomienie wszystkich komponentów systemu na jednym komputerze.

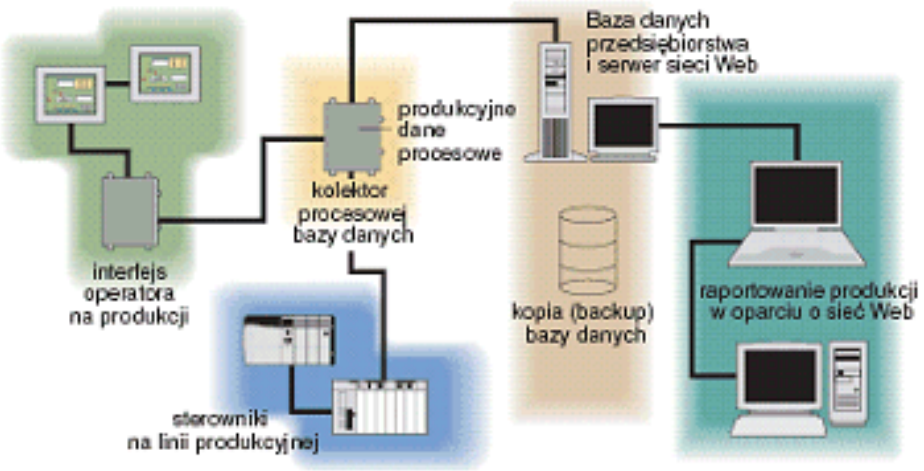


Przykłady


Warstwa prezentacji
Portal mapowy, mobilne aplikacje
Administracja systemem
Warstwa usług
Usługa przeglądania WMS, WMTS, GeoREST
Usługa pobierania WFS, ATOM
Usługa wyszukiwania CSW
Usługi zewnętrzne WMS, WMTS
Uwierzytelnianie i autoryzacja
Konfiguracja
Monitoring
Usługi aktualizacji magazynów danych
Warstwa danych
Zbiory danych
Zbiory metadanych
Dane usług
Zarządzanie danymi

Przykłady

Architektura systemu informatycznego obsługi produkcji



Źródło: Control Engineering z danymi Program4 Engineering Inc.

- *Powyższy schemat pokazuje podstawową architekturę produkcyjnego systemu informatycznego, zbudowaną na bazie danych procesów firmy, serwera sieci Web oraz kolektora danych z procesów*
- Podstawowe kryteria projektowe dla produkcyjnego systemu informatycznego obejmują: udokumentowany kod, utrzymanie ruchu zakładu, własność klienta, otwartą dołączalność bazy danych oraz konieczność zastąpienia produktów w przyszłości. Opierając się na tych kryteriach trzy podstawowe moduły konstrukcyjne systemu to: serwer procesowej bazy danych, kolektor danych procesowych oraz raportowanie oparte na sieci Web.

Wymagania do zajęć nr 5

- ***Przedstawić (opisać , zamodelować) architekturę systemu informatycznego***
- ***Zaproponować narzędzia do realizacji projektowanego systemu (język programowania, baza danych, inne) i umotywować ich zastosowanie***

Seminarium Dyplomowe

Dziękuję!!!!! …… Tyle na dzisiaj