Inżynieria Oprogramowania

Modelowanie Systemu Informatycznego

prezentacja 8

Modelowanie struktury – diagram komponentów i diagram wdrożenia

wersja 1.0

dr inż. Paweł Głuchowski

Wydział Informatyki i Telekomunikacji, Politechnika Wrocławska

Treść prezentacji

1

Diagram komponentów

Diagram komponentów /component diagram/

- Modeluje strukturę oprogramowania na wyższym poziomie abstrakcji, niż diagram klas.
  - **System składa się z komponentów:**
    - są niezależne od siebie (niezależnie kompilowane i niezależnie implementowane),
    - można je rozbudowywać,
    - można je wymieniać na inne,
    - można je używać w różnych systemach,
    - udostępniają swoją funkcjonalność przez interfejsy i porty.

Diagram komponentów

Komponent /component/

- Klasyfikator ze stereotypem **«*component*»** lub ikoną komponentu.
- **Logiczna część systemu:**
  - implementacyjnie niezależna od innych,
  - rozbudowywalna,
  - wymienialna,
  - ponownie używalna.

- **Udostępnia swoją funkcjonalność** przez interfejsy i porty.
- Może pokazywać szczegóły swej implementacji.
- **Modeluje** np. aplikację, bibliotekę, współdziałanie.
  - Sposób jego modelowania, ulepszania i wdrażania określa cykl życia oprogramowania.
- **Artefakt** /artifact/ – klasyfikator implementujący komponent (np. plik *.jar*).
  - jego wdrożenie i aktualizacja powinny być niezależne.

Zawartość komponentu

- **Stereotypy** («*component*» i dodatkowe inne) – modyfikacja komponentu.
- **Nazwa** – nazwa komponentu.
- Przedziały:
  - **atrybuty** /propeties/ – prywatne atrybuty (własności) komponentu;
  - **interfejsy zapewniane** /provided interfaces/ przez komponent;
  - **interfejsy wymagane** /required interfaces/ przez komponent;
  - **realizacje** /realizations/ – składowe klasyfikatory komponentu realizujące interfejsy;
  - **artefakty** /artifacts/ – klasyfikatory implementujące komponent;
  - **wewnętrzna struktura** /internal structure/ – składowe komponenty (i relacje między nimi) realizujące funkcjonalność komponentu → diagram komponentów;
  - **elementy spakowane** /packaged elements/ – składowe klasyfikatory (klasy, obiekty i ich relacje) realizujące funkcjonalność komponentu → diagram struktur złożonych.

Dodatkowe stereotypy komponentu

- Określają, co lub jak komponent modeluje, np.:
  - **«*document*»** – dokument dowolnego typu;
  - **«*file*»** – plik z danymi lub kodem źródłowym;
  - **«*library*»** – biblioteka programu;
  - **«*executable*»** – wykonywalny program;
  - **«*process*»** – proces (działa etapowo - stanowo);
  - **«*service*»** – usługa (działa bezetapowo - bezstanowo);
  - **«*subsystem*»** – podsystem dużej skali (złożony z wielu komponentów);
  - **«*entity*»** – encja (trwałe dane), zwykle bez funkcjonalności;
  - **«*table*»** – tabela bazy danych;
  - **«*specification*»** – specyfikacja komponentu (są interfejsy, NIE ma implementacji);
  - **«*realization*»** – konkretna realizacja komponentu «*specification*».

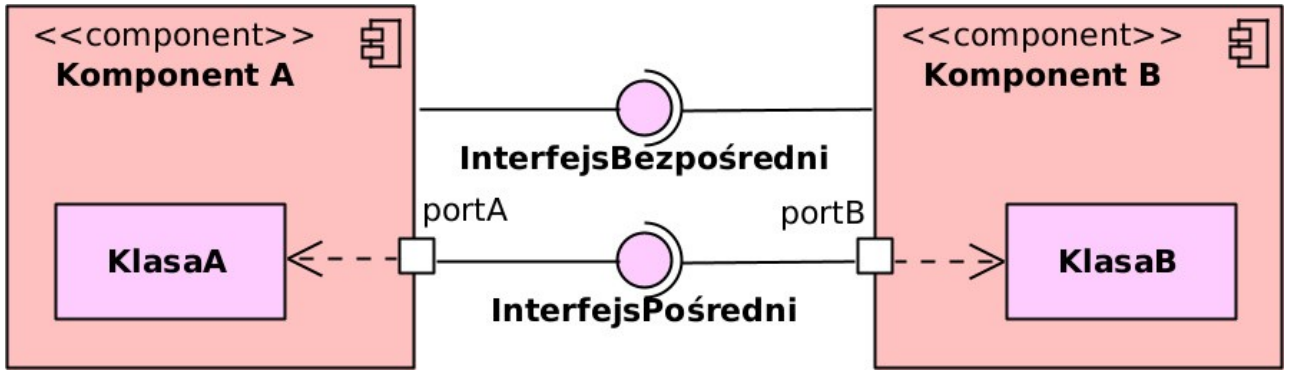
Relacje między komponentami

- Komponenty mogą być powiązane ze sobą:
  - **pośrednio** – łączy je interfejs (np. z użyciem portów),
  - **bezpośrednio** – łączy je inna relacja (np. z użyciem portów),
  - **hierarchicznie** – jeden jest częścią drugiego.

- Komponent używający porty to opakowany klasyfikator.

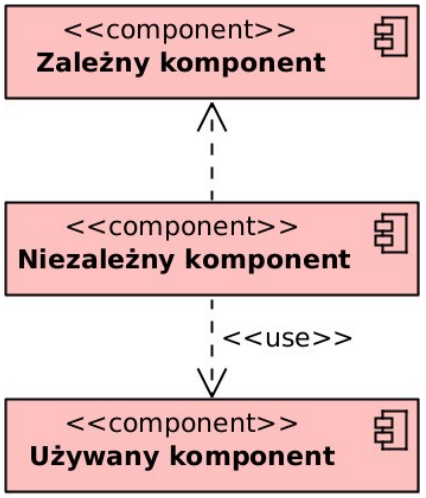
Interfejsy

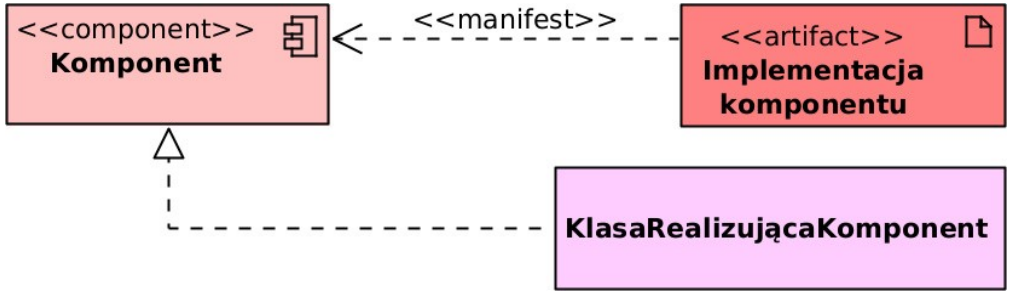
- Dostęp komponentu do innego komponentu
  - za pośrednictwem interfejsu:
    - w postaci „lizaka” (gdy zapewniany) lub „łapki” (gdy wymagany) lub w postaci klasy ze stereotypem: «*interface*»,
    - interfejs może być zaczepiony w porcie.
- **Relacja zależności** – łączy spakowane elementy (klasyfikatory), które zapewniają interfejsy lub wymagają interfejsów komponentu, z portami tych interfejsów:
  - wskazuje na klasyfikator.
- Diagramy klas, modelujące komponent, powinny zawierać jego interfejsy i spakowane klasy.

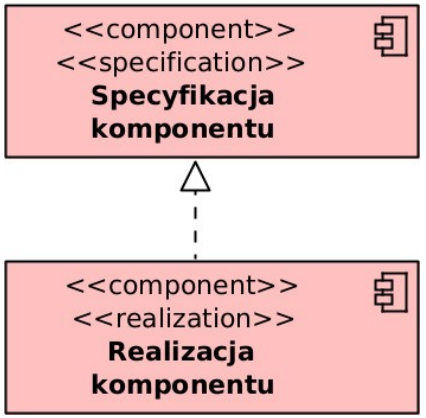


Relacje zależności i realizacji

- Dostęp komponentu do innego komponentu
  - BEZ pośrednictwa interfejsu;
  - przez **relację zależności**:
    - gdy komponent (lub artefakt) jest zależny od innego komponentu,
    - rodzaj zależności może określić jej stereotyp.
  - przez **relację realizacji**:
    - gdy komponent (lub klasa) jest realizacją innego komponentu.

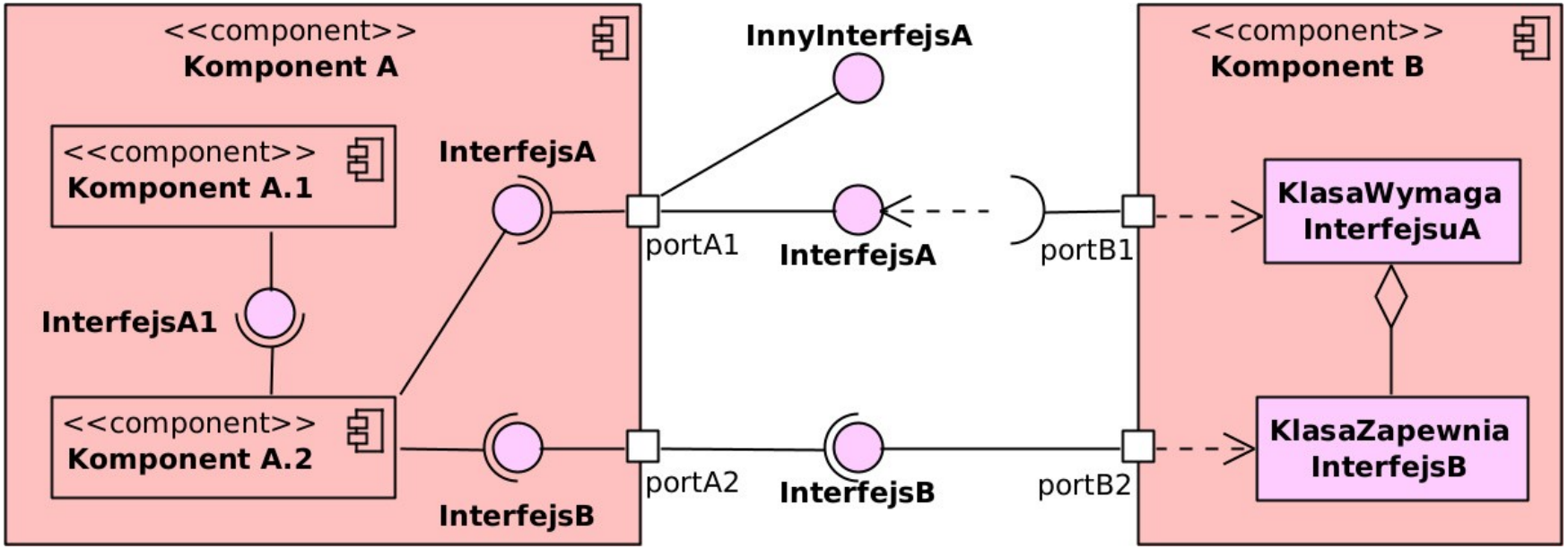






Przykład relacji między komponentami (i spakowanymi elementami)

- **Komponent A** pokazuje swoją wewnętrzną strukturę:
  - **Komponent A.1** współpracuje z **Komponentem A.2** przez **InterfejsA1**.
- **Komponent B** pokazuje swoje spakowane elementy:
  - klasy uczestniczące we współpracy z **Komponentem A** i ich związek.
- **portA1** udostępnia 2 interfejsy.



2

Diagram wdrożenia

Diagram wdrożenia /deployment diagram/

- Modeluje fizyczną i logiczną strukturę systemu:
  - powiązanie oprogramowania ze sprzętem (na którym jest wdrażane), innym systemem (z którym współpracuje) i artefaktami;
  - jako układ węzłów powiązanych strukturalne i komunikacyjne:
    - fizyczne urządzenia,
    - środowiska wykonawcze,
    - artefakty (elementy fizyczne),
    - komponenty (elementy logiczne).
- W fazie specyfikacji wymagań: umiejscawia logiczne komponenty oprogramowania w infrastrukturze klienta.
- W fazie wdrożenia: dokumentuje sposób instalacji oprogramowania.
- Głównie dla systemów wbudowanych i rozproszonych typu klient-serwer.
- Klasyfikator ***«deployment spec»*** – plik ze specyfikacją wdrożenia.

Diagram wdrożenia

Węzeł /node/

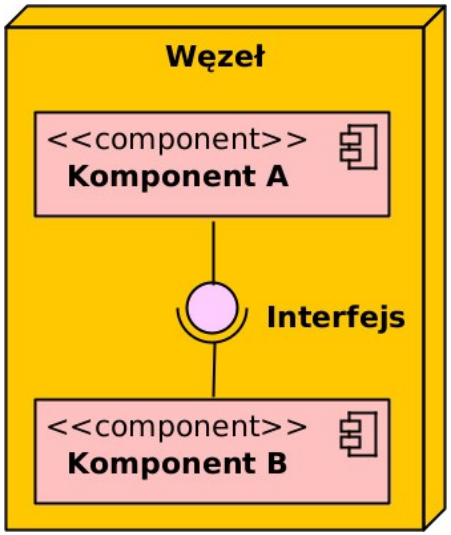
- Fizyczny element systemu lub jego otoczenia: sprzęt lub oprogramowanie.
- Węzeł może mieć stereotyp – określa jego rodzaj:
  - **«*device*»** – fizyczne urządzenie (sprzęt),
  - **«*executionEnvironment*»** – środowisko uruchomieniowe lub wykonawcze komponentów systemu.
  - Można definiować własne stereotypy.

- Węzeł może być zagnieżdżony.
- **Elementy związane z węzłem** (węzły, komponenty, artefakty):
  - są w nim umieszczone,
  - są z nim w relacji bezpośredniej (zależność i asocjacja),
  - są z nim w relacji pośredniej (np. interfejs łączący komponenty).
- Relacja asocjacji między węzłami to ścieżka komunikacji między nimi.

Komponent /component/

- Klasyfikator ze stereotypem **«*component*»** lub ikoną komponentu.
- **Logiczny element:**
  - część oprogramowania systemu,
  - implementacyjnie niezależny od innych.

- **Relacje między komponentami:**
  - zależność, asocjacja, agregacji, kompozycji, uogólnienie;
  - powiązanie przez interfejs (w postaci „lizaka” i „łapki”).
- Komponent może być zagnieżdżony.
- **Wdrożenie komponentu** przez węzeł:
  - umieszczenie go w węźle.



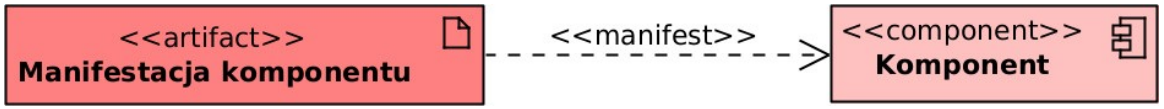
Artefakt /artifact/

- Klasyfikator ze stereotypem **«*artifact*»** lub ikoną dokumentu.
- **Fizyczny element:**
  - używany lub wytwarzany przez wdrażany system;
  - manifestacja komponentu systemu (jego fizyczna reprezentacja, implementacja);
  - np. informacja, plik źródłowy, plik biblioteki, plik wykonywalny, plik danych, tablica bazy danych.
- **Relacje między artefaktami**:
  - zależność, asocjacja, agregacji, kompozycji, uogólnienie.
- Artefakt może być zagnieżdżony.
- Artefakt może mieć atrybuty.

Artefakt

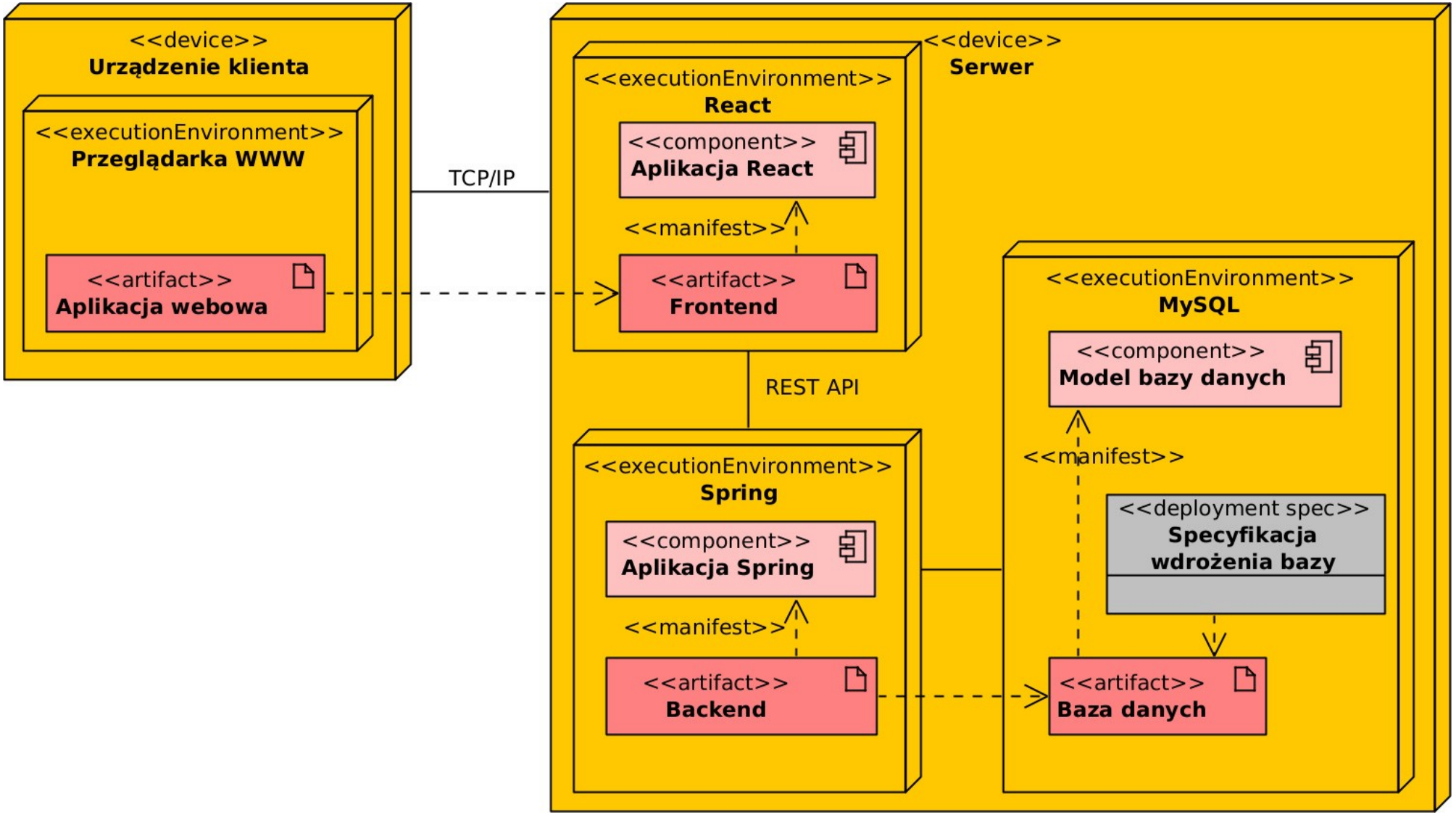
- **Używanie lub wytwarzanie artefaktu** przez węzeł:
  - umieszczenie go w węźle,
  - połączenie go z węzłem relacją zależności **«*deploy*»**.

- **Manifestacja komponentu przez artefakt**:
  - połączenie go z komponentem relacją zależności **«*manifest*»**.



Przykład diagramu wdrożenia

- Koncepcja wdrożenia webowej aplikacji bazodanowej.



Temat następnej prezentacji

Wzorce projektowe