Programowanie współbieżne i rozproszone

WYKŁAD 15

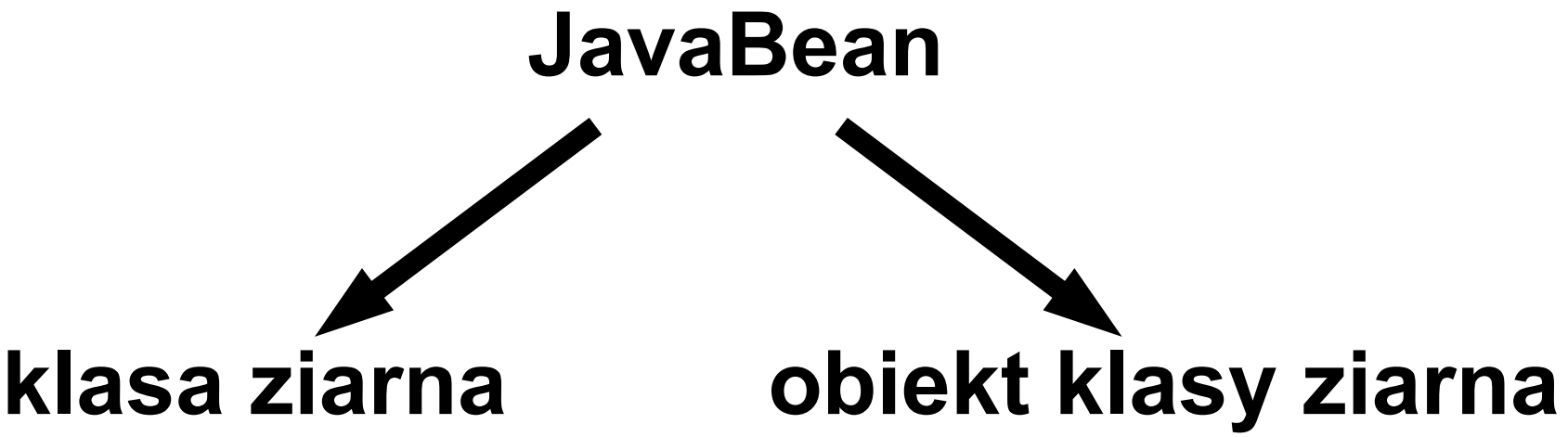
Komponenty

- Komponenty posiadają następujące cechy
  - są odpowiednio odizolowane od środowiska i innych komponentów
  - są wystarczająco samodzielne
  - są zaopatrzone w odpowiednią specyfikację, określającą jego wymagania i możliwości
  - obudowują swoją implementację i współpracują z otoczeniem za pośrednictwem dobrze określonego interfejsu

JavaBean

- *Bean* – ziarno
- JavaBean – programowy komponent "wielokrotnego użytku", którego właściwości i funkcjonalność mogą być odczytywane i/lub zmieniane uniwersalnymi środkami programistycznymi.
- Klasy-ziarna są typowymi klasami języka Java, które stosują odpowiedni interfejs programistyczny.
- Np. wszystkie komponenty pakietu Swing są ziarnami.

JavaBean (cd.)



Rozróżnienie wynika z kontekstu.

Właściwości i akcesory

- Ziarna posiadają właściwości (atrybuty).
- Dostęp do właściwości zapewniają metody klasy-ziarna nazywane akcesorsami.
- *getter* – akcesor pobierający właściwość
- *setter* – akcesor ustawiający właściwość
- Rodzaje właściwości:
  - proste (właściwości posiadające jedną wartość)
  - indeksowane (właściwości posiadające wiele wartości umieszczonych w tablicy)

Akcesory

- Akcesory dla właściwości prostej niebinarnej:
  - getter:

```
T getNNN( )
```

NNN – nazwa właściwości

T – typ właściwości

  - setter:

```
void setNNN(T)
```

NNN – nazwa właściwości

T – typ właściwości

Akcesory (cd.)

- Akcesory dla właściwości prostej binarnej:
  - getter:

```
boolean isNNN( )
```

NNN – nazwa właściwości

  - setter:

```
void setNNN(boolean)
```

NNN – nazwa właściwości

Akcesory (cd.)

- Akcesory dla właściwości indeksowanej:
  - getter (dla elementu):

```
T getNNN(int)
```

NNN – nazwa właściwości

T – typ elementów właściwości

  - getter (dla tablicy):

```
T[] getNNN()
```

NNN – nazwa właściwości

T – typ elementów właściwości

Akcesory (cd.)

- Akcesory dla właściwości indeksowanej:
  - setter (dla elementu):

```
void setNNN(int, T)
```

NNN – nazwa właściwości

T – typ elementów właściwości

  - setter (dla tablicy):

```
void setNNN(T[])
```

NNN – nazwa właściwości

T – typ elementów właściwości

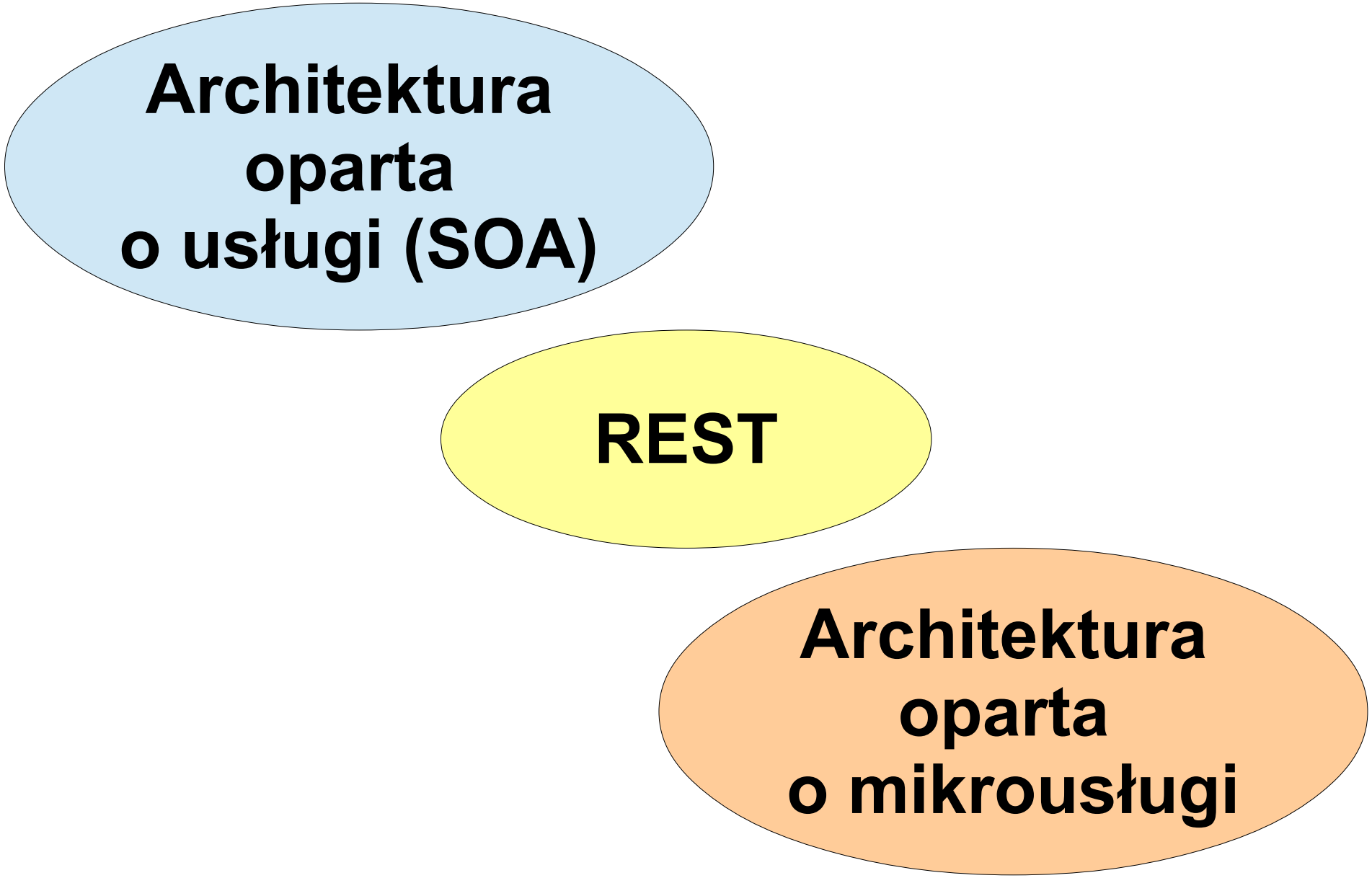
Tworzenie klas-ziaren

- Klasa-ziarno musi spełniać następujące wymagania:
  - klasa stosuje ogólnie przyjęte wzorce sygnatur metod i/lub uzupełniona jest przez dodatkową specjalną klasę opisującą niestandardowe informacje o ziarnie (implementacja interfejsu **BeanInfo**)
  - klasa zapewnia serializację obiektów
  - klasa posiada konstruktor bezargumentowy
  - klasa uwzględnia działania w środowisku wielowątkowym (do obiektu-ziarna może równocześnie odwoływać się wiele wątków)

Cechy technologii obiektów rozproszonych

- Abstrakcyjny język definiowania interfejsów.
- Przeźroczystość.
- Obiektowy charakter oferowanych interfejsów.
- Automatyczne generowanie pni i szkieletów.
- Wykorzystanie brokerów.
- Usługa nazwowa.

Podejścia do tworzenia aplikacji opartych o usługi



SOA

- SOA (ang. *Service Oriented Architecture*) – architektura oparta na usługach.
- SOA – idea budowania aplikacji poprzez udostępnianie usług sieciowych.
- SOA – idea tworzenia luźno powiązanych systemów bazujących na usługach.
- Koncepcja SOA pozwala na tworzenie systemów informatycznych stawiąc nacisk na definiowanie usług, które spełnią wymagania użytkownika.

SOA

- SOA standaryzuję komunikację maszyna – maszyna
- SOA ułatwia tworzenie komponentów i ich składanie do postaci aplikacji.
- SOA służy jako model architektoniczny do tworzenia złożonych aplikacji działających na różnych instancjach wirtualnych.

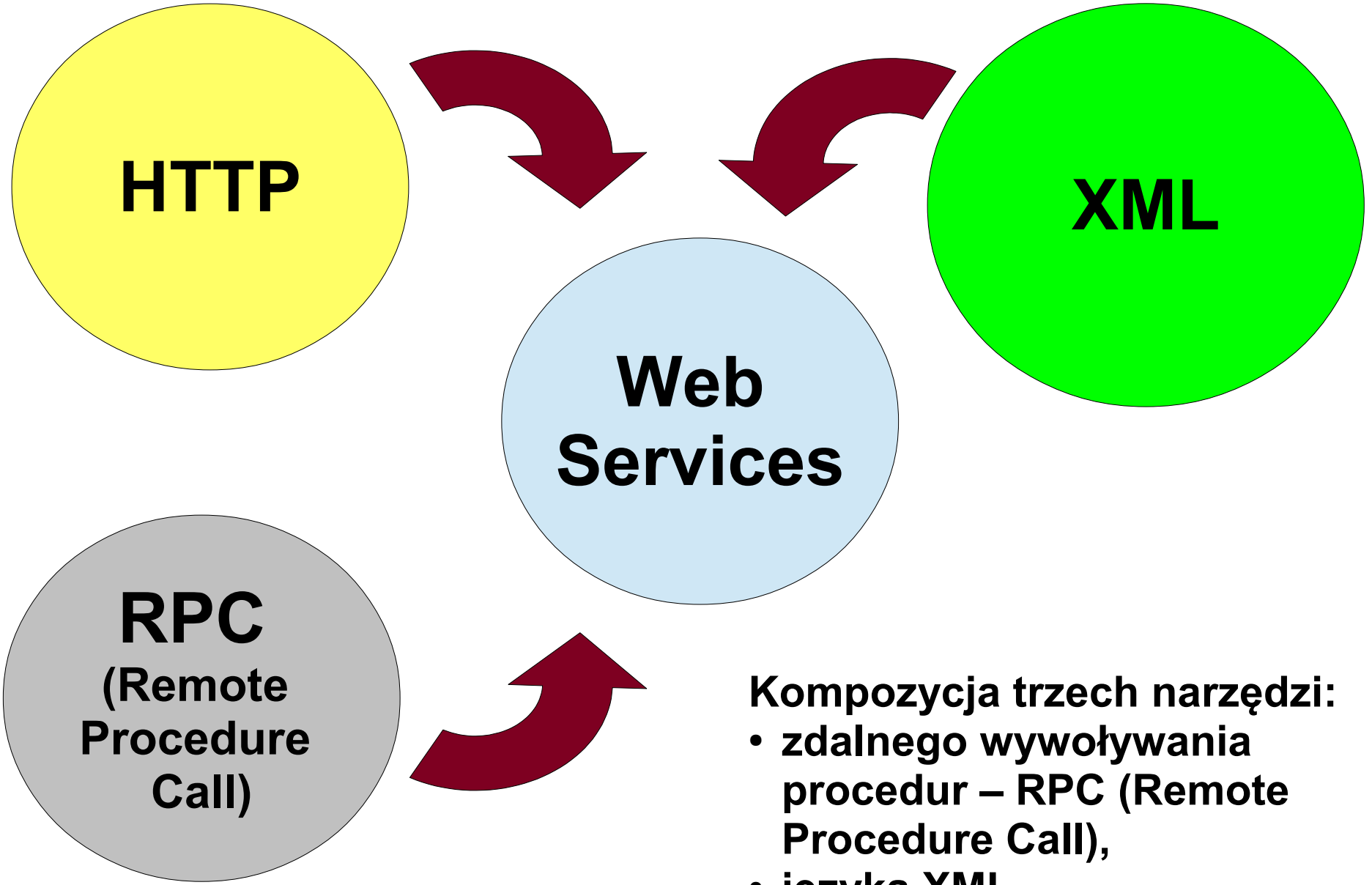
SOA

- SOA wyznacza trzy podstawowe komponenty architektury systemu:
  - **dostarczyciel usług** (utrzymywanie usług, zapewnianie zdalnego dostępu do nich, publikowanie opisu usług w rejestrze),
  - **odbiorca usług** (odszukiwanie i uruchamianie usług),
  - **rejestr usług** (umożliwienie publikacji usług).

Web Services

- Web Services – usługi sieciowe:
  - idea
    - rozproszonych aplikacji webowych,
    - komunikacji pomiędzy aplikacjami w sieci Internet.
  - wywoływane są tak jakby były to lokalne procedury.

Web Services



Kompozycja trzech narzędzi:

- zdalnego wywoływania procedur – RPC (Remote Procedure Call),
- języka XML,
- protokołu HTTP.

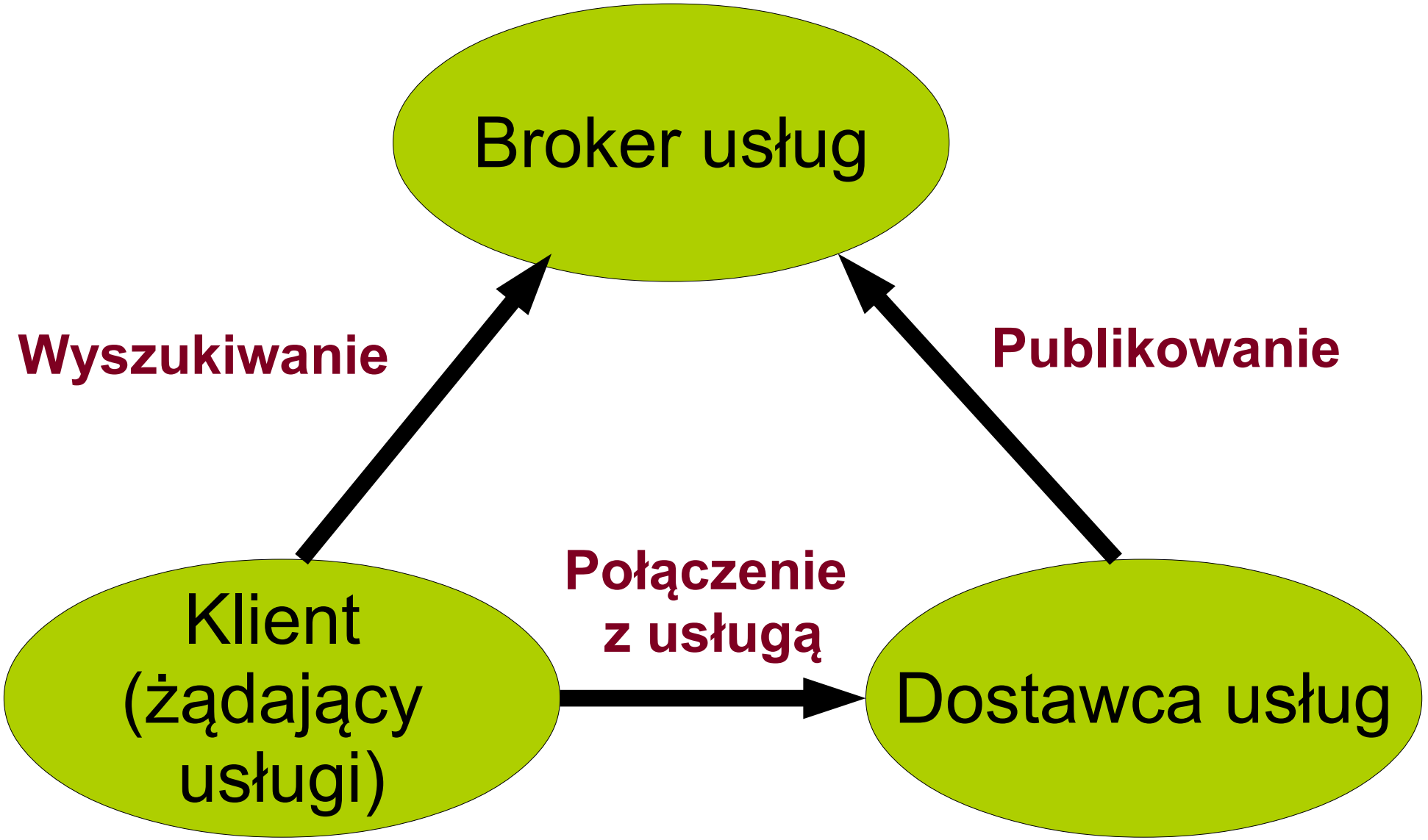
Web Services

- Technologia Web Services stanowi połączenie:
  - oprogramowania pośredniczącego (middleware),
  - aplikacji WWW.
- Technologia Web Services pozwala przezwyciężyć bariery współdziałania rozproszonych aplikacji przez wykorzystanie niezależności od platform (ustandaryzowane protokoły komunikacyjne oraz oparte na tekście formaty zasobów).

Web Services

- Web Service jest dostępnym poprzez sieć komponentem aplikacyjnym przeznaczonym do wykorzystania przez inne aplikacje, zwane aplikacjami klienckimi.
- Web Services - **hermetyzowane** (implementacja nie jest widoczna na zewnątrz), **luźno skojarzone** (modyfikacja danej implementacji nie powoduje problemu propagacji zmiany), **kontraktowane** (opis działania funkcji oraz specyfikacja ich interfejsów jest publicznie dostępna) **funkcje oferowane przez standardowe protokoły**.

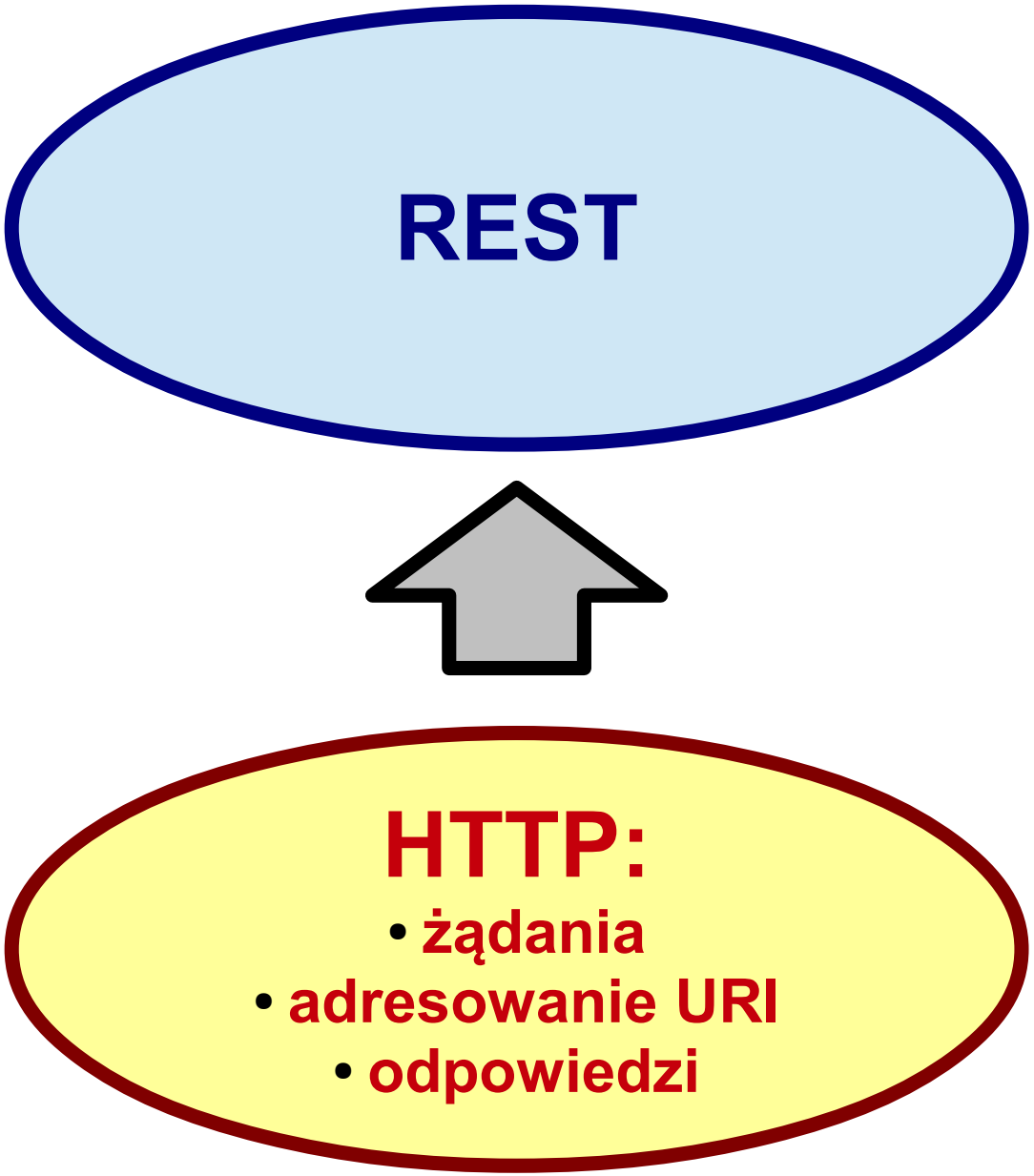
Web Services



REST

- REST (*REpresentational State Transfer*) – transfer stanu reprezentacyjnego
  - podejście architektoniczne określające sposób przesyłania informacji w webowych aplikacjach rozproszonych zaproponowane w roku 2000 przez R.T. Fieldinga.
- REST nie jest protokołem, jest zbiorem zasad dostarczania usług sieciowych (typu RESTful).

REST



RESTful

- RESTful wykorzystuje przede wszystkim cztery metody protokołu HTTP:
  - POST
  - GET
  - PUT
  - DELETE

RESTful

- Podstawowe usługi RESTful wykorzystujące protokół HTTP realizują funkcje odpowiadające tradycyjnym operacjom bazodanowym nazywanym CRUD:
  - **C**reate – utworzenie – **POST**
  - **R**ead – odczyt – **GET**
  - **U**pdate – aktualizacja – **PUT**
  - **D**elete – usunięcie - **DELETE**

RESTful

- Operacje wykonywane są na zasobach w systemie.
- Zasoby identyfikowane są przez adres URI.
- Przesyłane są reprezentacje zasobów, np. w formatach:
  - JSON,
  - XML,
  - HTML.

Projektowanie usług

- Usługi sieciowe (Web services) odpowiedzialne są za komunikację online M2M (*machine-to-machine*) w aplikacjach rozproszonych przy wykorzystaniu sieci Internet.
- SOAP oraz REST umożliwiają tworzenie własnego API (*Application Programming Interface*).

Projektowanie usług

SOAP:

- ustandaryzowany protokół definiujący ścisłe reguły używania

- głównie technologie oparte o C++, Java, .NET

- ***function-driven*** – wykonywanie różnych operacji (funkcji, procedur, metod)

- format XML

REST:

- zbiór zasad

- potrzebuje mniej zasobów

- jest bardziej eleastyczny

- ***data-driven*** – operacje tworzenia, odczytu, aktualizowania i usuwania zasobów

- formaty XML, JSON, HTML i inne

Mikrousługi

- **Architektura oparta o mikrousługi** (**mikroserwisy**) – kolejny etap w sposobie wytwarzania oprogramowania nawiązujący do architektury opartej o usługi (SOA).
- Termin „Micro-Web-Services” pojawił się w 2005 roku.
- Koncepcja mikrousług została zaprezentowana na konferencji GeeCon 2013.

Mikrousługi

- **Architektura oparta o mikrousługi** (**mikroserwisy**) – sposób projektowania aplikacji jako autonomicznych usług, które mogą być:
  - implementowane,
  - testowane,
  - wersjonowane,
  - wdrażane

  niezależnie.

Mikrousługi

- Idea mikrousług oparta jest na filozofii systemu Unix:

„Do one thing and do it well”

- Najważniejsze prawo przy budowaniu aplikacji składającej się z mikrousług:

„*High in cohesion, low in coupling*”

(mikrousługi wysoce spójne, luźne powiązanie pomiędzy mikrousługami)

Mikrousługi

- Mikrousługi stanowią elementy składowe aplikacji rozproszonych.
- Aby efektywnie uruchomić aplikację rozproszoną można wykorzystać mikrousługi dostępne jako rozwiązania w chmurze.