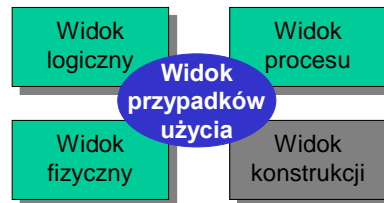


Projektowanie oprogramowania

Diagram pakietów i diagram komponentów, diagramy wdrożenia

1

Widok konstrukcji



- Opisuje sposób, w jaki części systemu są zorganizowane w moduły oraz komponenty.
- Zawiera zazwyczaj diagramy:
 - Pakietów
 - Komponentów.

2

Diagram komponentów

3

Czym jest komponent?

- Jest hermetyzowaną, możliwą do powtórnego wykorzystania oraz zastępowalną częścią oprogramowania.
- Komponenty powinno dać się traktować jako klocki, z których można tworzyć wynikowy program.
- Ważne jest luźne powiązanie pomiędzy komponentami, tak aby zmiany w jednym z nich nie wpływały na resztę systemu. W tym celu dostęp do komponentów realizowany jest z użyciem **interfejsów**.

4

Komponent – jednostka implementacji, dobrze wyizolowana z kontekstu, z dobrze zdefiniowanym interfejsem, nadająca się do **wielokrotnego** wykorzystania.

UML 1.*



UML 2.0




5

Kategorie modelowania (1)

Kategoria modelowania	Notacja
<i>komponent</i> (ang. component)	
<i>interfejs udostępniany</i> (ang. provided interface)	
<i>interfejs wymagany</i> (ang. required interface)	
<i>port</i> (ang. port)	
<i>Port złożony</i> (ang. complex port)	

6

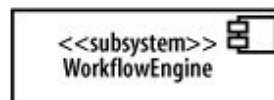
Kategorie modelowania (2)

Kategoria modelowania	Notacja
zależność (ang. dependency)	
realizacja (ang. realization)	
Konektor delegowany (ang. connector with «delegate» stereotype)	
Konektor składany (ang. ball & socket)	

7

Podsystem

- W sytuacji, w której komponent jest w rzeczywistości podsystemem bardzo dużego systemu, stereotyp <<component>> zamienia się na <<podsystem>>.
 - Stereotyp ten powinien być stosowany dla większej części całego systemu.



8

Interfejsy komponentu

- Komponenty do komunikacji pomiędzy sobą używają interfejsów:
 - **Interfejs udostępniany** przez komponent jest interfejsem przez niego realizowanym
 - **Opisuje dostarczane usługi przez komponent**
 - **Interfejs wymagany** jest interfejsem, którego komponent wymaga do poprawnego działania (który realizuje inny komponent)
 - **Deklaruje usługi, które będą używane przez komponent.**

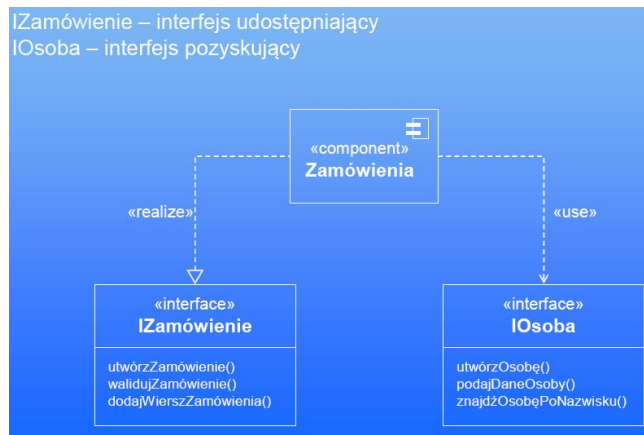
9

Interfejsy komponenty – notacja kuli i gniazda



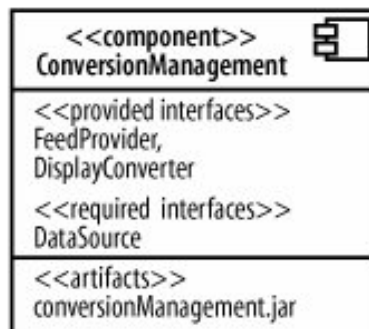
10

Interfejsy komponenty – notacja stereotypu



11

Interfejsy komponentu – notacja tekstowa

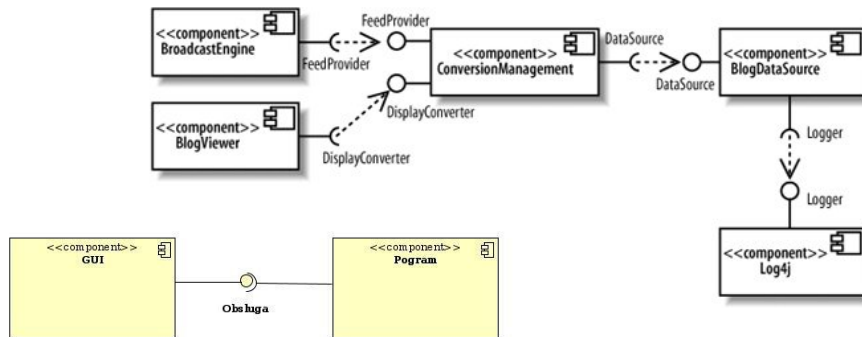


Cześć <<artifacts>> pozwala na wymienienie artefaktów lub fizycznych plików określających komponent.

12

Prezentacja współdziałania komponentów

- Jeżeli komponent wymaga jakiegoś interfejsu, wtedy musi użyć innej klasy lub komponentu, który go dostarcza.



13

Specyfikacja komponentu

- W postaci **czarnej skrzynki**:
 - z tzw. perspektywy zewnętrznej, bez pokazywania zawartości komponentu; specyfikowane są wyłącznie interfejs i/lub operacje oraz atrybuty komponentu.
- W postaci **białej skrzynki**:
 - z tzw. perspektywy wewnętrznej; wprowadzono tu dodatkową sekcję specyfikującą klasy (ew. komponenty), o które oparto realizacją komponentu. Inne dodatkowe sekcje mogą być wykorzystane np. dla specyfikowania konektorów czy artefaktów skojarzonych z komponentem.

14

Specyfikacja komponentu



15

Alternatywna reprezentacja zawartości komponentu



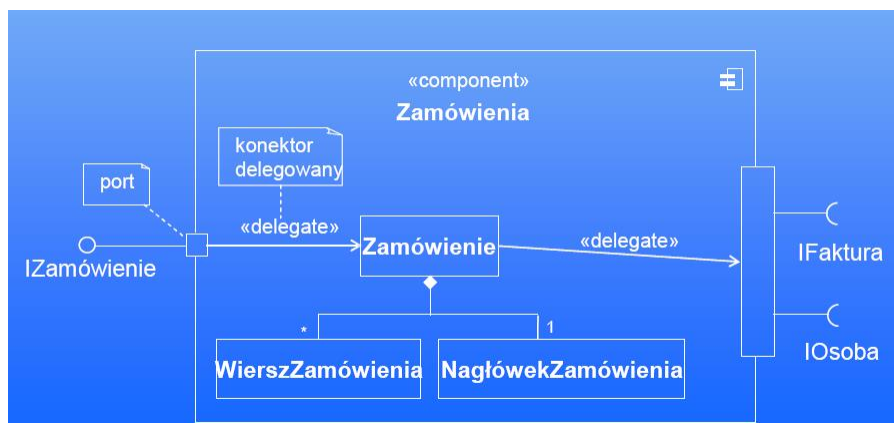
16

Porty i konektory

- Port:
 - Wyróżniony element związany z interfejsem, przez który komponent komunikuje się z otoczeniem.
- Port złożony:
 - Port skojarzony z więcej niż jednym interfejsem
- Konektor:
 - Wykorzystywany do łączenia elementów diagramu komponentów.

17

Porty i konektory

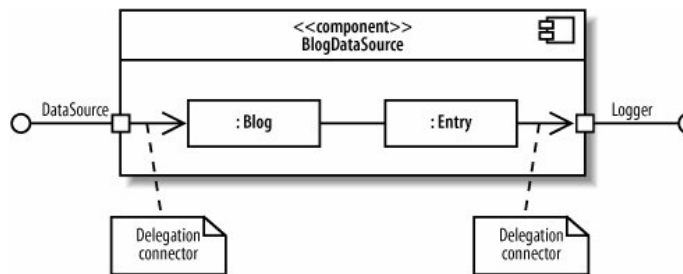


18

Porty i konektory

Konektory:

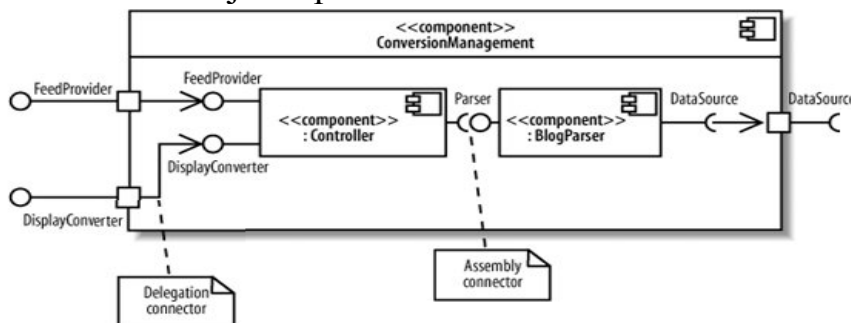
- Delegujące – używane aby przedstawić fakt, że części wewnętrzne realizują lub używają interfejsów komponentu



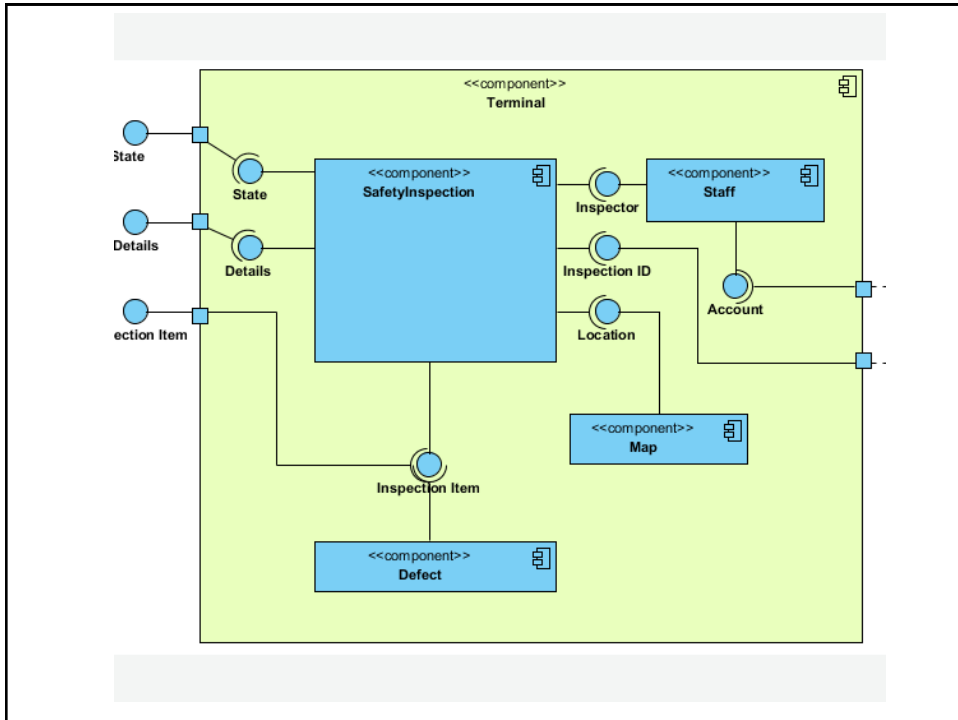
19

Konektory montażowe

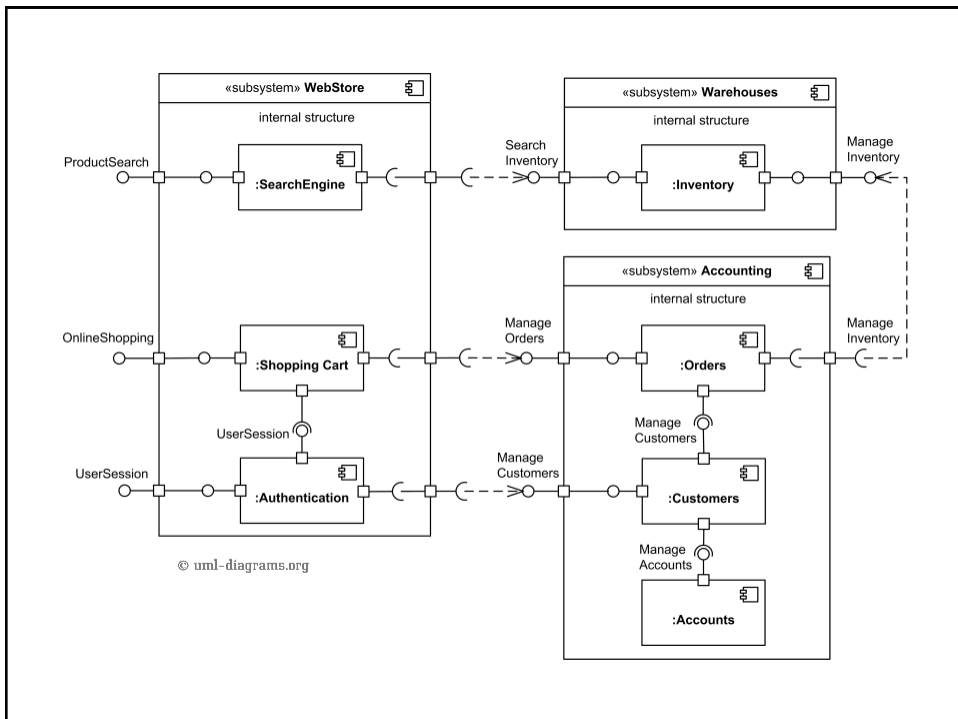
- Wskazują, które komponenty wymagają interfejsu udostępnianego przez inny komponent.
 - Stosowany podczas prezentowania struktury złożonej komponentów.



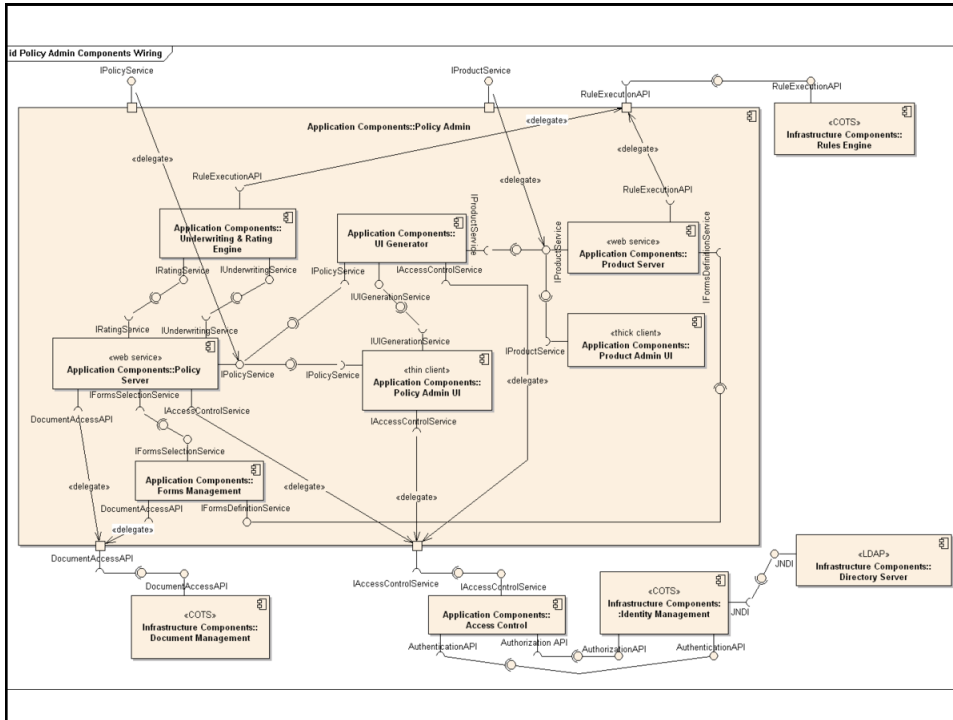
20



21



22

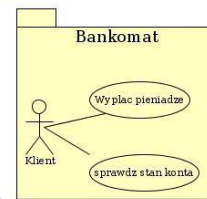


23

Diagramy pakietów

24

Pakiety



- Pakiety - logicznie powiązane grupy klas.
- Pakiety mogą zostać uszkodzone, gdy zostanie zmieniony inny pakiet, od którego bieżący zależy.
- Diagramy pakietów – wykorzystywane do przedstawiania zależności pomiędzy pakietami, co jest niezbędne do zapewnienia stabilności systemu.
- Pakiety mogą służyć do porządkowania także innych elementów języka UML (np. przypadków użycia).

25

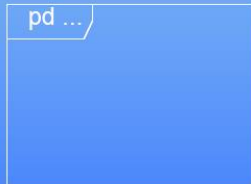
Diagramy pakietów

- Pakiety mogą być zagnieżdżone.
- Diagramy pakietów są istotne przede wszystkim dla dużych projektów, składających się z wielu jednostek funkcjonalnych (ze złożonymi relacjami pomiędzy tymi jednostkami) oraz grupę współpracujących osób.

26

Reprezentowanie diagramu pakietów

<nagłówek-diagramu> = (<wyróżnik_diagramu>) + <nazwa-diagramu> + ({<parametr>})



pd – wyróżnik diagramu komponentów
(package diagram)

27

Kategorie modelowania

Kategoria modelowania	Notacja
<i>pakiet</i> (ang. package)	
<i>zależność</i> (ang. dependency)	

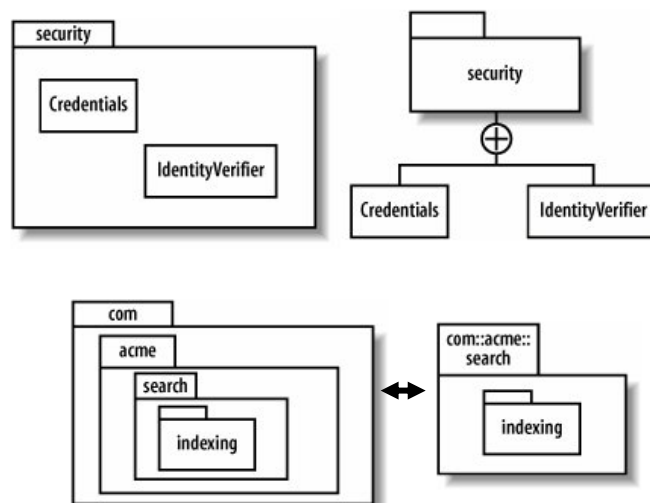
28

Kategorie modelowania

- Nazwę pakietu umieszcza się albo wewnątrz większego prostokąta (gdy nie pokazuje się zawartości pakietu) albo wewnątrz mniejszego prostokąta – zakładki.
- **Symbol pakietu można używać we wszystkich rodzajach diagramów.** Można pokazywać zależności, asocjacje czy inne relacje zachodzące między wewnętrznymi elementami pakietów, czy też pakietami. Zazwyczaj uwidaczniany jest jedynie fakt istnienia zależności.

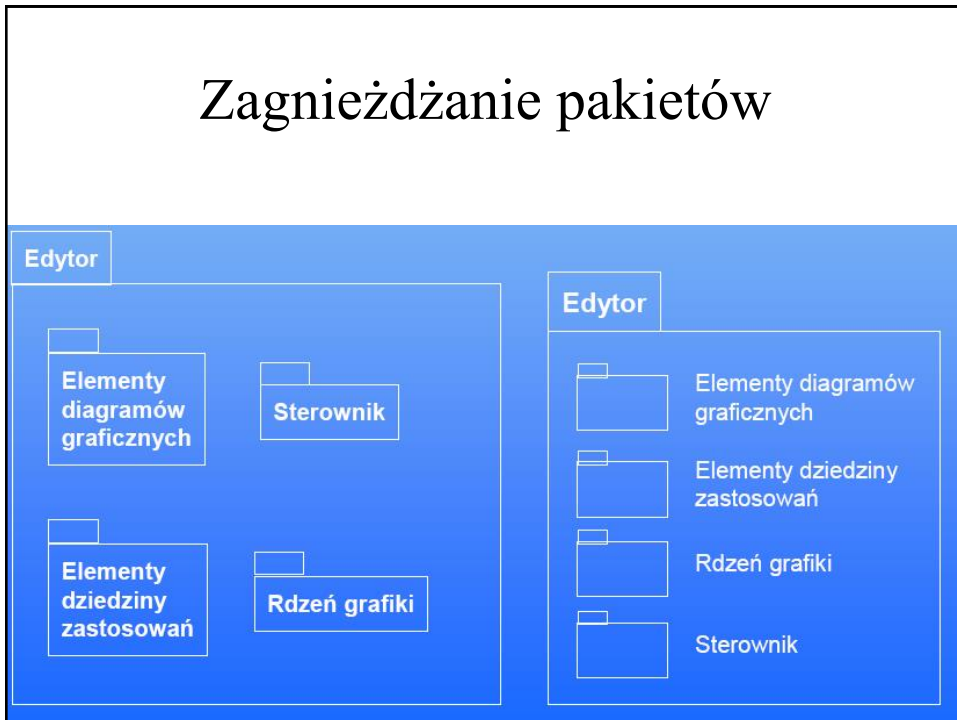
29

Zawartość pakietu



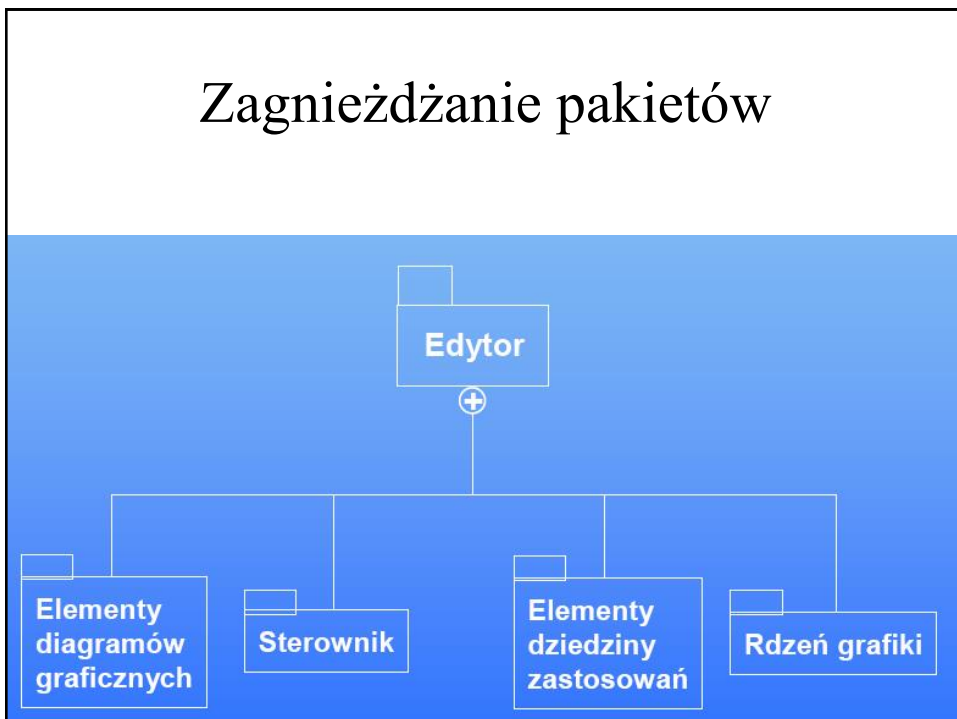
30

Zagnieżdżanie pakietów



31

Zagnieżdżanie pakietów

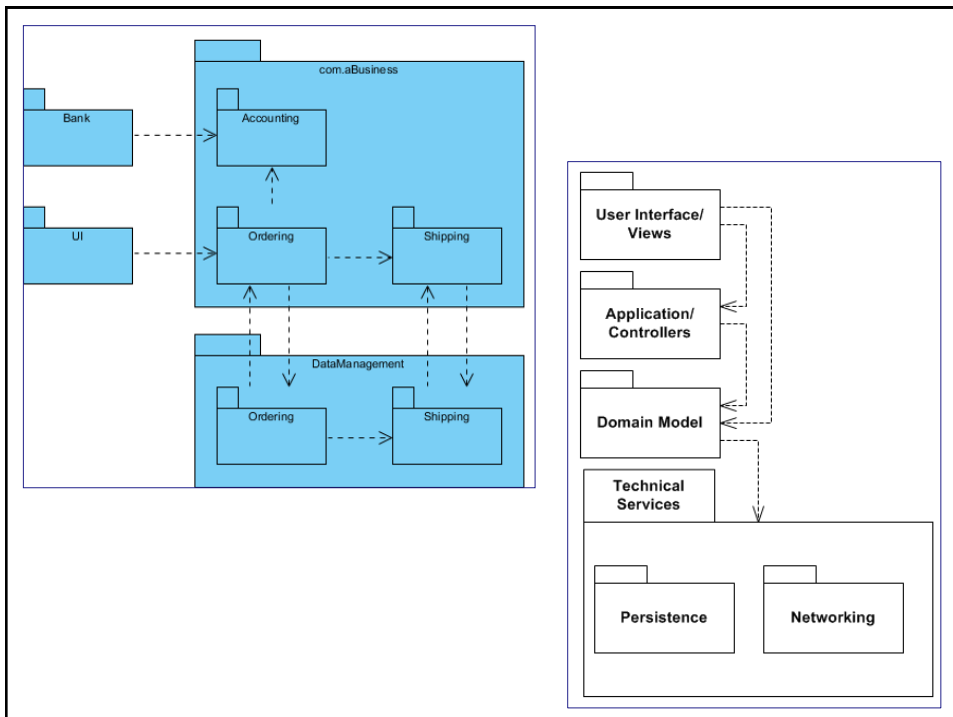


32

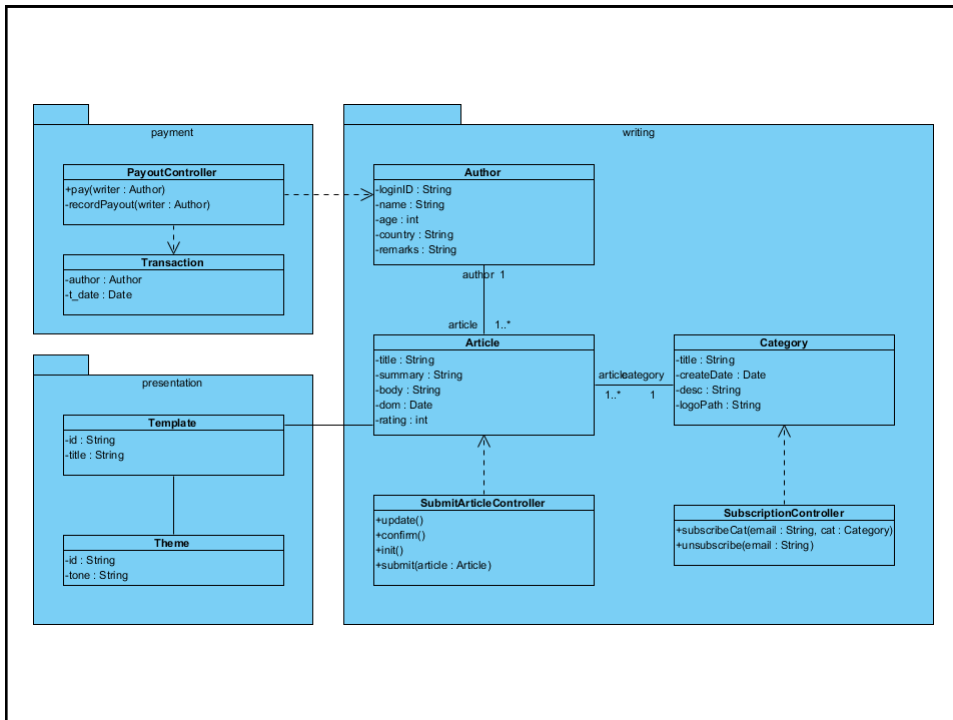
Zależności

- **Zależności występujące pomiędzy pakietami wynikają z relacji między ich elementami składowymi.**
- Zależności są przedstawiane na diagramach pakietów w postaci strzałek z przerywaną linią i mogą być opatrywane stereotypami.
- **Relacje między elementami, opisywane pośrednio przez zależności, mogą być różnego rodzaju, ale tego typu informacja zazwyczaj nie jest przenoszona przez diagramy pakietów.**
 - Dzięki specyfikacji zależności uwidaczniany jest jedynie fakt występowania relacji, a nie jej rodzaj, np. fakt istnienia asocjacji między dwiema klasami umieszczonymi w dwóch różnych pakietach.

33

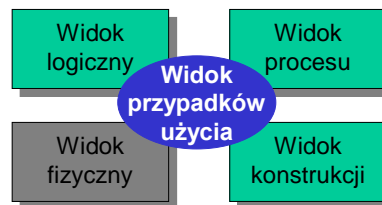


34



35

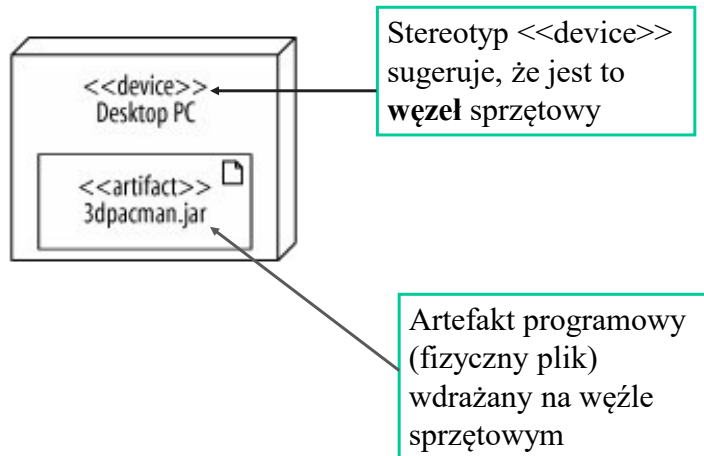
Widok fizyczny



- Wyjaśnia w jaki sposób projekt systemu opisany w trzech poprzednich widokach jest powoływany do życia w postaci zestawu rzeczywistych obiektów.
- Rzeczywiste wdrożenie systemu.
- Zawiera zazwyczaj **diagramy wdrożenia**.

36

Wdrażanie prostego systemu



37

Jakie informacje diagram wdrożeniowy powinien zawierać?

- Istotne dla odbiorcy, czyli:
 - Informacje o sprzęcie,
 - Oprogramowanie producenta,
 - System operacyjny,
 - Środowisko uruchomieniowe,
 - Sterowniki urządzeń systemu,
- Jeśli rzeczywiście ważne są dla odbiorcy!**

38

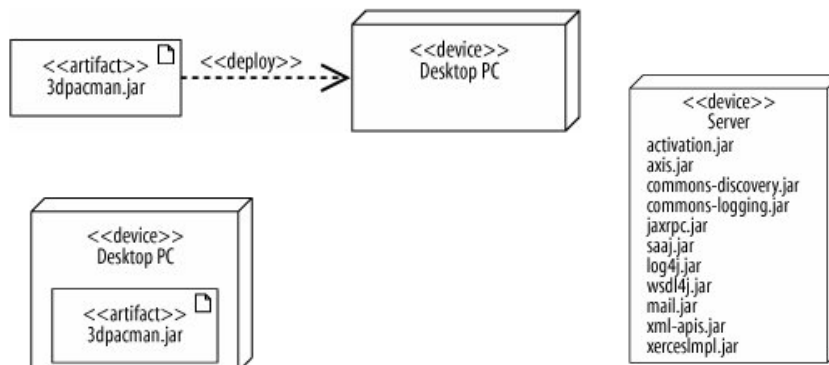
Artefakty

- Artefakty są fizycznymi plikami, które są wykonywane lub używane przez oprogramowanie, np.:
 - Pliki wykonywalne: *.exe lub *.jar,
 - Pliki biblioteczne: *.dll
 - Pliki źródłowe: *.java lub *.cpp
 - Pliki konfiguracyjne: *.xml, *.properties
- Przedstawiane są przy użyciu prostokąta ze stereotypem <<artifacts>> lub specjalną ikoną.

39

Wdrażanie artefaktu w węzle

- Artefakt jest wdrażany w węzle, co oznacza, że rezyduje on (lub jest zainstalowany) w nim.



40

Wiązanie komponentów i pakietów z artefaktami

- W przypadku, gdy artefakt jest fizycznym urzeczywistnieniem komponentu, **manifestuje** go.
- Artefakt może manifestować także pakiety oraz klasy.

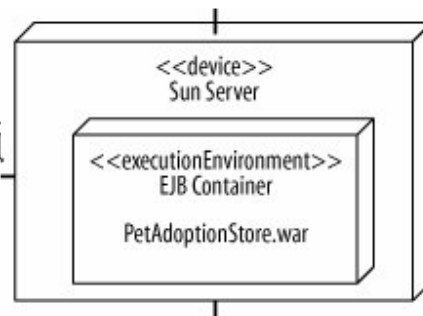


- Związek manifestowania używany również bywa na diagramach komponentów poprzez umieszczenie listy manifestujących komponent artefaktów wewnątrz symbolu komponentu.

41

Węzeł sprzętowy oraz środowisko uruchomieniowe

- Środowisko uruchomieniowe oznaczone poprzez <<executionEnvironment>> jest wykonywane na pewnej maszynie sprzętowej.
- Fakt umieszczenia środowiska na pewnej maszynie sprzętowej:

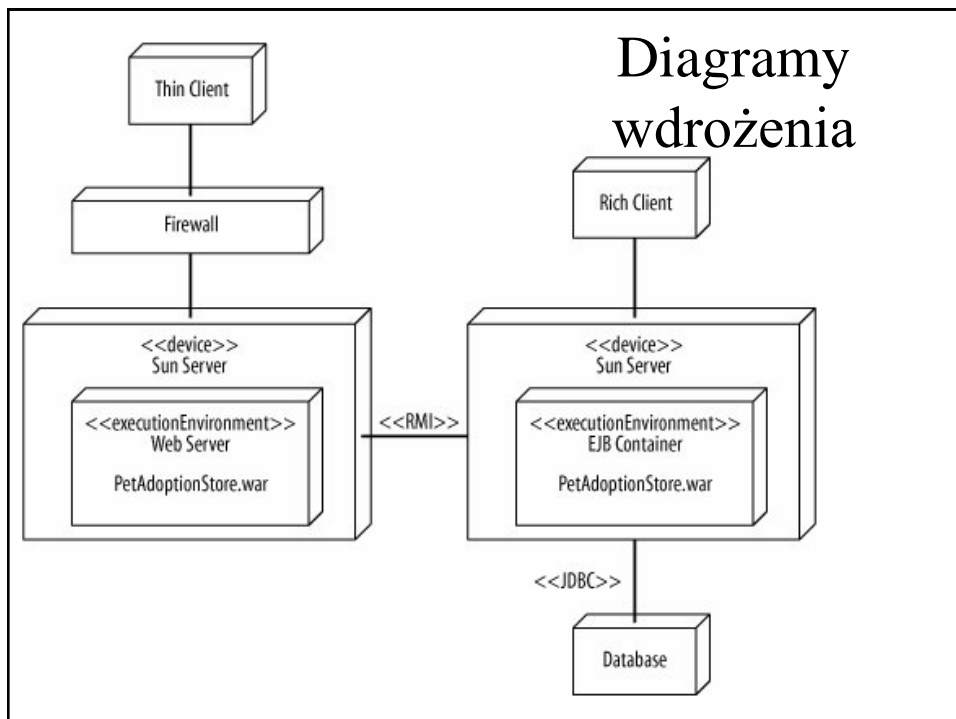


42

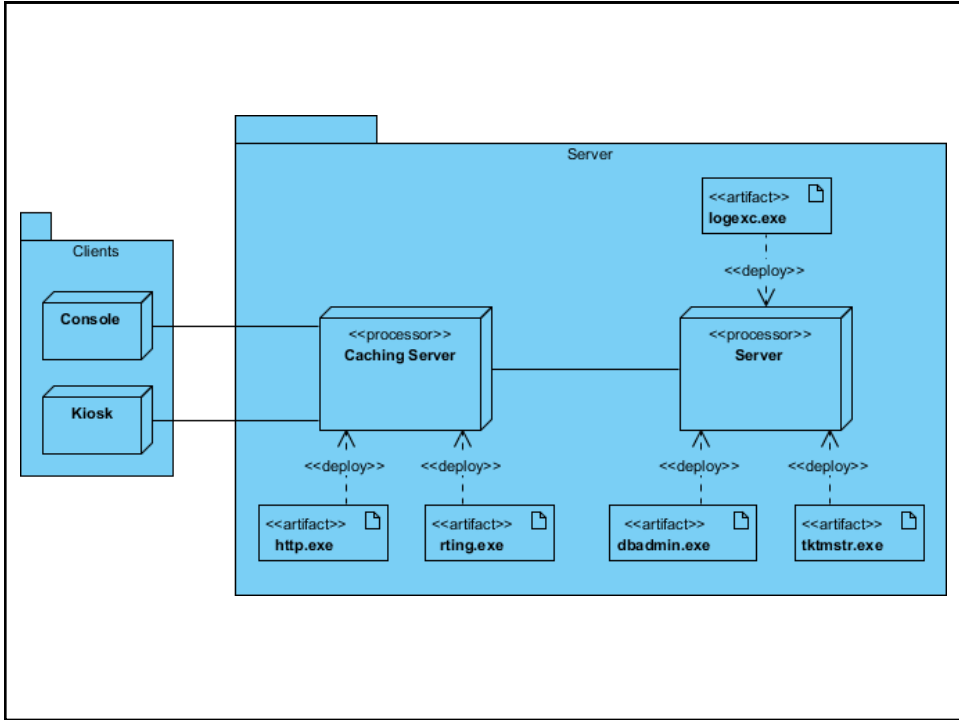
Komunikacja pomiędzy węzłami

- Do przedstawienia komunikacji używane są **ścieżki komunikacji** w postaci ciągłej linii łączącej dwa węzły.
 - Rodzaj komunikacji jest określany poprzez dodanie do ścieżki odpowiedniego stereotypu.
 - Stereotyp komunikacji powinien być na najwyższym poziomie, ponieważ na tym poziomie jest największa ilość informacji o systemie (np. <<RMI>> zamiast <<TCP/IP>>).

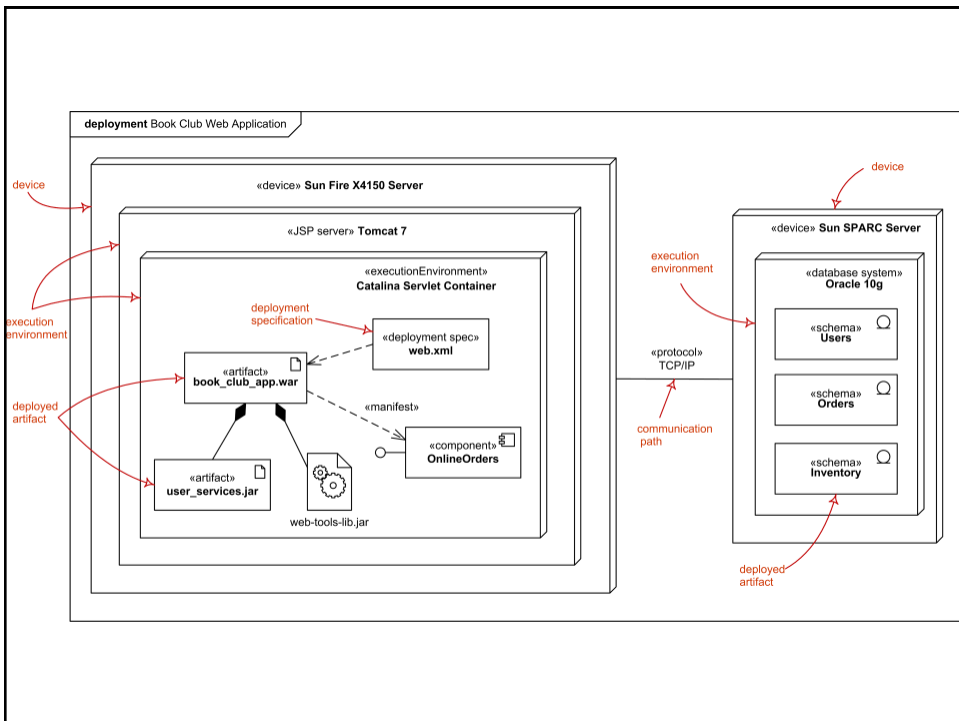
43



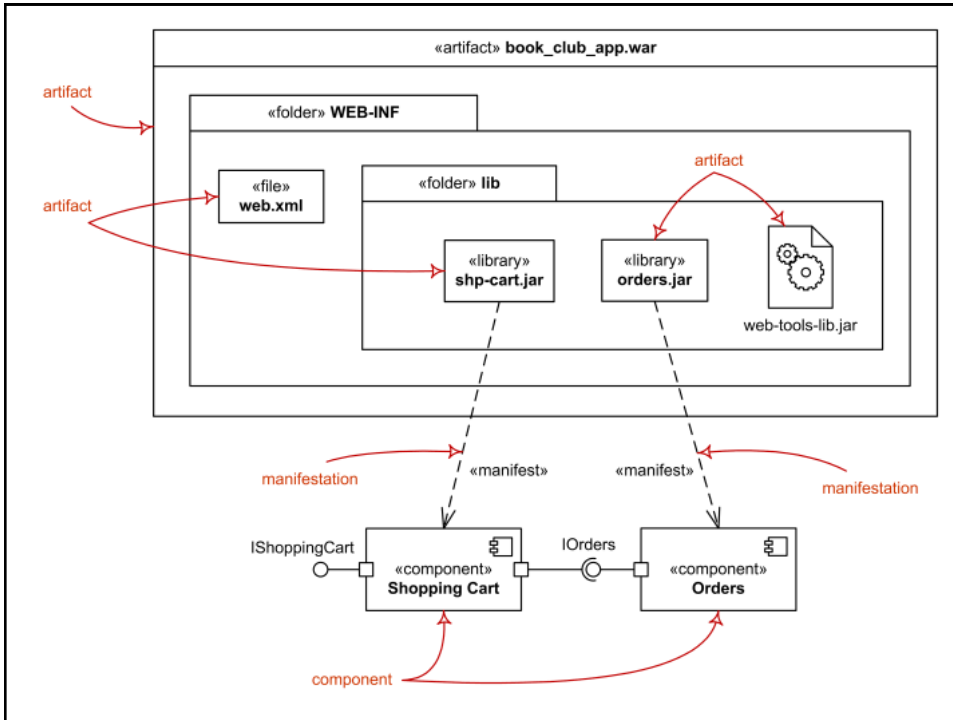
44



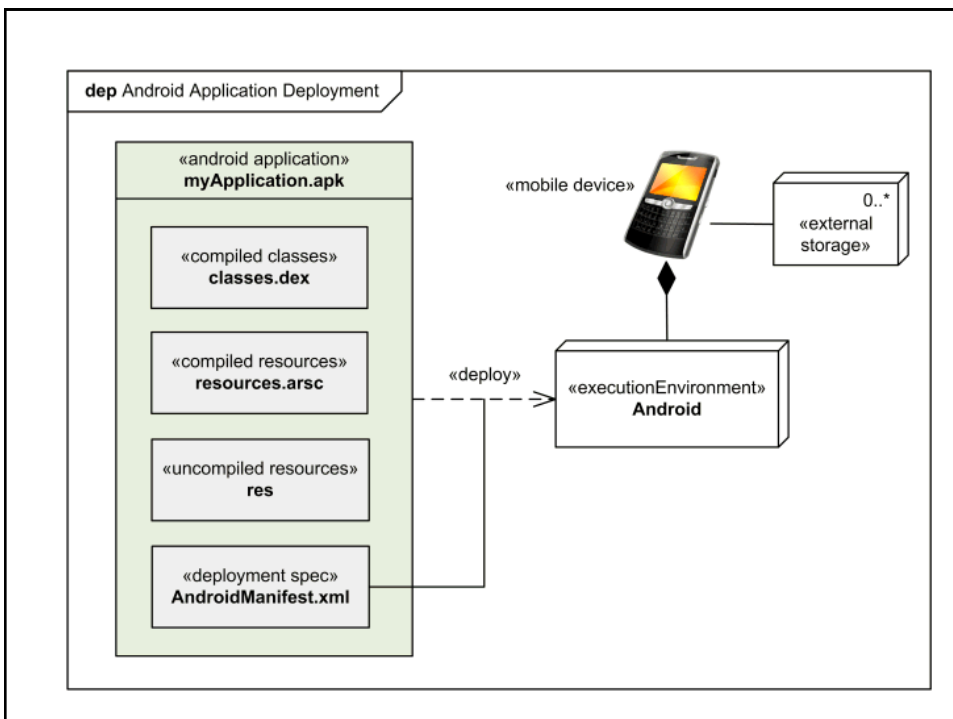
45



46



47



48