Model-Driven Engineering

Radosław Klimek

2015-22

http://home.agh.edu.pl/rklimek

Diagramy implementacji

Diagramy implementacji – rodzaje diagramów implementacji

UML definiuje dwa rodzaje diagramów implementacyjnych:

- **Diagramy komponentów** – ilustrują strukturę kodu
- **Diagramy wdrożeniowe** – przedstawiają konfigurację systemu

Diagram komponentów

Diagram komponentów (ang. *component diagram*) pokazuje podział systemów programowych na mniejsze podsystemy.

Komponent to wymienialny, wykonywalny fragment systemu, z ukrytymi szczegółami implementacyjnymi (np. pliki, biblioteki, gotowe wykonywalne programy itp.).

Diagramy komponentów to zasadniczo diagramy klas, które koncentrują się na komponentach systemu, które często są używane do modelowania statycznego widoku implementacji systemu.

Diagram komponentów – uwagi

Znaczenie diagramu komponentów:

- służy do ilustracji organizacji i zależności pomiędzy komponentami.
- służy do określania szczegółów niezbędnych do budowy systemu.
- prezentuje system na wyższym poziomie abstrakcji niż diagram klas, gdyż każdy z komponentów może być implementacją jednej lub większej liczby klas.
- zapewnia one wysoki poziom widoku komponentów w systemie.
- ilustruje komponenty oprogramowania, które zostaną użyte do zbudowania systemu.
- może być tworzony na podstawie modelu klasowego i napisany od podstaw dla nowego systemu lub może pochodzić z innych projektów i dostawców zewnętrznych.

Diagram komponentów – elementy diagramu

Diagram komponentów jest zawarty w specyfikacji UML od pierwszej wersji (1.1) i zawiera następujące elementy:

- **component**
- **interface**
- **port**
- **internal structure**
- **part**
- **connector**

Diagram komponentów – uwagi

•Komponent udostępnia zestaw interfejsów, może też wymagać pewnych interfejsów do funkcjonowania.

•Komponent to wymienny, wykonywalny fragment systemu o hermetyzowanych szczegółach implementacyjnych.

• Komponenty z natury służą do ponownego wykorzystania poprzez połączenie ich z innymi komponentami, zwykle poprzez ich skonfigurowanie, bez potrzeby rekompilacji.

•Komponenty są wysokopoziomowymi agregacjami mniejszych części oprogramowania i zapewniają podejście „czarnej skrzynki” do budowy oprogramowania

Diagram komponentów – rodzaje komponentów

Komponenty w UML mogą reprezentować

- komponenty logiczne
  - baza danych
  - interfejs użytkownika
- komponenty fizyczne (sprzętowe)
  - obwód
  - mikroczip
  - urządzenie
  - jednostka biznesowa (dostawca, lista płac lub wysyłka itp.)

Komponent - uwagi

Komponent — reprezentuje modułową część systemu, która hermetyzuje jego zawartość, definiuje jego zachowanie w zakresie dostarczonych i wymaganych interfejsów.

Jako taki, komponent służy jako typ, którego zgodność jest definiowana przez dostarczone i wymagane interfejsy (obejmujące zarówno ich semantykę statyczną, jak i dynamiczną).

Jeden komponent można zatem zastąpić innym tylko wtedy, gdy typ obu komponentów jest zgodny.

Diagram komponentów – symbole komponentu

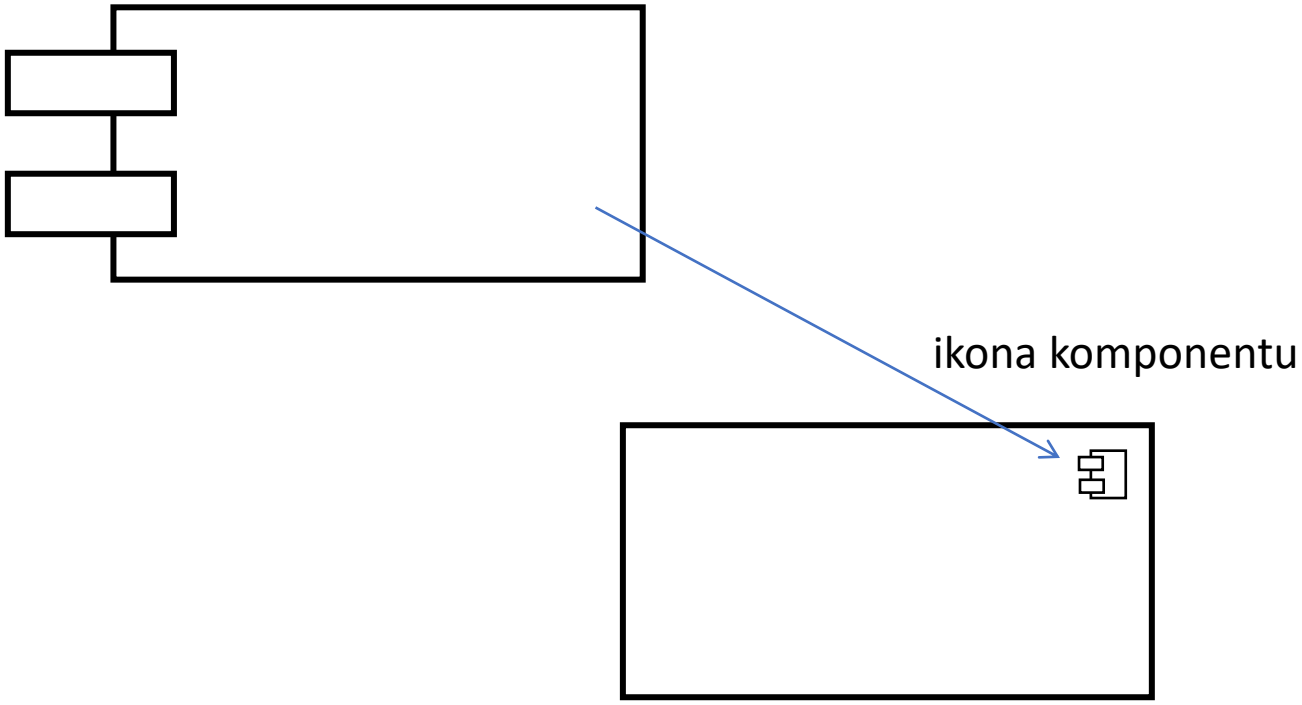


Diagram komponentów – symbole komponentu

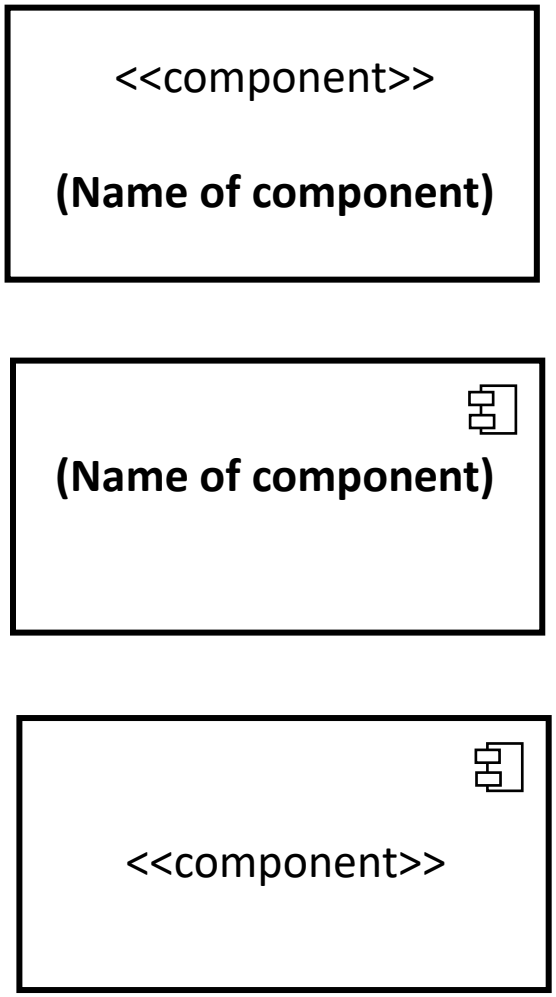


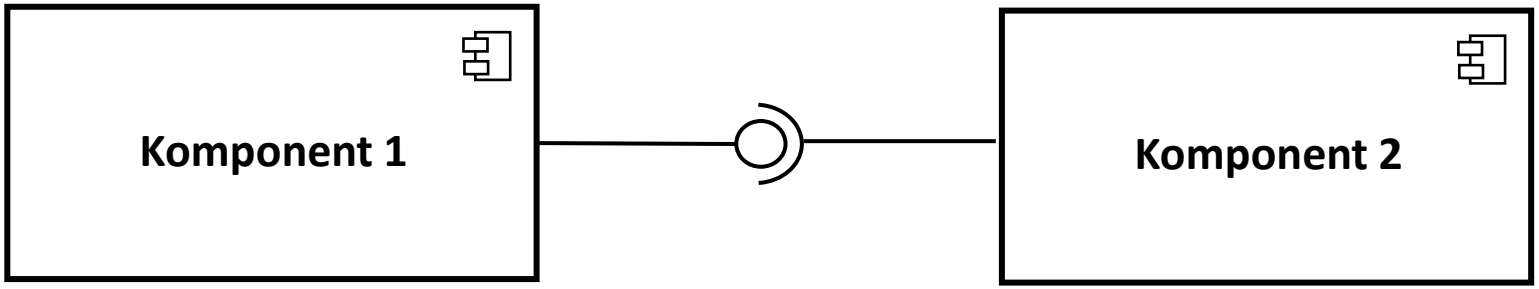
Diagram komponentów – stereotypy komponentu

Komponent może posiadać stereotyp

- *executable* – określa komponent, który można wykonać na węźle;
- *library* – określa dynamiczną lub statyczną bibliotekę obiektów;
- *table* – określa komponent reprezentujący tabelę bazy danych;
- *file* – określa komponent reprezentujący dokument zawierający kod źródłowy lub dane;
- *document* – określa komponent reprezentujący dokument;
- *subsystem* – określa komponent, który jest agregatem dla komponentów dużej skali

Diagram komponentów – interfejsy komponentów

Komponenty wymieniają dane między sobą za pomocą **interfejsów**.



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 – interfejsy komponentów

Dostarczony interfejs (ang. *provided interface*) definiuje „zestaw publicznych atrybutów i operacji, które muszą być dostarczone przez klasy implementujące dany interfejs ”

Wymagany interfejsy (ang. *required interface*) definiuje „zestaw publicznych atrybutów i operacji wymaganych przez klasy zależne od danego interfejsu”

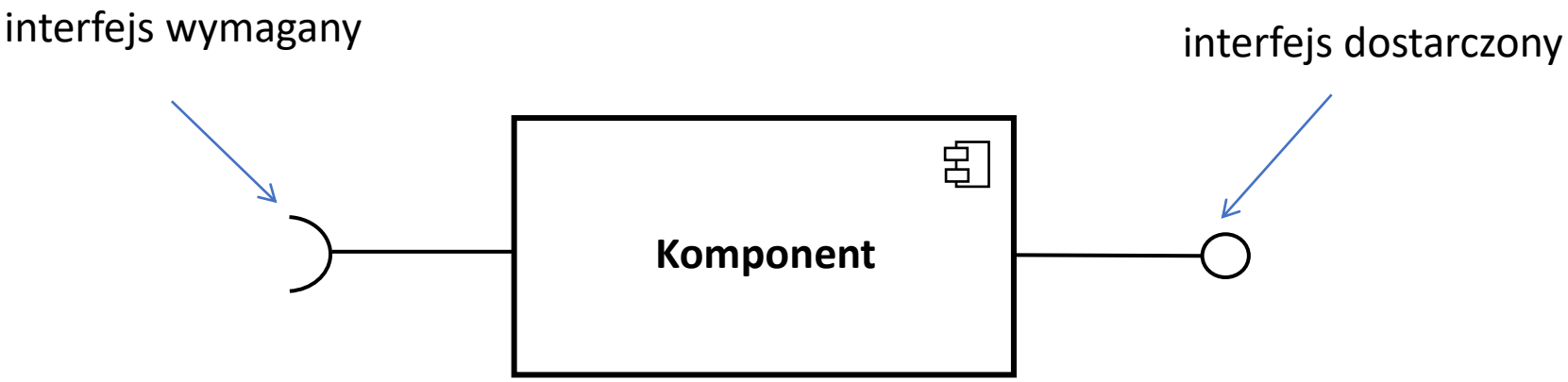


Diagram komponentów – interfejsy komponentów

Interface :

- określa kontrakt składający się z zestawu spójnych publicznych atrybutów i operacji dla klasy.
- każde wystąpienie klasy, które realizuje interfejs, musi spełniać ten kontrakt.
- ponieważ interfejsy są deklaracjami, nie można ich tworzyć.
- specyfikacja interfejsu jest implementowana przez instancję klasy.
- każda klasa może implementować więcej niż jeden interfejs
- każdy interfejs może być implementowany przez wiele różnych klas.

Diagram komponentów – interfejsy komponentów

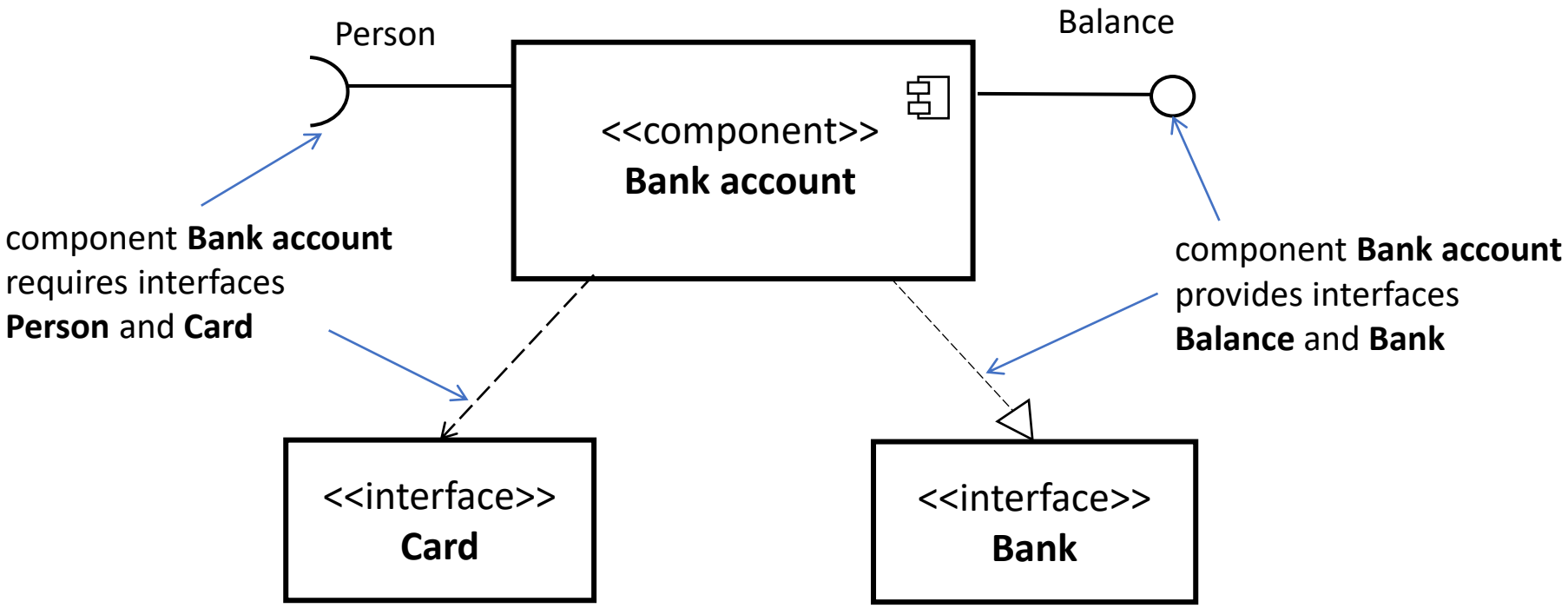
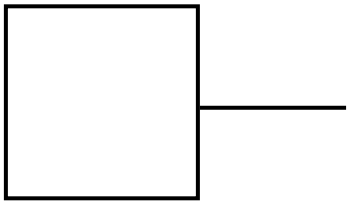


Diagram komponentów – port komponentu

port – określa oddzielny punkt interakcji między komponentem a środowiskiem zewnętrznym.
Jest używany, gdy komponent deleguje interfejsy do klasy wewnętrznej



symbol portu

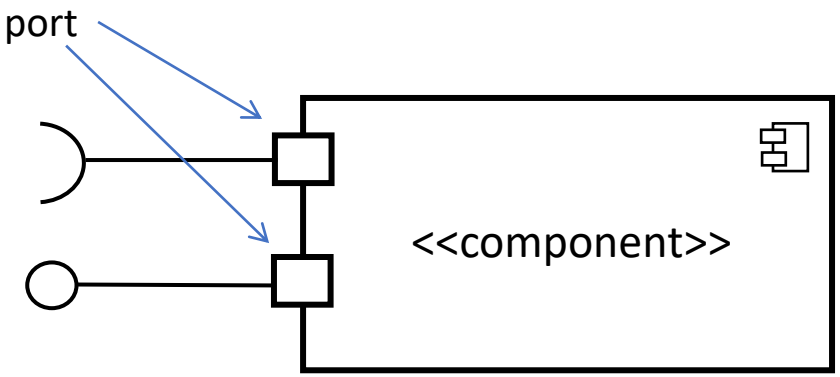
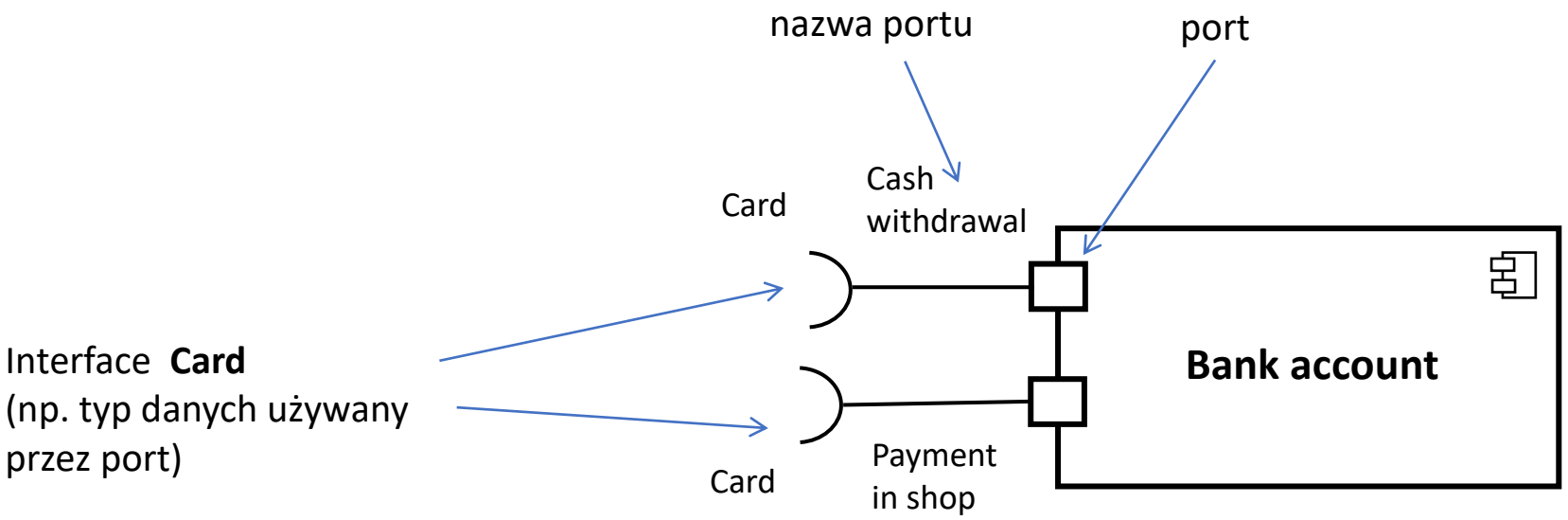


Diagram komponentów – port komponentu

Port — jawne okno do hermetyzowanego komponentu.
•Wszystkie interakcje do i z takiego komponentu przechodzą przez porty.
•Każdy port udostępnia lub wymaga co najmniej jednego określonego interfejsu.
•Może istnieć wiele portów zapewniających lub wymagających tego samego interfejsu.
•Pozwala na większą kontrolę nad implementacją i interakcją z innymi komponentami

Port komponentu



Dwa porty wymagają tego samego interfejsu **Card.**
Port **Cash withdrawal** jest używany, gdy klient banku pobiera gotówkę z bankomatu) za pomocą swojej karty.
Port **Payment in shop** jest używany podczas dokonywania płatności kartą w sklepie.

Diagram komponentów – struktura wewnętrzna komponentu, część

Struktura wewnętrzna (ang. *internal structure*) — używana do określenia struktury złożonego komponentu, tj. zazwyczaj komponenty składają się z mniejszych komponentów, tworząc w ten sposób system.

Część (ang. *part*) — komponent, który buduje wewnętrzną strukturę bardziej złożonego komponentu (podkomponent)

Diagram komponent – struktura wewnętrzna komponentu

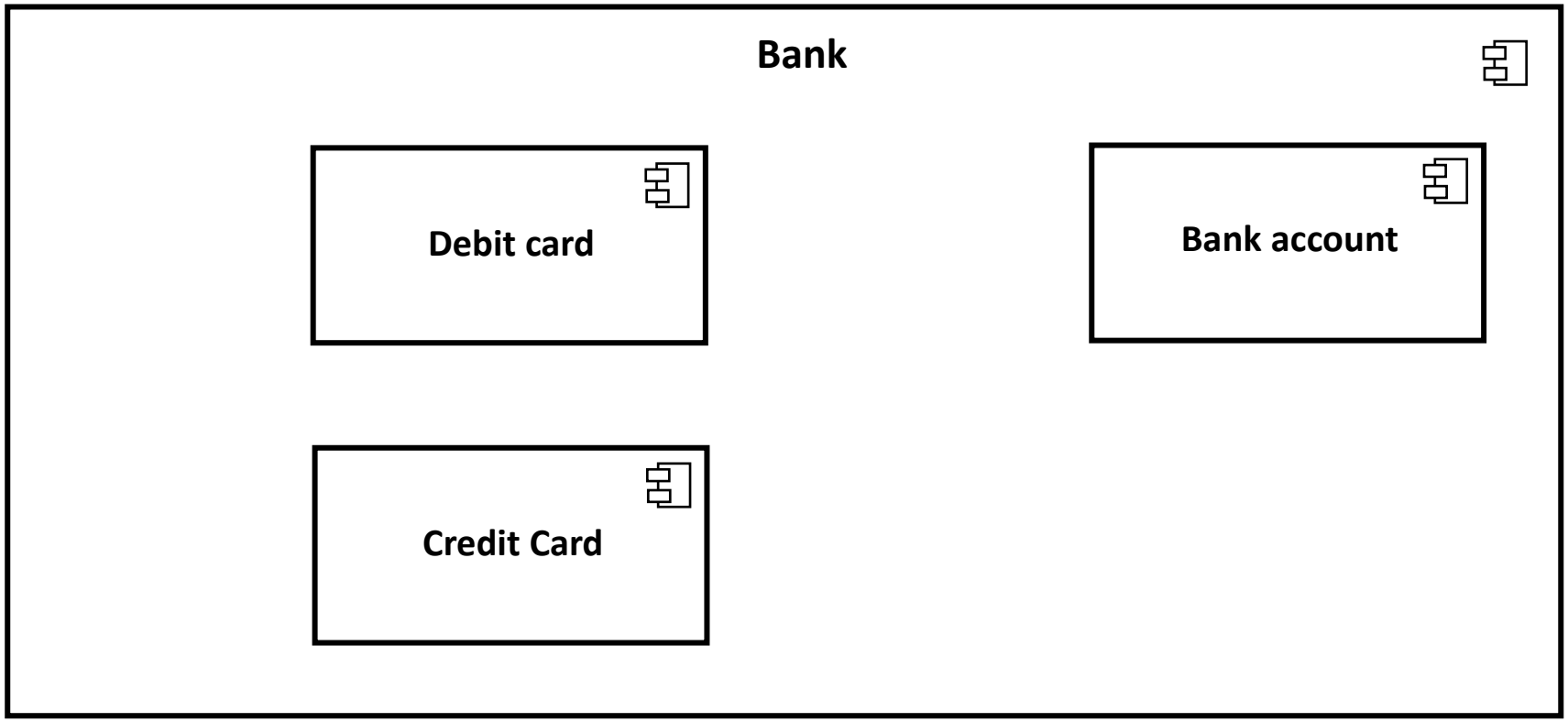


Diagram komponent – złącze

Złącze (ang. *connector*) — relacja pomiędzy portami komponentów. Jeśli jeden port zapewnia interfejs wymagany przez drugi port, można je ze sobą połączyć.

Istnieje kilka typów złączy, które możemy narysować między częściami i komponentami:
złącze bezpośrednie – łączy bezpośrednio ze sobą dwa porty części
złącze przez interfejs – łączy ze sobą dwa porty poprzez interfejsy wymagany z dostarczonym za pomocą notacji piłka–gniazdo (ang. *ball and socket*)
złącze delegacyjne – łączy razem port części i port komponentu dostarczając w ten sposób interfejs dostarczony (tj. dostarcza usługi innym komponentom) lub interfejs wymagany (tj. konsumując usługi innego komponentu).

Diagram komponentów – złącze

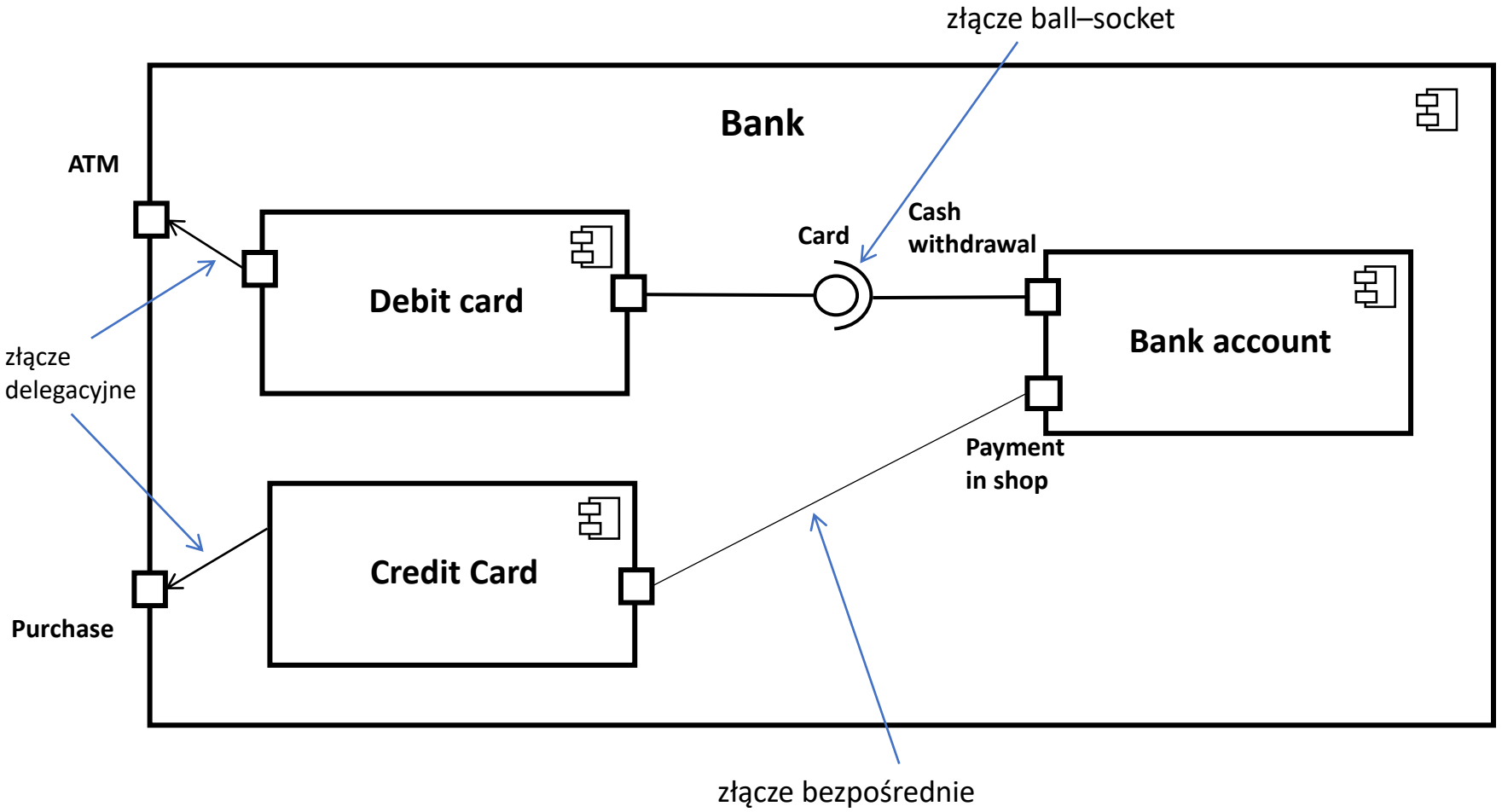


Diagram komponentów – techniki łączenia komponentów za pomocą interfejsów

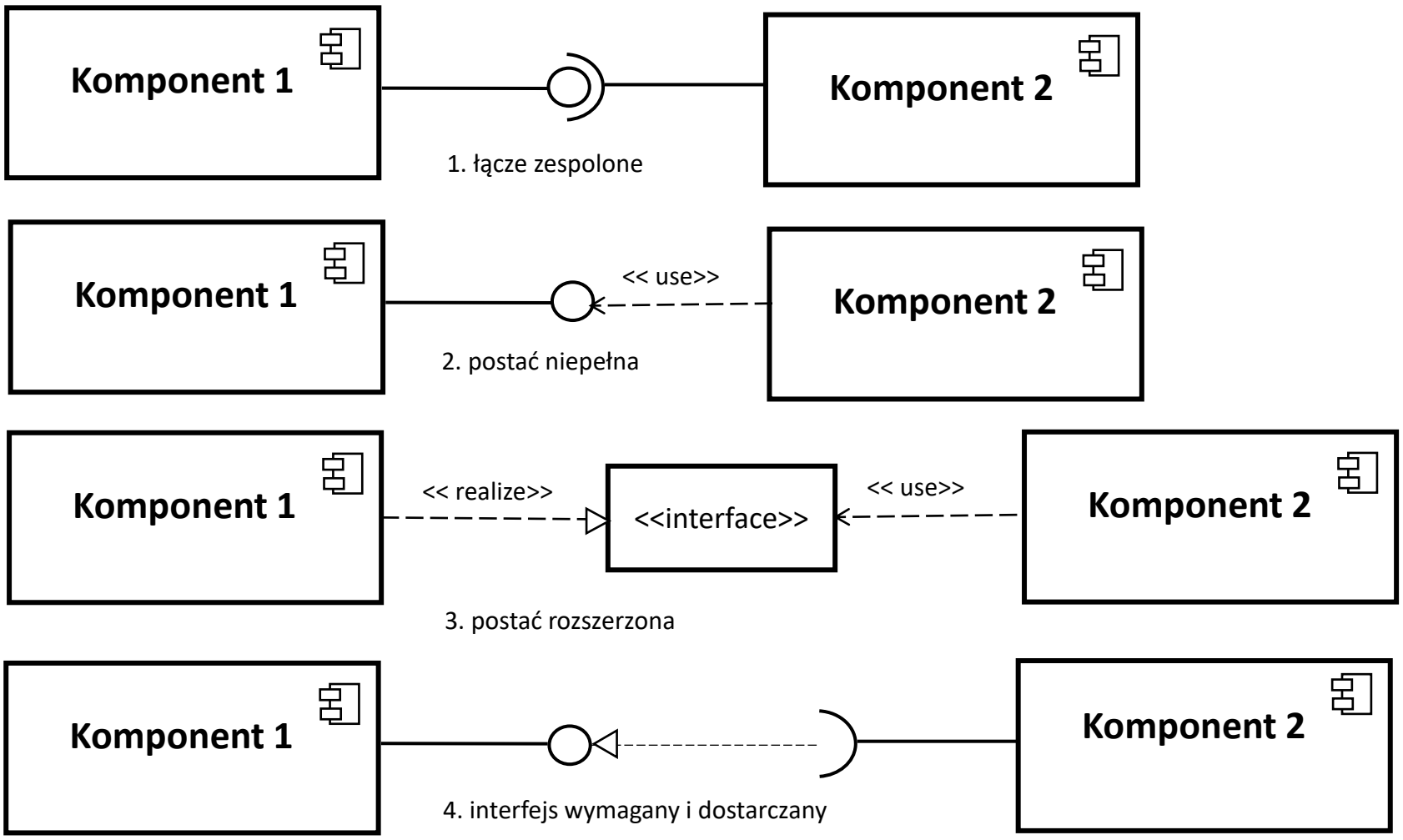
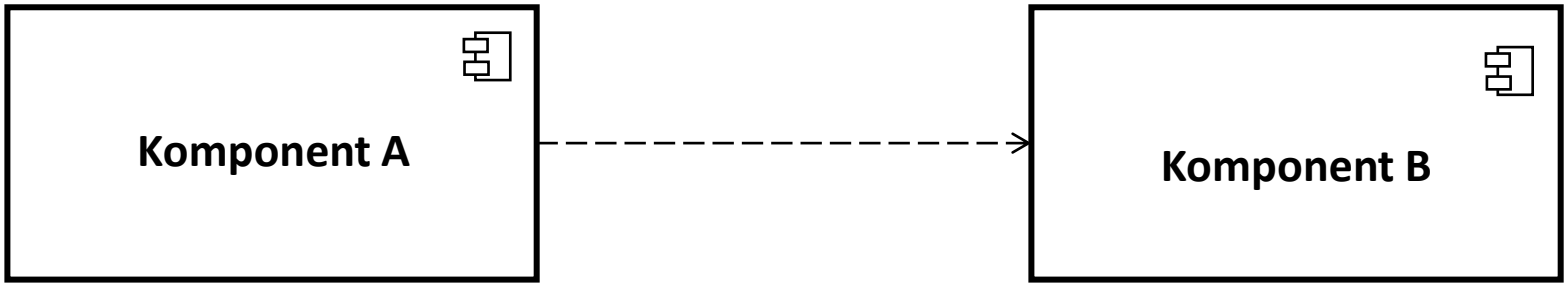


Diagram komponentów – zależność pomiędzy komponentami



Komponenty są między sobą powiązane relacją zależności, ponieważ wymagają do ich realizacji własnej funkcjonalności.

Zależność między A i B oznacza, że komponent A korzysta z komponentu B i zmiana w komponencie B może spowodować konieczność zmiany w A.

Diagram komponentów – Czarna skrzynka

W UML istnieją dwa widoki komponentów: widok czarnoskrzynkowy i widok białoskrzynkowy.

W specyfikowaniu komponentu w charakterze **czarnej skrzynki** (ang. *black box*) przedstawia się komponent wraz z jego interfejsami w sposób ogólny, czyli ilustruje jego perspektywę zewnętrzną.

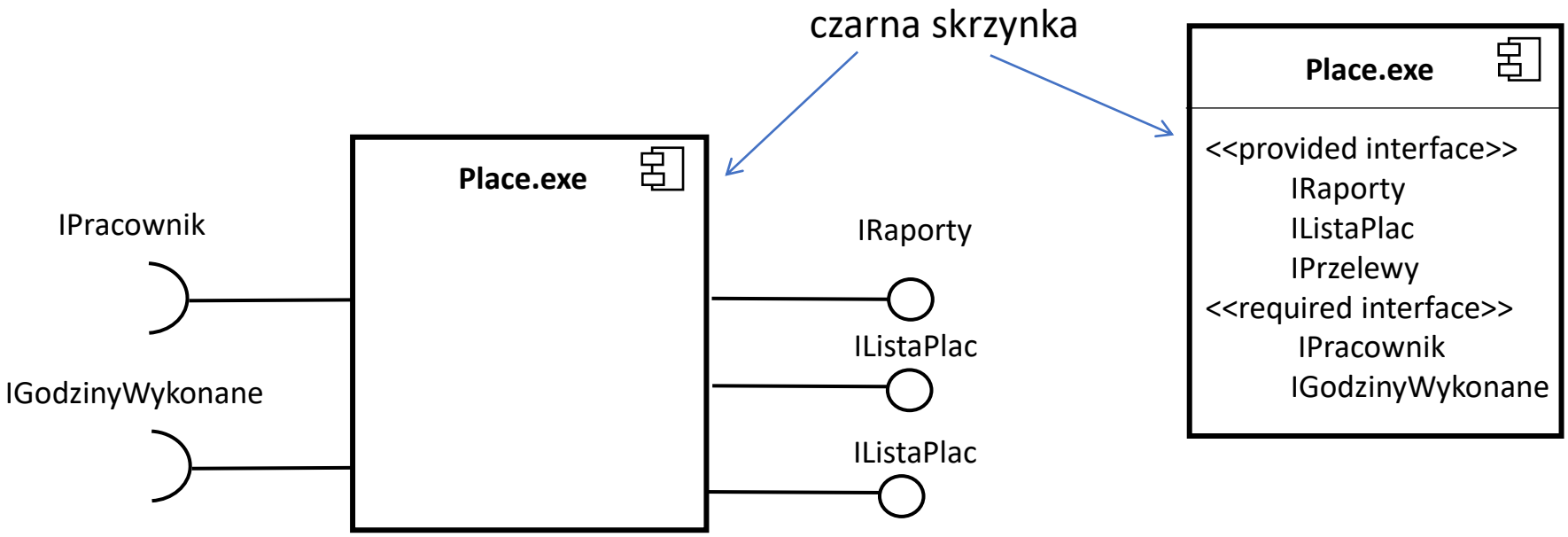


Diagram komponentów – Biała skrzynka

Biała skrzynka (ang. white box) zawiera specyfikację zawartości komponentu, czyli jego perspektywę wewnętrzną. Zachowanie komponentu jest realizowane przez instancje jego wewnętrznych klasyfikatorów. Specyfikacji udziału klasyfikatorów można dokonać poprzez wprowadzenie dodatkowych sekcji komponentu, opisanych stereotypami tekstowymi.

Poza standardową sekcją interfejsów pozyskujących i udostępniających specyfikacja ta zawiera sekcje:

- klasyfikatorów – <<realizations>>- wskazuje się nazwy instancji klasyfikatorów, które są odpowiedzialne za realizacjęzachowania komponentu
- artefaktów – <<artifacts>>

Diagram komponentów – Biała skrzynka

Biała skrzynka

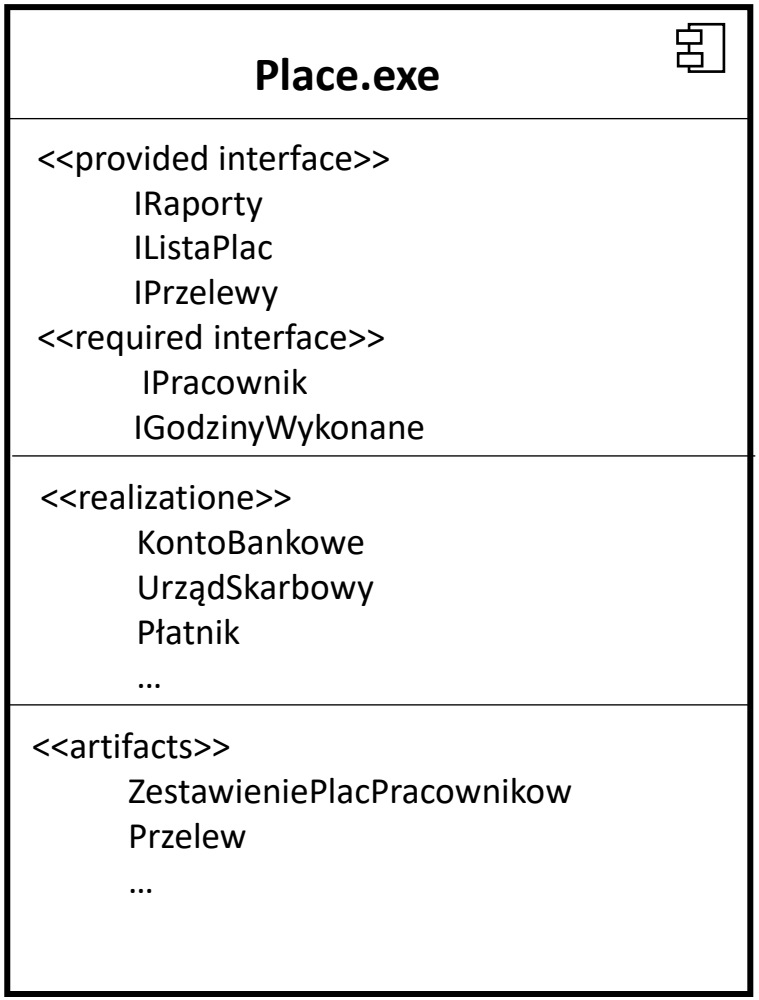


Diagram wdrożenia

Diagram wdrożenia (ang. *deployment diagram*) to rodzaj diagramu UML, który określa reprezentację fizyczną architektury rozwiązania, określa fizyczny sprzęt na którym będzie realizowany system.

Diagramy wdrożenia pokazują relacje między komponentami oprogramowania a sprzętem a także relacje pomiędzy poszczególnymi elementami infrastruktury technicznej.

Na diagramie wdrożenia prezentuje się węzły systemu, którymi są komputery, serwery i inny sprzęt komputerowy funkcjonujący w działającym systemie oraz związki między nimi.

Diagram wdrożenia – uwagi

Diagram wdrożenia

- odwzorowuje architekturę oprogramowania utworzoną w projekcie na fizyczną architekturę systemu, która go wykonuje.
- w systemach rozproszonych modeluje dystrybucję oprogramowania w węzłach fizycznych.
- wizualizuje, jak oprogramowanie współdziała ze sprzętem w celu zrealizowania pełnej funkcjonalności systemu.
- służy do opisu interakcji oprogramowania ze sprzętem i odwrotnie.
- diagramy wdrożenia są używane wyłącznie w celu opisania sposobu wdrażania oprogramowania w systemie sprzętowym.

Diagram wdrożenia – uwagi

Diagramy wdrożenia rzadko są używane przy modelowaniu mniejszych i średnich systemów, dlatego zwykle ich rola jest ograniczona.

Ponieważ posługują się zaledwie kilkoma symbolami, dlatego kluczową rolę odgrywają stereotypy nadawane poszczególnym elementom.
Pozwalają one doprecyzować znaczenie i funkcję oprogramowania oraz sprzętu.

Diagramy wdrożenia odgrywają istotną rolę przy wdrażaniu dużych, rozproszonych systemów.

Diagram wdrożenia – notacja i symbole

Diagram wdrożenia zawiera następujące elementy:

- **artefakt**
- **węzeł**
- **ścieżka komunikacyjna**

Diagram wdrożenia – notacja i symbole

Diagram wdrożenia przedstawia powiązania między oprogramowaniem (artefaktami) a sprzętem (węzłami).

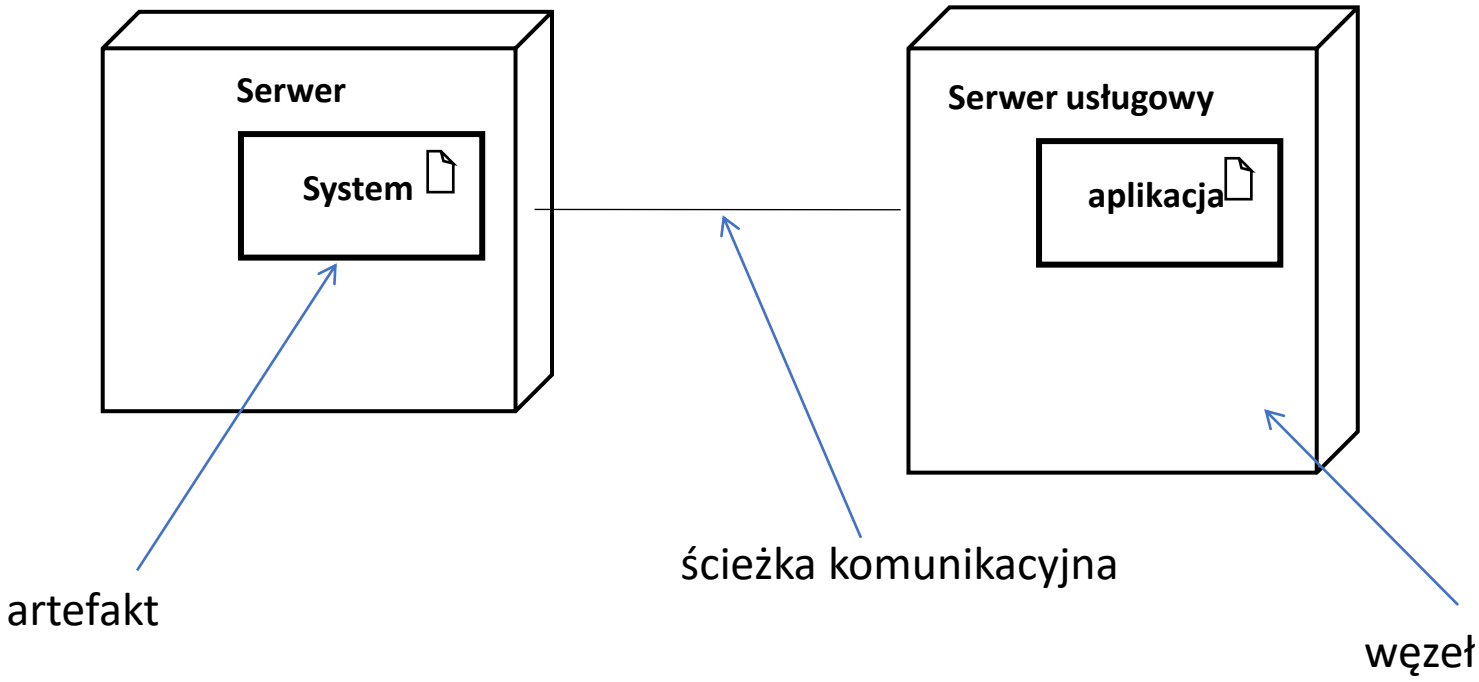
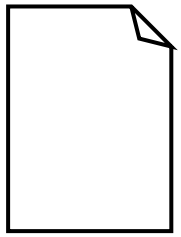


Diagram wdrożenia – artefakt

Artefakt (ang. *artifact*) – każda wytworzona lub użyta jednostka fizyczna (produkt informatyczny) wykorzystany w systemie oprogramowania.



ikona artefaktu

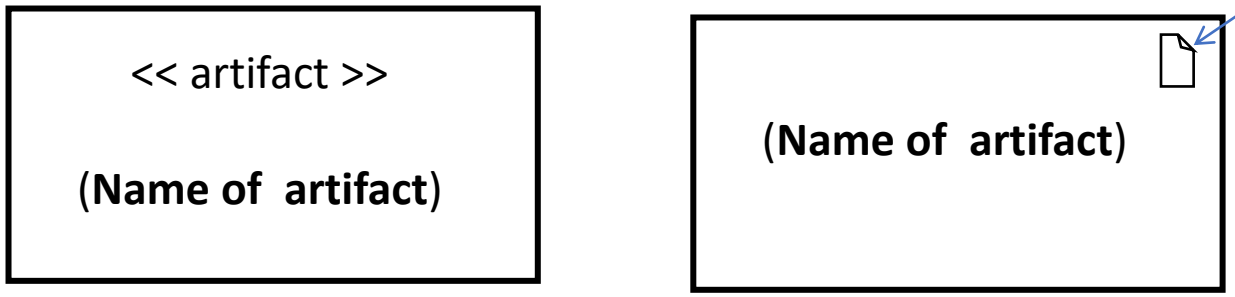


Diagram wdrożenia – artefakt

Przykłady artefaktów:

- pakiet oprogramowania
- baza danych
- pliki źródłowe
- pliki wykonywalne
- arkusz kalkulacyjny
- specyfikacja wymagań systemu
- klasa
- model biznesowy
- scenariusz przypadku użycia
- plan testów
- raporty z testów
- podręczniki użytkownika
- komponent

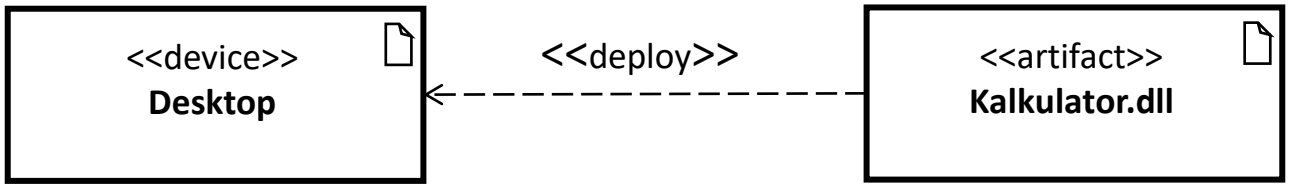
Diagram wdrożenia – stereotypy artefaktów

Artefakt może posiadać stereotyp

- *executable* – programy wykonywalne
- *file* – pliki
- *sourse* – pliki z kodem źródłowym
- *library* – biblioteki programów
- *table* – fizyczne bazy danych i tabele baz danych;
- *service* – komponenty przetwarzające
- *subsystem* – podsystemy

Diagram wdrożenia – metody wdrażania artefaktu

Artefakt poza węzłem



Artefakt wewnątrz węzła

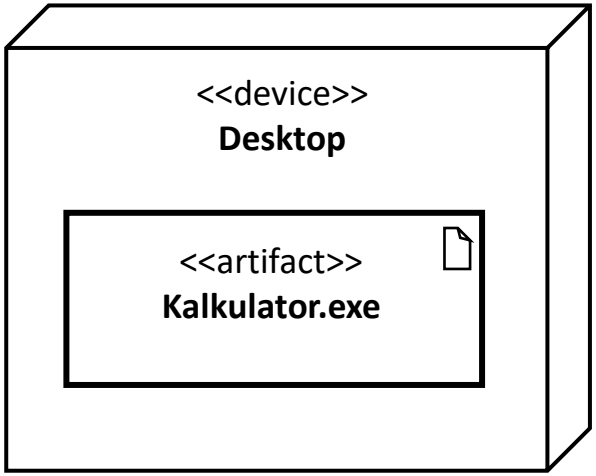


Diagram wdrożenia – zależność między artefaktami

W przypadku, gdy więcej niż jeden artefakt jest wdrożony na danym węźle sprzętowym, lub gdy artefakty są rozmieszczone na różnych docelowych węzłach sprzętowych, może istnieć zależność między nimi (czyli jeden artefakt opiera się na usługach świadczonych przez inny artefakt).

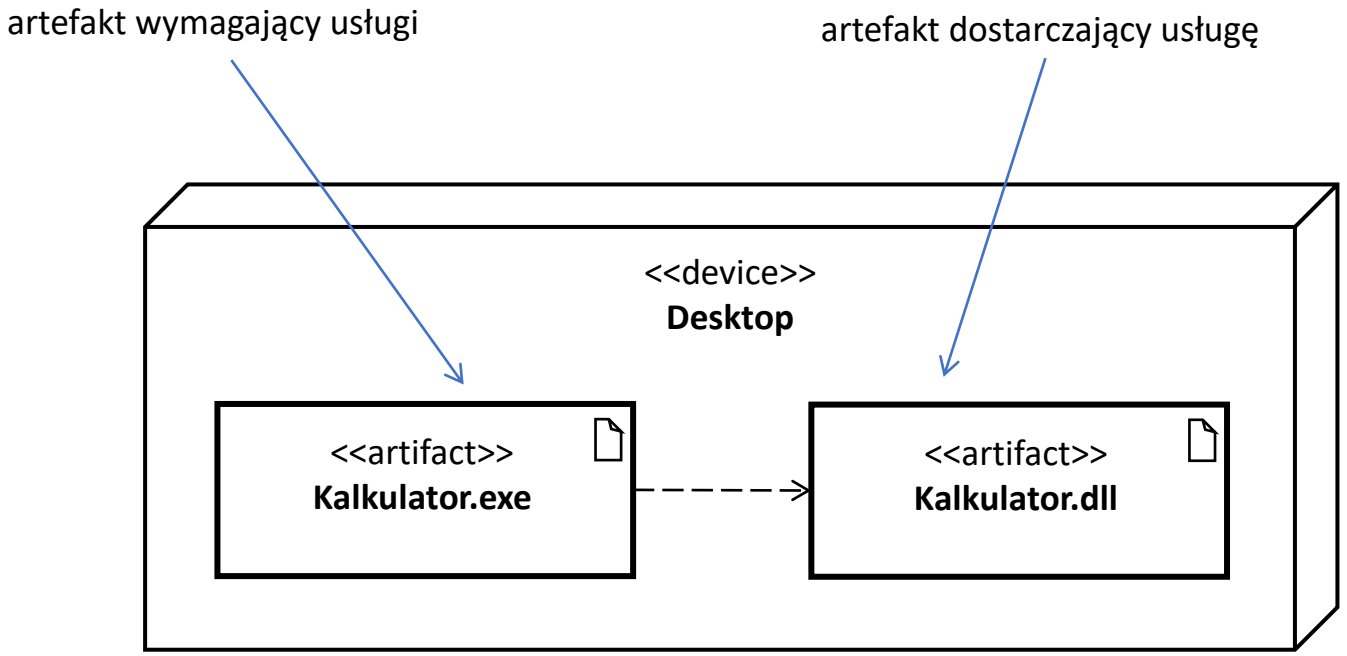


Diagram wdrożenia – relacja manifest dla artefaktów

Jeśli artefakt oznacza fizyczne wdrożenie elementu pakietowego takiego jak np. komponent, to mówi się, że artefakt reprezentuje (ang. *manifest*) ten element.

Element manifestowany jest dostępny poprzez interfejs artefaktu.

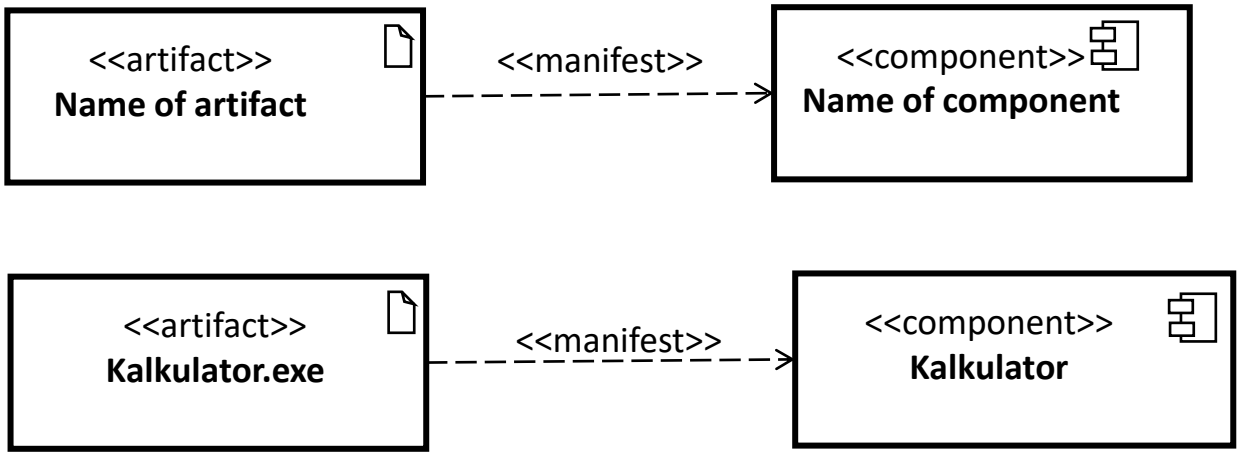


Diagram wdrożenia – zależność kompozycji dla artefaktów

Artefakty mogą być powiązane ze sobą za pomocą relacji kompozycji.

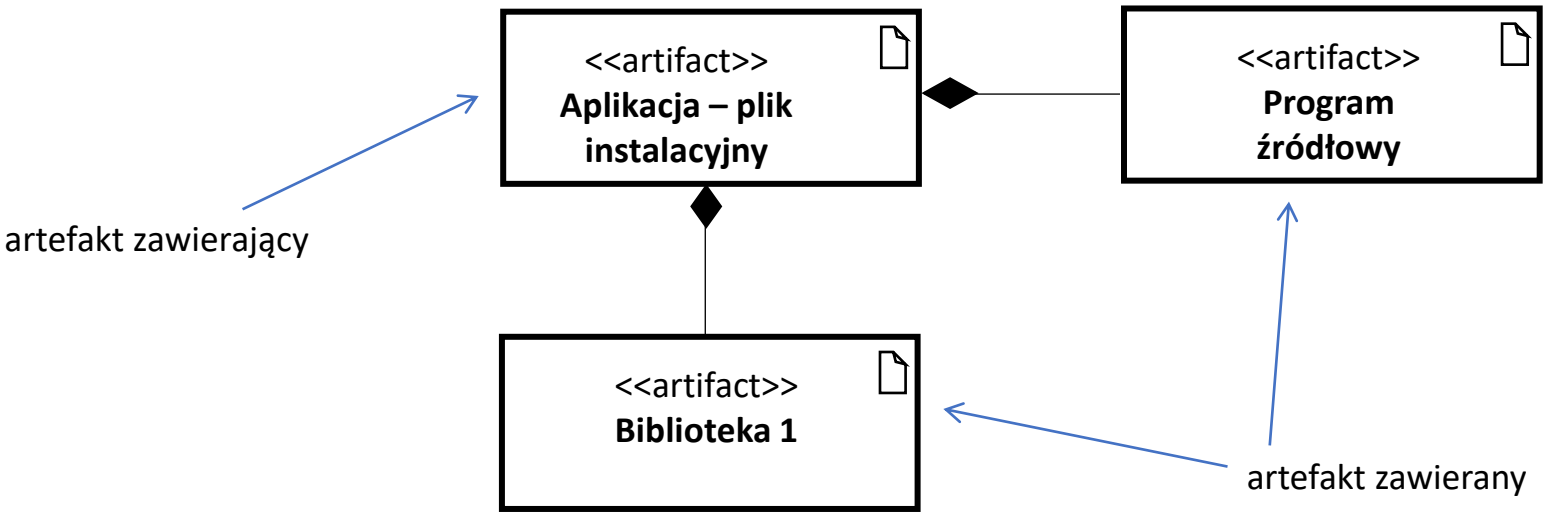


Diagram wdrożenia – instancje artefaktu

Instancja artefaktu – jest elementem modelu, który reprezentuje instancję lub aktualne wystąpienie artefaktu. Instancje artefaktów są oparte na istniejących artefaktach.

artifact- name : artifact- type

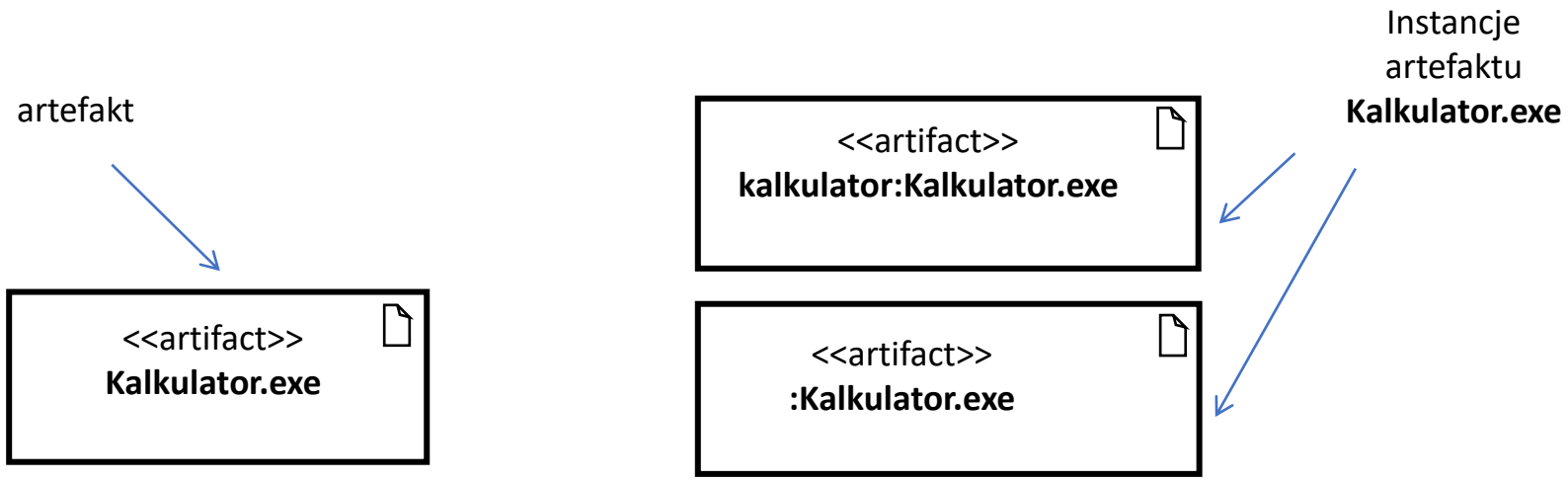
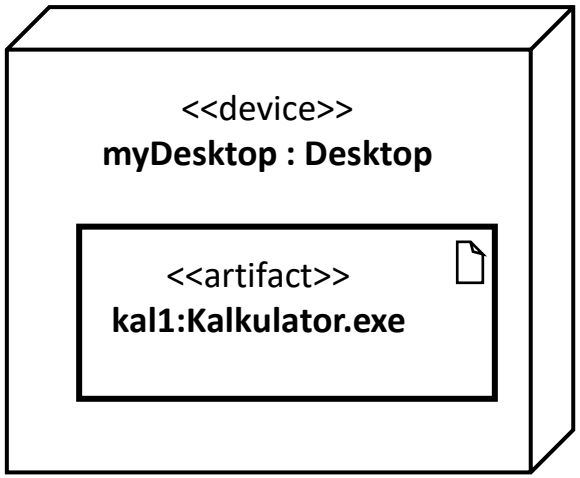


Diagram wdrożenia – instancje artefaktu

Artefakt po wdrożeniu do węzła sprzętowego lub określonego środowiska wykonawczego staje się instancją.

W różnych instancjach węzłów można wdrożyć różne instancje artefaktu, a każda instancja artefaktu może mieć inne wartości właściwości.



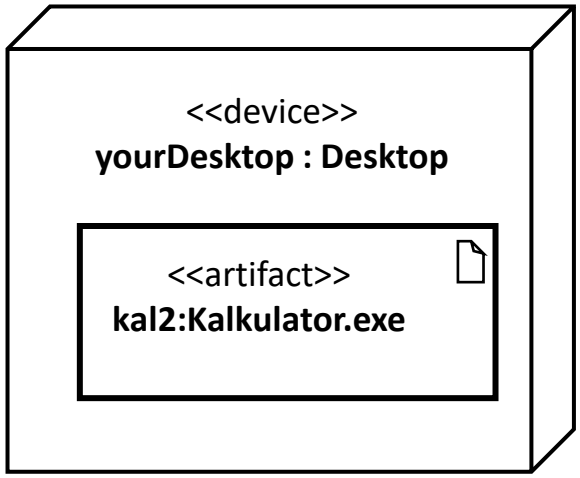


Diagram wdrożenia – specyfikacja wdrożenia

Specyfikacja wdrożenia (ang. *deployment specification*) jest to szczególny przypadek artefaktu, który określa parametry mające być używane przez inny artefakt.

Specyfikacja wdrożenia jest artefaktem ze stereotypem «deployment spec».

Diagram wdrożenia – specyfikacja wdrożenia

Zasadniczo jest to plik konfiguracyjny, taki jak dokument XML lub plik tekstowy, który definiuje sposób wdrażania artefaktu w węźle (zarówno sprzętowym jak i oprogramowania).

Specyfikacja zawiera listę tych właściwości, które muszą być zdefiniowane, aby nastąpiło wdrożenie artefaktu, składnika w węźle, tak jak to jak przedstawiono na diagramie wdrażania.

Parametry mogą obejmować współbieżność, wykonanie i opcje specyficzne dla transakcji, które są wyrażone jako atrybuty.

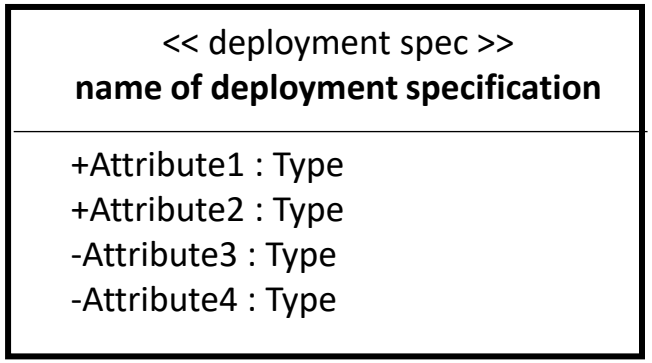
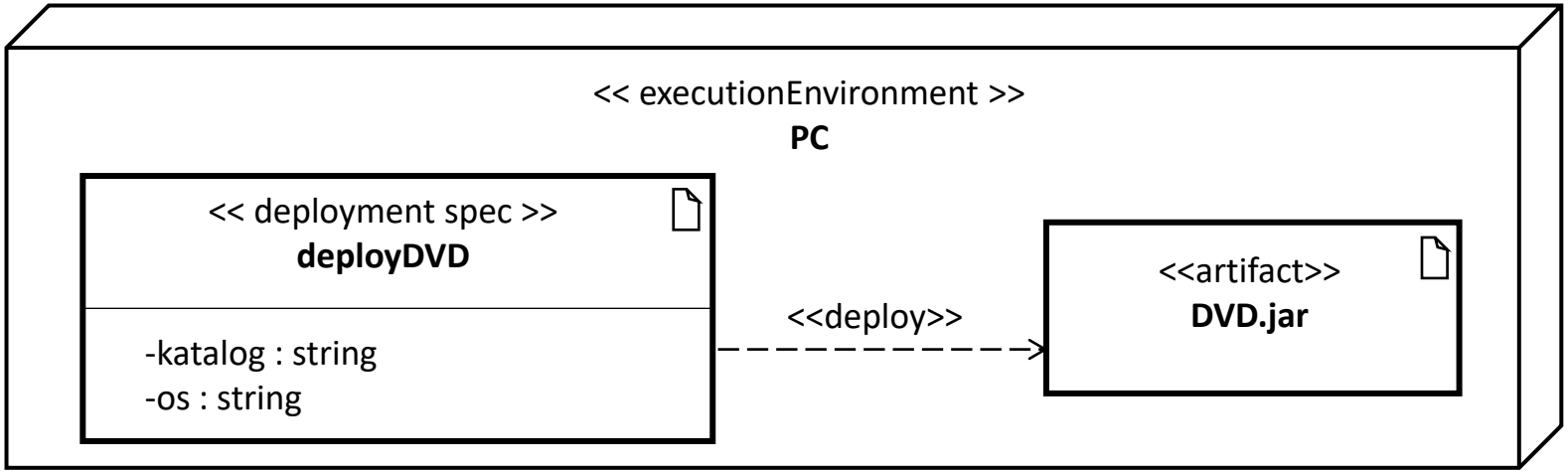


Diagram wdrożenia – specyfikacja wdrożenia

Specyfikacja wdrożenia określa właściwości, które definiują parametry wykonania składnika lub artefaktu wdrożonego w węźle.

Parametry mogą obejmować współbieżność, wykonanie i opcje specyficzne dla transakcji, które są wyrażone jako atrybuty.



artefakt DVD.jar jest wdrażany do środowiska wykonawczego PC

Diagram wdrożenia – specyfikacja wdrożenia

Inny sposób reprezentacji wdrożenia artefaktu DVD.jar do środowiska wykonawczego PC

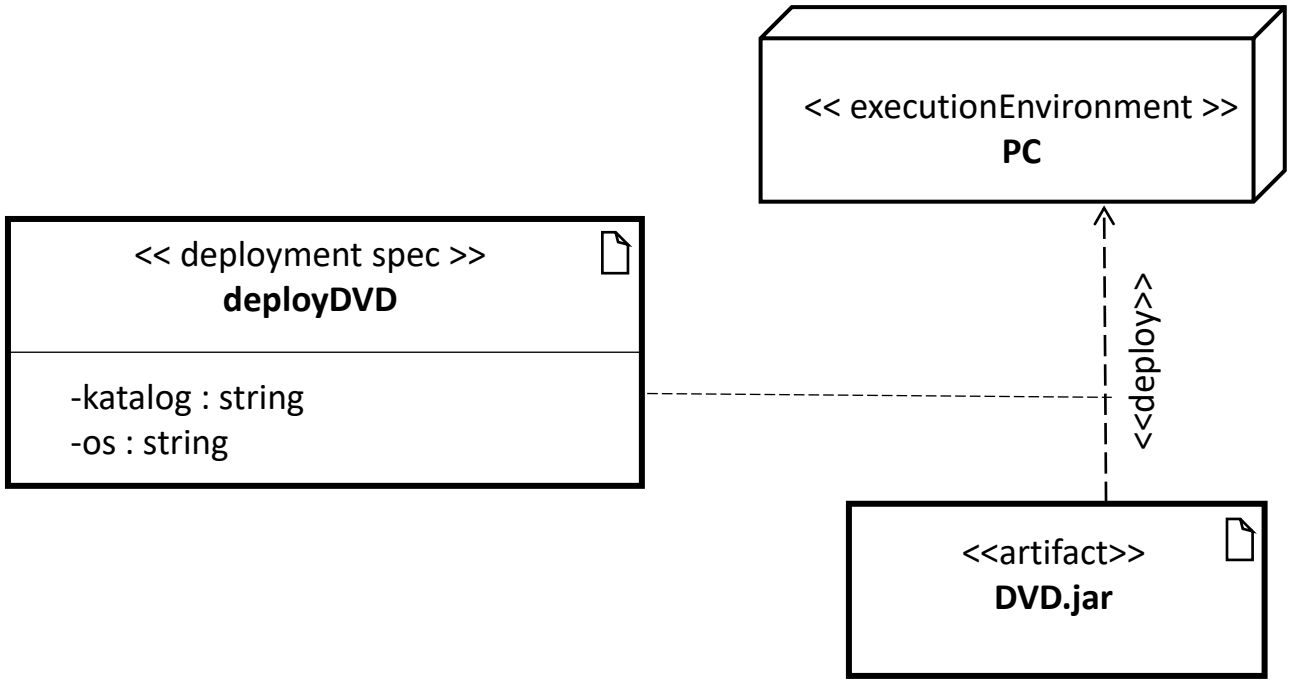
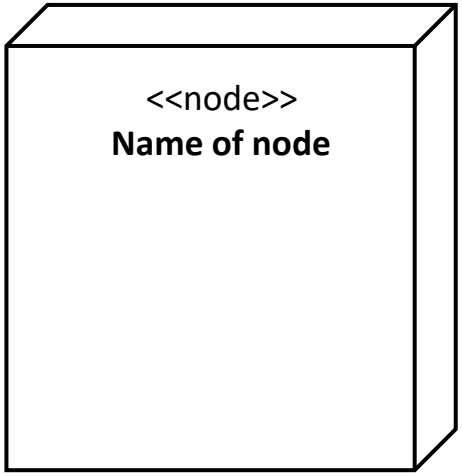
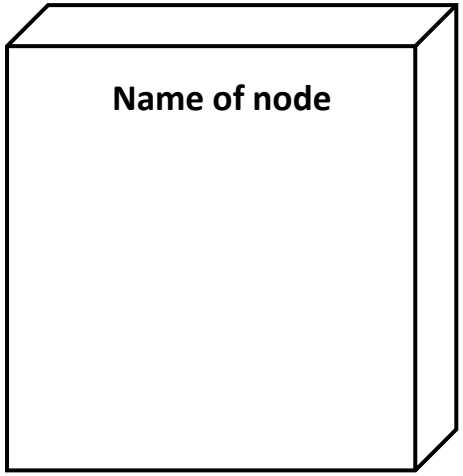


Diagram wdrożenia – węzeł

Węzeł (ang. *node*) – to fizyczne urządzenie, na którym wdrożona jest aplikacja. Węzeł zazwyczaj zawiera komponenty i inne wykonywalne fragmenty kodu.





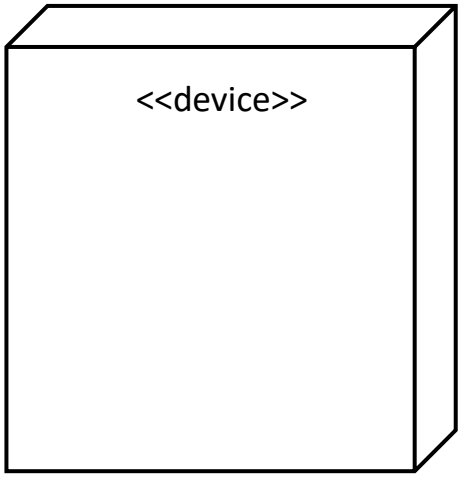
Dwa podstawowe typy węzłów:

- sprzętowym <<device>>
- oprogramowania << executionEnvironment >>

Diagram wdrożenia – węzeł

Dwa podstawowe typy węzłów:

- sprzętowym <<device>>
- oprogramowania << executionEnvironment >>



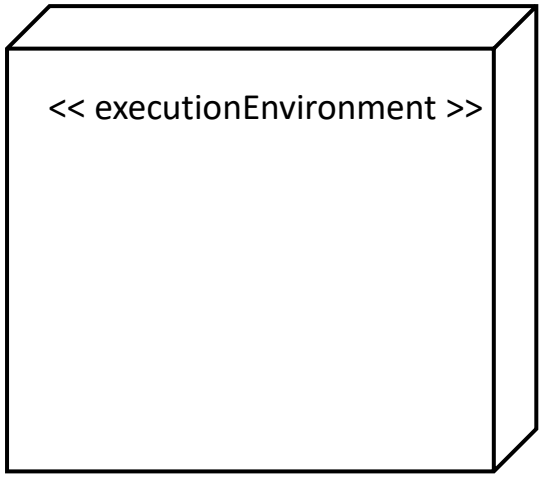
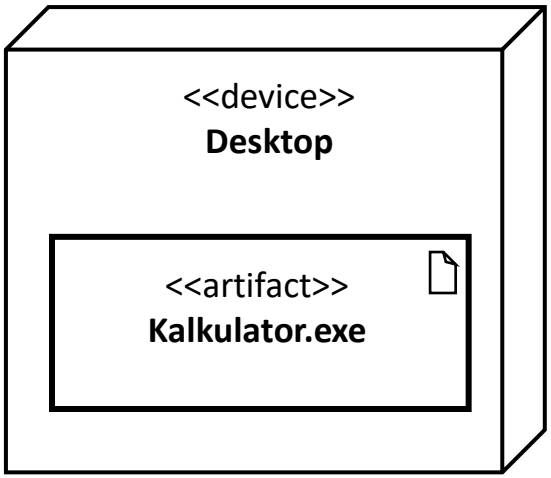


Diagram wdrożenia – przykład

Najprostszy diagram wdrożenia, to taki, który składa się z pojedynczego pliku wykonywalnego (.exe), uruchamiany na jednym komputerze.

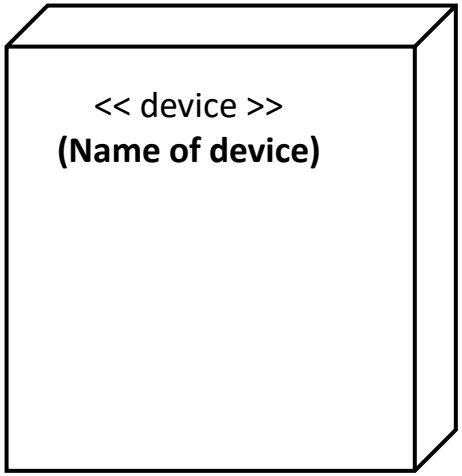


Artefakt umieszczany jest wewnątrz węzła, aby pokazać, że artefakt jest wdrażany w tym węźle.

Diagram wdrożenia – urządzenie

Urządzenie (ang. *device*) – jest rodzajem węzła, który reprezentuje fizyczny zasób obliczeniowy w systemie, taki jak serwer aplikacji.

Urządzenie może składać się z innych urządzeń, na przykład serwer aplikacji zawiera procesor.
Urządzenie jest węzłem ze stereotypem <<device>>.



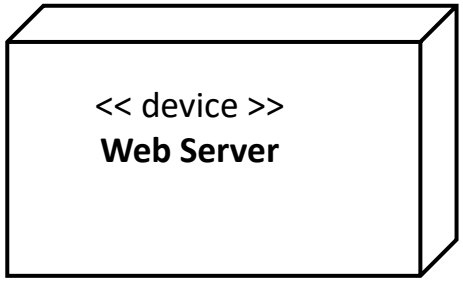


Diagram wdrożenia – urządzenie

Różnica pomiędzy zwykłym węzłem (*node*) a urządzeniem (*device* – *hardware node*) polega na tym, że urządzenia są używane w sytuacjach, w których przedstawiany element jest typową maszyną – sprzętem komputerowym.

Na diagramie wdrożenia urządzeniami mogą być:

- serwer aplikacji
- stacja robocza klienta
- router
- switch
- firewall
- laptop
- telefon komórkowy
- urządzenie wbudowane

Diagram wdrożenia – środowisko wykonawcze

Środowisko wykonawcze (ang. *Execution Environment*) to węzeł, który oferuje środowisko dla określonych typów komponentów, które są na nim wdrożone w postaci wykonywalnych artefaktów. Węzeł ten nazywany jest też węzłem oprogramowania. Jest to węzeł ze stereotypem << *executionEnvironment* >>.

Ten rodzaj węzła reprezentuje konkretną platformę wykonawczą, taką jak system operacyjny lub system zarządzania bazą danych. Środowisk wykonawczych można użyć do opisania kontekstu, w którym odbywa się wykonanie modelu.

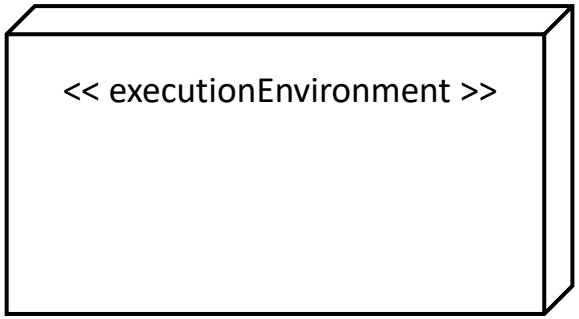


Diagram wdrożenia – środowisko wykonawcze

Przykłady węzłów oprogramowania:

- środowisko bazy danych
- system operacyjny
- serwer aplikacji
- oprogramowanie serwera WWW (Apache, Microsoft Internet Information Server
- środowiska programistyczne (np. Java Runtime Environment – JRE)

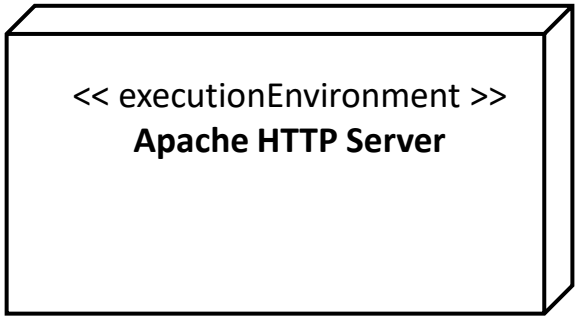


Diagram wdrożenia – relacja kompozycji

Węzeł reprezentujący oprogramowanie może być powiązany z węzłem sprzętowym poprzez relacje kompozycji.

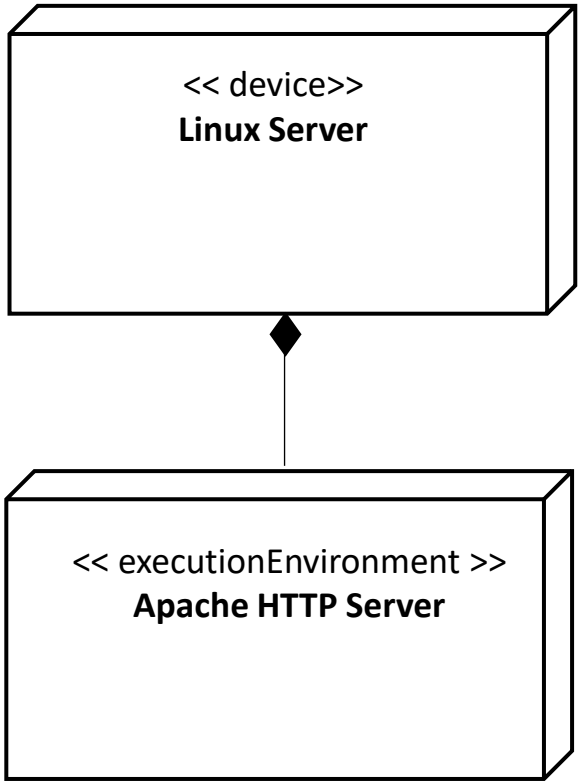


Diagram wdrożenia – stereotypy węzłów

Inne stereotypy węzłów:

- *cdrom*
- *cd-rom*
- *computer*
- *disk array*
- *pc*
- *pc client*
- *user pc*
- *pc server*
- *server*
- *secure*
- *storage*
- *database system*
- *datebase*
- *JSP server*

Diagram wdrożenia – baza danych

Węzeł **baza danych** reprezentuje dane przechowywane przez wdrożony system. Jest to węzeł ze stereotypem <<database>> lub <<database system>>.

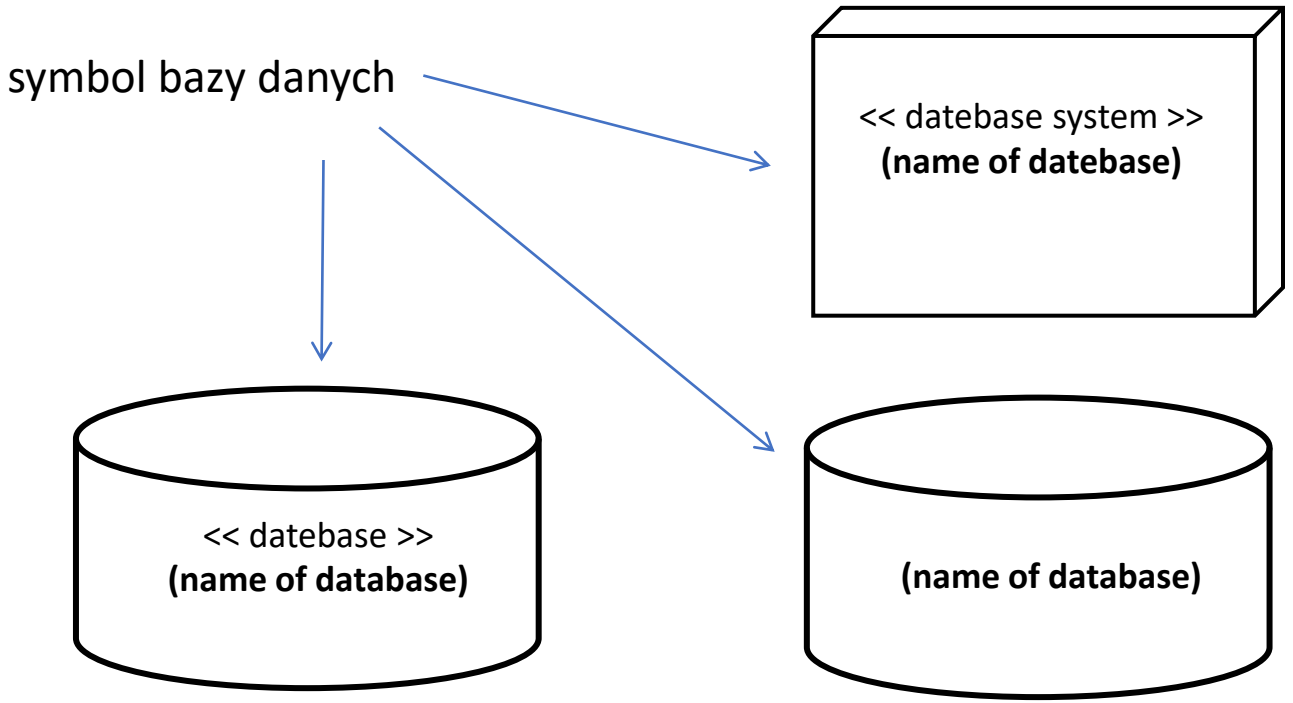


Diagram wdrożenia – instancja węzła

Instancja węzła – jest elementem modelu, który reprezentuje instancję lub rzeczywiste wystąpienie węzła. Instancje węzłów są oparte na istniejących węzłach.

node – name : node – type

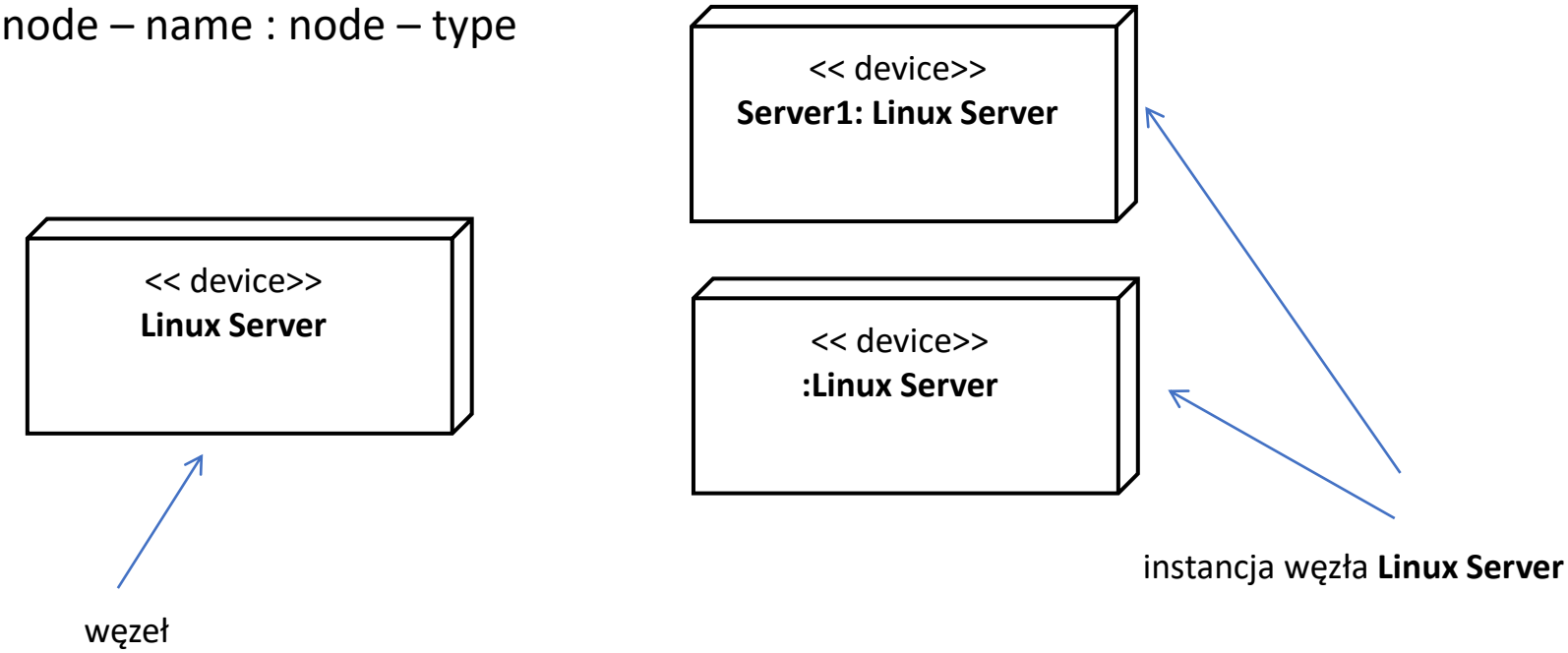


Diagram wdrożenia – węzeł jako kontener

Węzeł jako kontener(ang. *node as contener*) to węzeł, który zawiera w sobie inny/inne węzły (zagnieżdżanie).

W modelowaniu UML można zagnieżdżać węzły w węzłach, aby reprezentować komponenty sprzętowe i programowe w systemie, który zawiera inne komponenty.

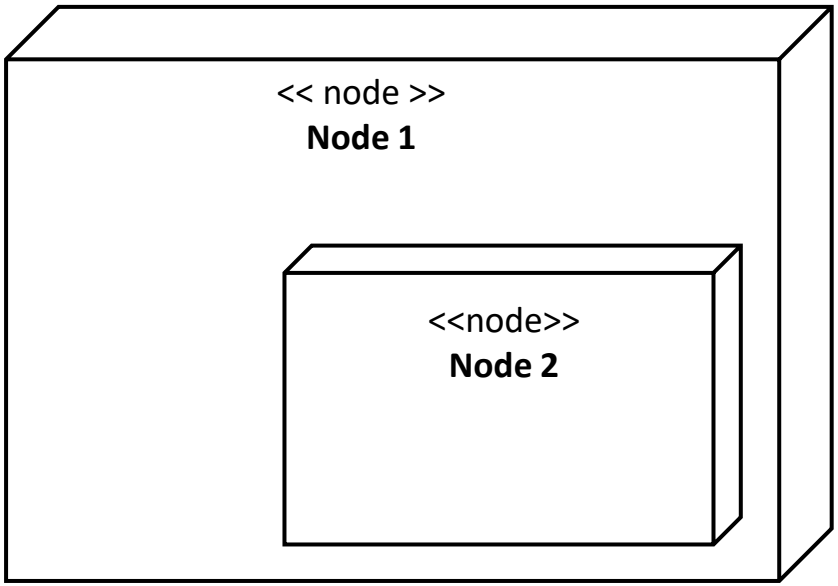


Diagram wdrożenia – ścieżka komunikacyjna

Ścieżka komunikacyjna (ang. *communication path*) – jest rodzajem powiązania (asocjacji) między węzłami na diagramie wdrażania, który pokazuje, w jaki sposób węzły wymieniają wiadomości i sygnały.

Podczas określania ścieżek komunikacyjnych można zdefiniować liczbę węzłów, które można podłączyć na każdym jej końcu, można też użyć etykiet do identyfikacji typu protokołu lub sieci używanej w komunikacji.

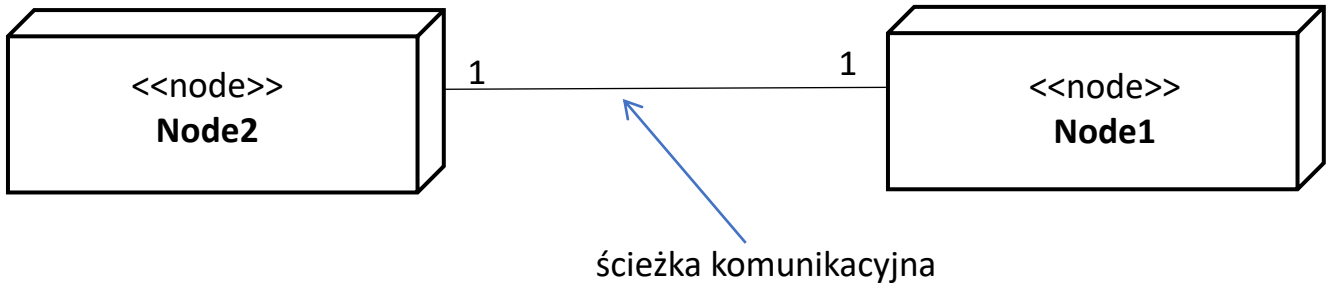
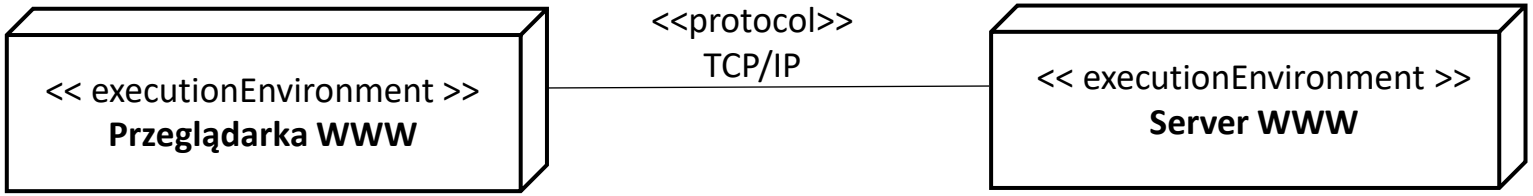


Diagram wdrożenia – ścieżka komunikacyjna

komunikacja pomiędzy dwoma urządzeniami



komunikacja pomiędzy dwoma węzłami oprogramowania

Diagram wdrożenia – symbole specjalne

W UML istnieje możliwość zastąpienia symboli standardowych symbolami specjalnymi, które mogą znacząco ułatwiać percepcję diagramów.