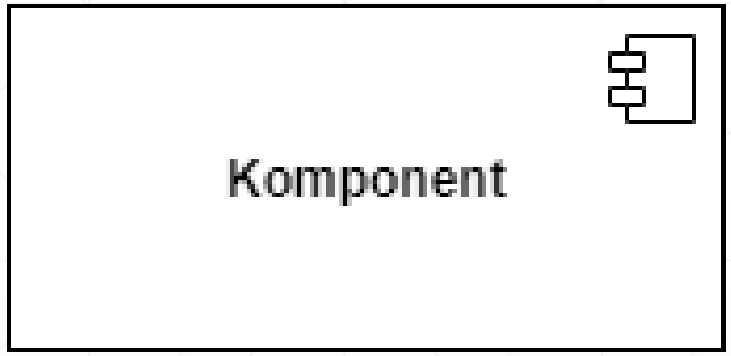
Diagram komponentów

Autor: Aneta Pędzińska

Diagram komponentów umożliwia przedstawienie konstrukcji systemu z najwyższego poziomu abstrakcji w celu rozpoczęcia projektowania struktury. Wytyczne przekazujemy programistom, a project menagerowi prezentujemy złożoność systemu oraz możliwy podział na etapy wdrożenia.

Komponent



Komponenty reprezentują części systemu wydzielone podczas projektowania. Grupują one wykonywane często wspólnie funkcje i oddzielają te, które współpracują rzadziej. Komponenty wymieniają dane między sobą za pomocą interfejsów. Nie mają możliwości komunikacji w żaden inny sposób. Ich wewnętrzna struktura, dane i operacje pozostają ukryte (enkapsulacja). Pozwala to na wprowadzanie zmian w komponencie, które są niewidoczne dla pozostałych i dalsze komunikowanie się w ten sam sposób za pomocą niezmienionych interfejsów. Dzięki temu podziałowi można także ponownie wykorzystywać komponenty w innych systemach.

Stereotypy komponentów

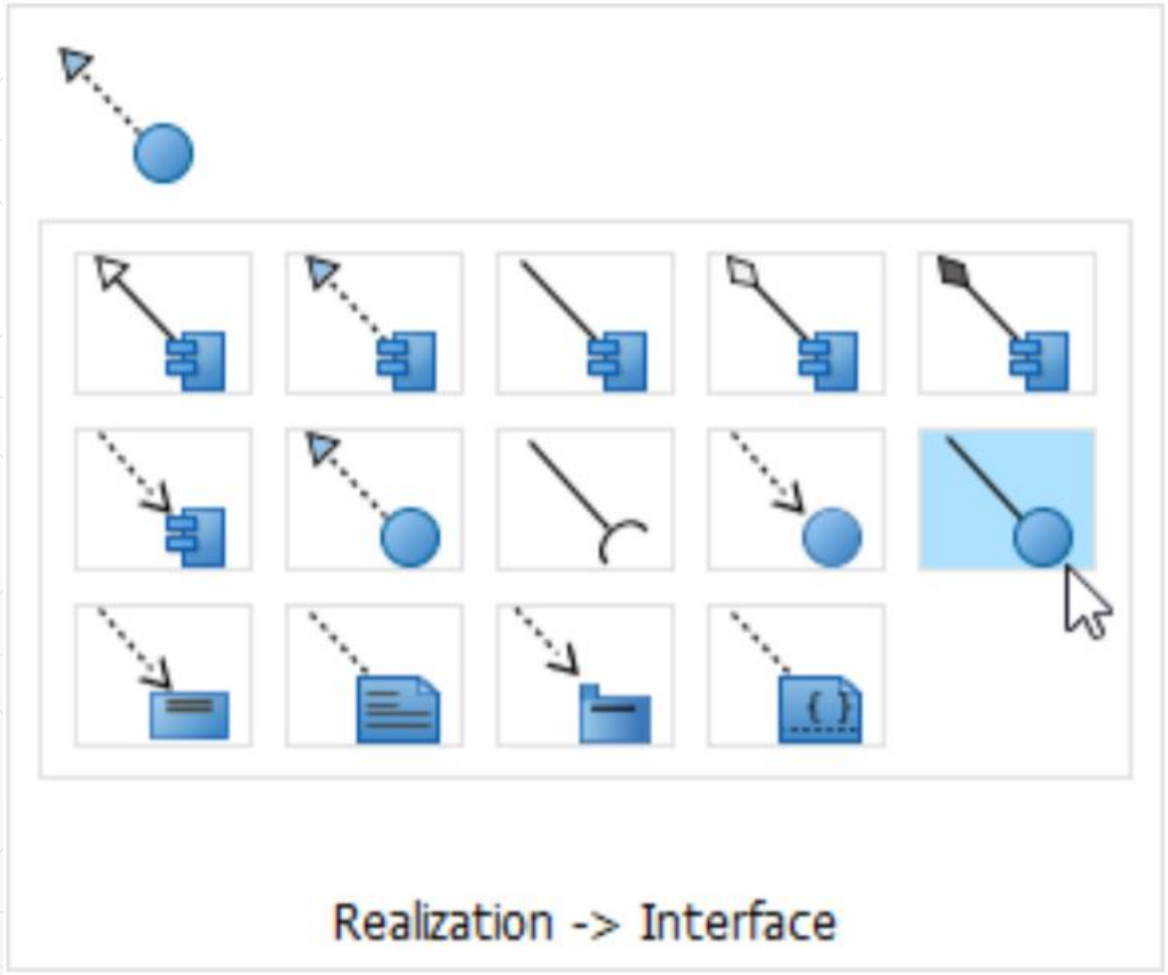
- executable - określa komponent, który można wykonać na węźle
- library - określa dynamiczną lub statyczną bibliotekę obiektów
- table - określa komponent reprezentujący tabelę bazy danych
- file - określa komponent reprezentujący dokument zawierający kod źródłowy lub dane
- document - określa komponent reprezentujący dokument

Interfejs

Interfejs (ang. interface) jest to zestaw operacji, które wyznaczają usługi oferowane przez komponent (lub klasę).

Rodzaje interfejsów

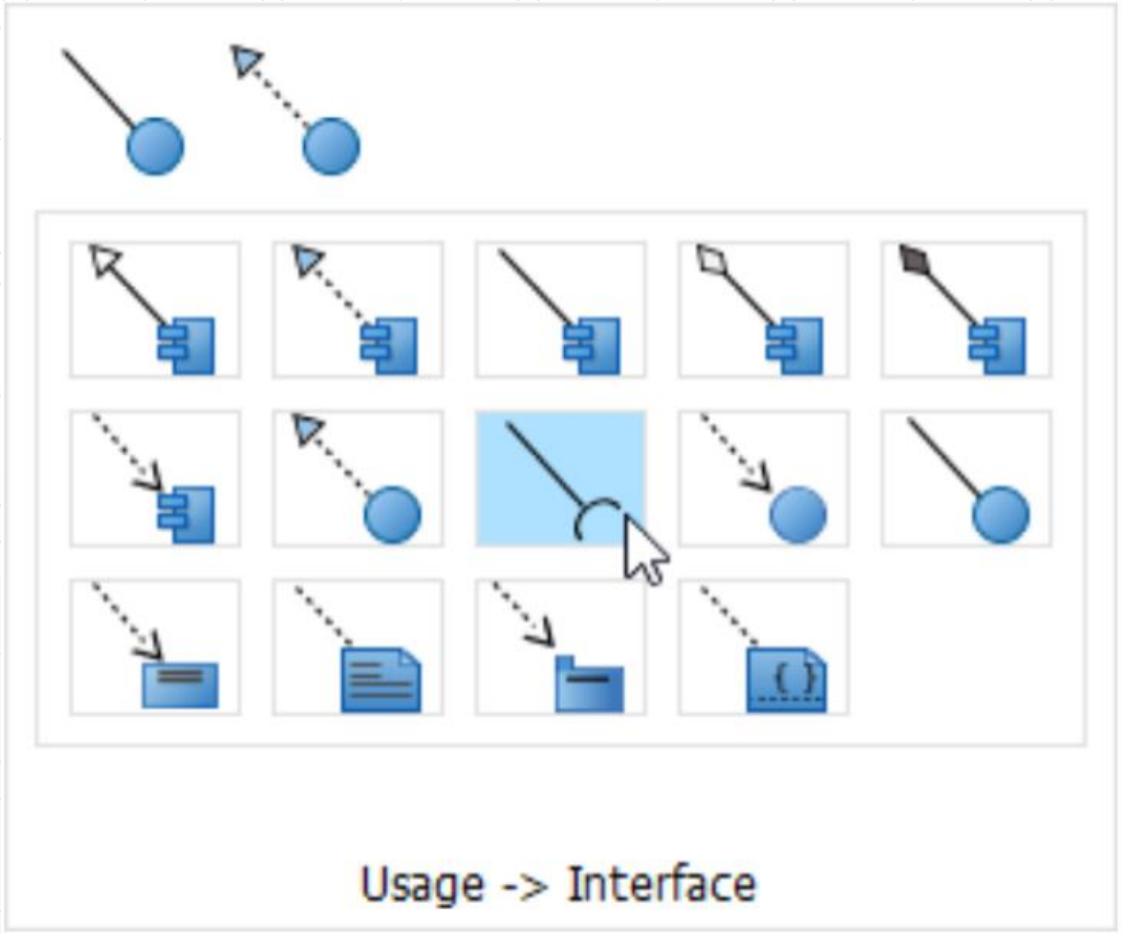
- Interfejs dostarczany – umożliwia pozyskanie danych z komponentu



Udostępniane (dostarczane) interfejsy są zaznaczone kółkami

Rodzaje interfejsów

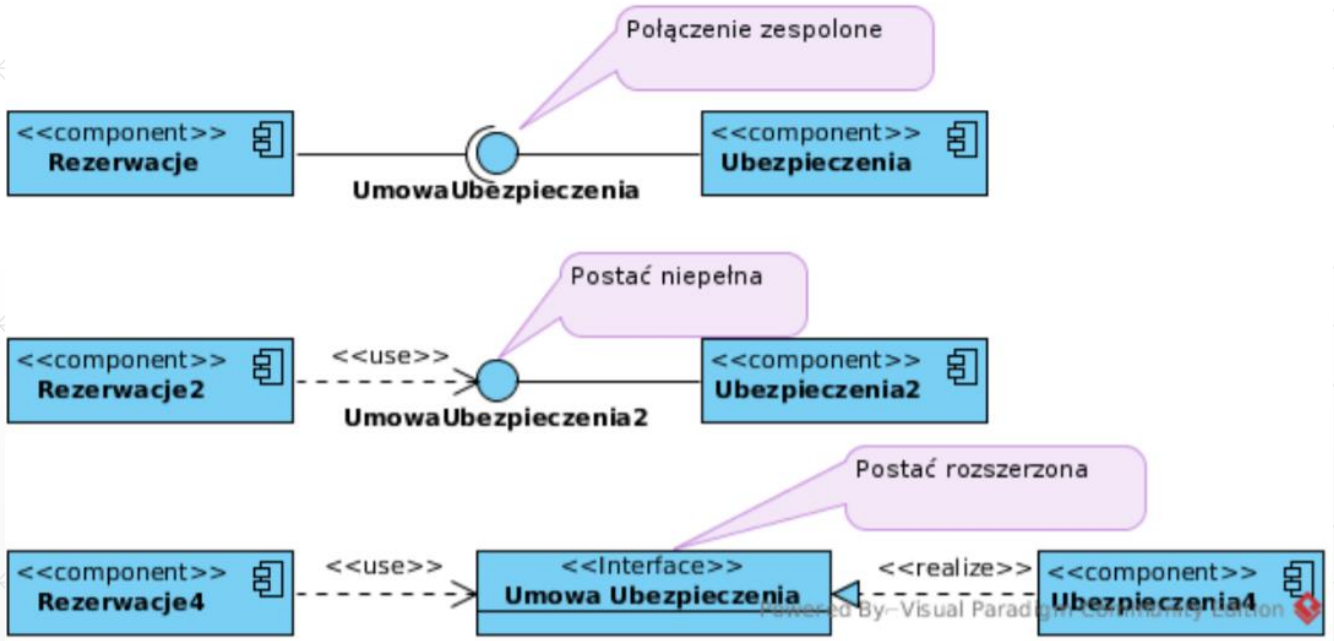
- Interfejs wymagany – wymaga dostarczenia określonych danych do wykonania operacji



Interfejsy, których dany komponent wymaga do swojego działania, są oznaczone półokręgiem - gniazdem.

Połączenie zespolone

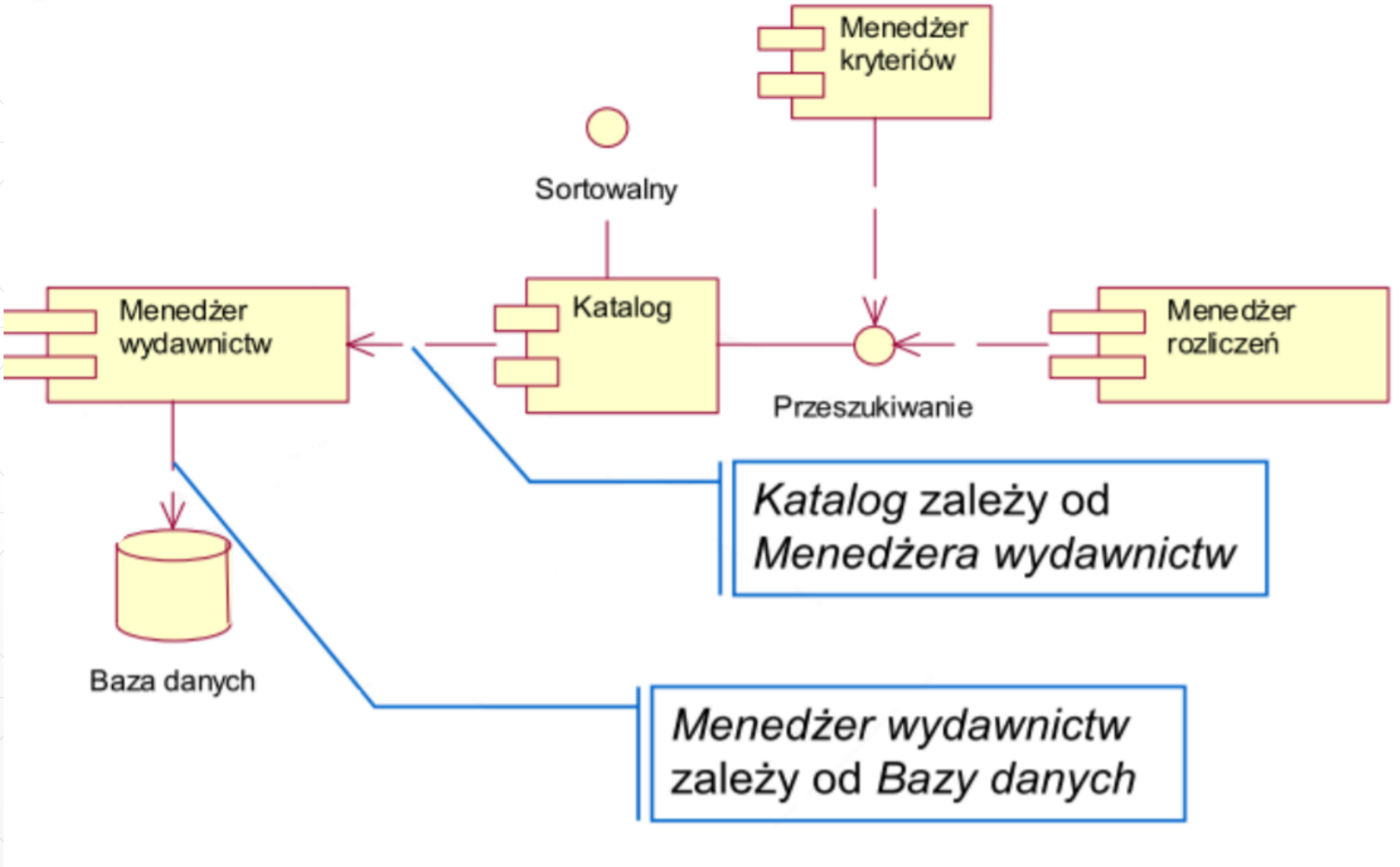
- Połączenie zespolone (ang. assembly connector) jest specyficzną formą połączenia, które pokazuje, że komponent dostarcza usługi, które są wymagane przez inny komponent.



Połączenie zespolone tym różni się od pozostałych dwóch interfejsów (dostarczającego i wymagającego), że wskazuje takie usługi, na które ma zawsze zapotrzebowanie inny komponent.

Zależności

Komponenty są między sobą powiązane relacją zależności, ponieważ wymagają ich do 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. Ilość i jakość tych zależności ma duże znaczenie dla oceny jakości modelu i projektu: duża liczba powiązań pomiędzy komponentami, a w szczególności zależności cykliczne, w znacznym stopniu utrudniają wyznaczanie obszarów zmienności i ich hermetyzację, a co za tym idzie - podnoszą koszt pielęgnacji oprogramowania. W odróżnieniu od tego, system o dobrze zdefiniowanych interfejsach komponentów pozwala na ich wymianę bez potrzeby modyfikacji pozostałej części systemu.


Menedżer kryteriów
Sortowalny
Menedżer wydawnictw
Katalog
Przeszukiwanie
Menedżer rozliczeń
Baza danych
Katalog zależy od Menedżera wydawnictw
Menedżer wydawnictw zależy od Bazy danych

Widoki

- Widok czarnej skrzynki - zewnętrzny (external, black box) - nie są pokazane żadne szczegóły wewnętrzne danego komponentu.
- Widok białej skrzynki - wewnętrzny (internal, white box) - pokazujemy szczegóły wewnętrznej budowy komponentu, czyli jego części składowe

Porty

- Port jest cechą komponentu, która określa jego punkt interakcji pomiędzy wewnętrznymi elementami komponentu lub wewnętrznymi elementami a środowiskiem zewnętrznym, w jakim funkcjonuje komponent.

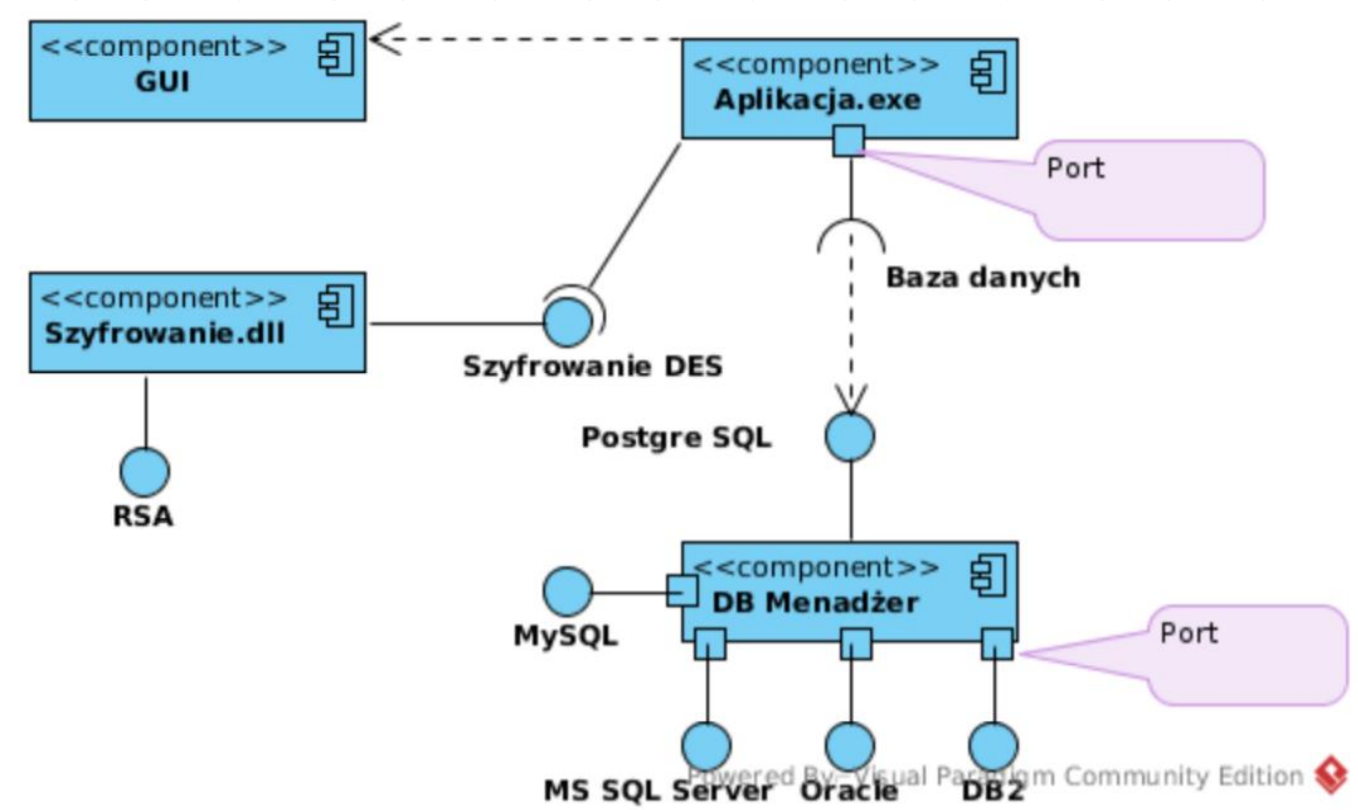


Diagram komponentów

Diagram komponentów przydaje się głownie wtedy, gdy chcemy podzielić system na prostsze elementy i pokazać zależności między nimi. Diagram komponentów dzieli system na fizyczne elementy oprogramowania: pliki, biblioteki, gotowe, wykonywalne programy itp.

Tworząc komponenty należy starać się aby:

- realizować w nich funkcjonalność
- uwzględnić w interfejsie wszelkie aspekty użycia komponentu
- uniezależnić stosowanie komponentu od zmienności sprzętu i oprogramowania systemowego
- uczynić komponent maksymalnie samowystarczalnym

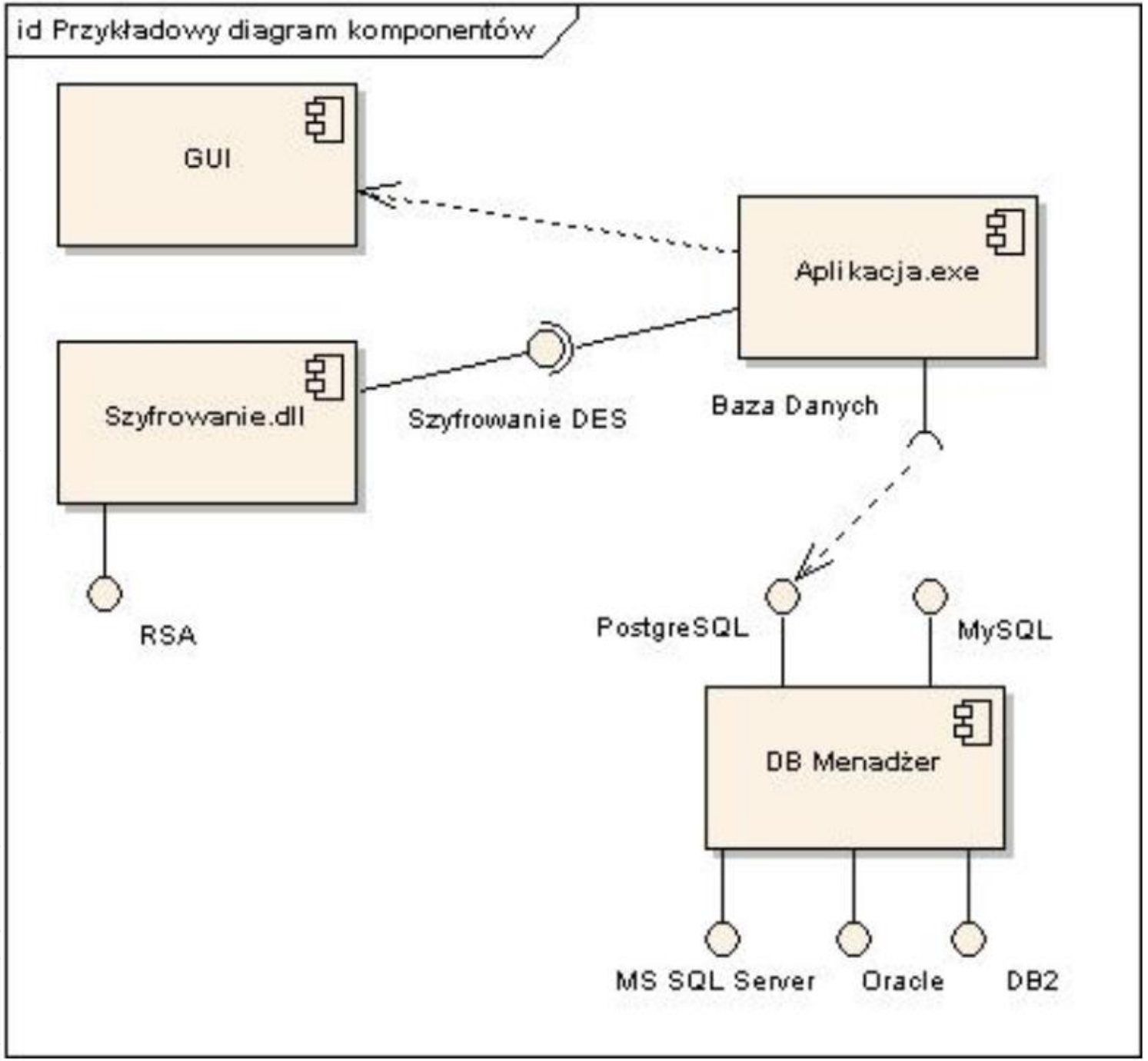
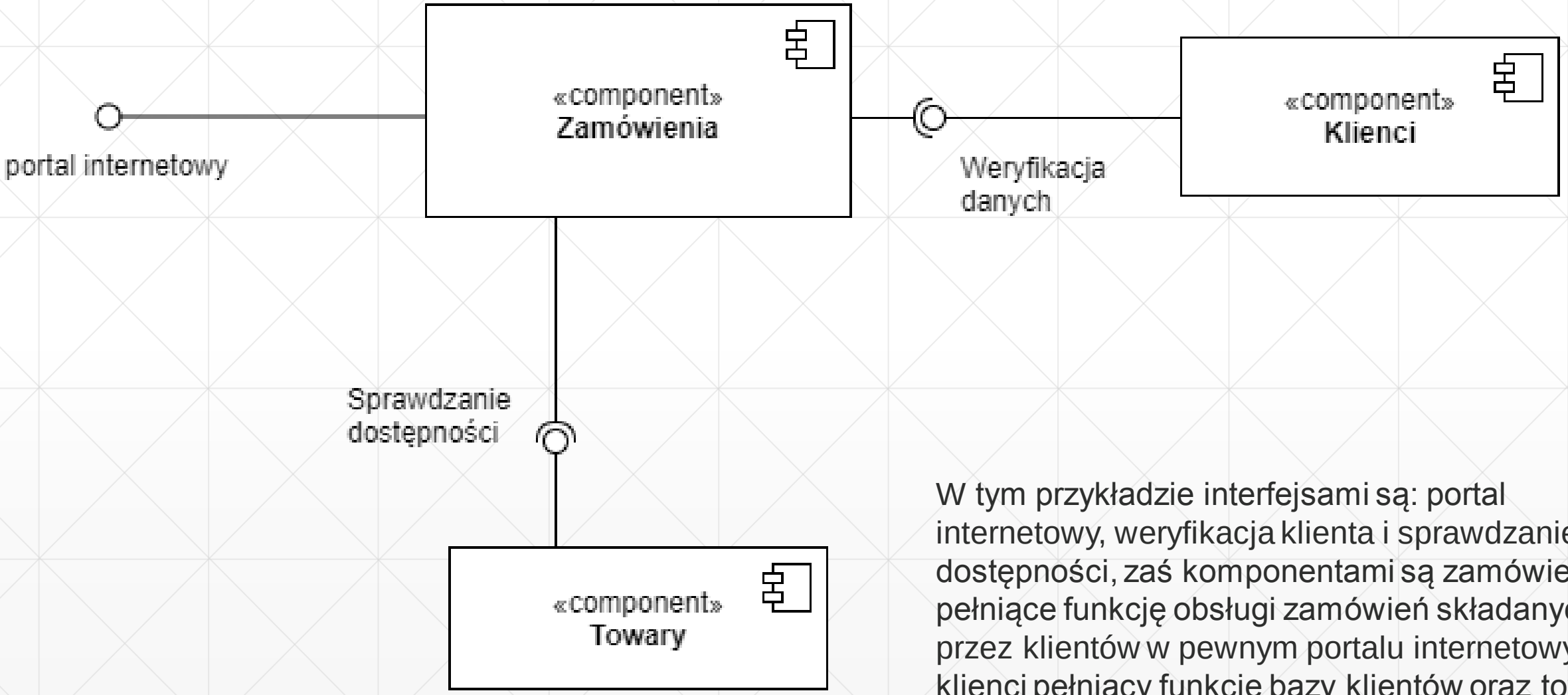

id Przykładowy diagram komponentów
GUI
Aplikacja.exe
Szyfrowanie.dll
Szyfrowanie DES
Baza Danych
RSA
PostgreSQL
MySQL
DB Menadżer
MS SQL Server
Oracle
DB2

Diagram komponentów

- Notacja kółko–gniazdo (ball and socket), przy pomocy której możemy pokazać połączenie dwóch komponentów poprzez interfejs została wprowadzona w UML 2.0, także na starszych diagramach jej nie spotkamy. Podobnie w UML 2.0 zmienił się sposób oznaczania komponentów, obecnie reprezentuje je prostokąt z ideogramem w prawym górnym rogu, dawniej komponent był oznaczany elementem, który stał się w wersji 2.0 ideogramem.

Przykład diagramu komponentów



W tym przykładzie interfejsami są: portal internetowy, weryfikacja klienta i sprawdzanie dostępności, zaś komponentami są zamówienia pełniące funkcję obsługi zamówień składanych przez klientów w pewnym portalu internetowym, klienci pełniący funkcję bazy klientów oraz towary pełniące funkcję bazy towarów.

Przykłady zastosowań

Diagram komponentów znajduje swoje zastosowanie podczas procesu projektowania systemów informatycznych ponieważ służy do określania szczegółów niezbędnych do budowy systemu.