35. Model i technologie infrastruktury komunikacyjnej usług webowych

# Architektury zorientowane na usługi - SOA (Service Oriented Architecture)

**SOA:**

* Następca architektury komponentowej (ten sam cel - ponowne wykorzystanie oprogramowania) - logika biznesowa nie stanowi monolitycznego programu, lecz jest rozbita pomiędzy wiele rozproszonych komponentów usługowych (usług) koordynowanych przez centralną aplikację sterującą. Łączność pomiędzy aplikacją sterującą a komponentami usługowymi odbywa się za pośrednictwem sieci Internet.
* Forma architektury systemu informatycznego, w którym głównym elementem są usługi, komunikujące się między sobą za pomocą dobrze zdefiniowanych interfejsów.
* Zbiór praktyk, zasad i metodyk na temat tworzenia architektury składającej się ze współdziałających usług.

**Usługi:**

* Autonomiczne, wieloużywalne części oprogramowania, wykonujące dobrze określone funkcje/zadania.
* Mogą być rozproszone w różnych lokalizacjach i działać na różnych platformach
* Ich interfejsy są definiowane w sposób niezależny od platformy
* Są luźno związane z aplikacją sterującą, nazywaną też konsumentem usług (**Service Consumer**) i mogą być współdzielone przez wiele aplikacji.
* Zwykle są implementowane i udostępniane przez niezależne podmioty, nazywane dostawcami usług (**Service Providers**).

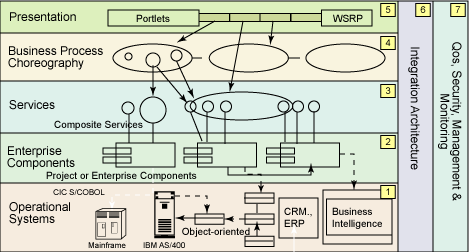
**Zasady SOA**

* **Autonomia usług** – usługi są niezależne od innych usług – niedziałanie innych usług na nie nie wpływa. Usługi mogą być niezależnie konstruowane, rozwijane, udostępniane i zarządzane.
* **Abstrakcja** (Wyraźnie wyznaczone granice) – konsumentów usług interesuje jak się skomunikować z usługą, a nie jak ona działa.
* **Kontrakty pomiędzy usługami** – służą do wymiany informacji.
* **Powtórne wykorzystanie** – cecha architektury komponentowej.
* **Odkrywalność** – istnieją mechanizmy wyszukujące usługi o zadanych kryteriach

**Możliwe realizacje SOA**

|  |  |
| --- | --- |
| Wersja podstawowa  Opis usługi znajduje się w dokumencie WSDL. Złożenie, że klient (konsument usługi) wie czego chce – musi mieć dane o usłudze aby się z nią skomunikować |  |
| Rejestr  W rejestrze znajdują się metadane o usługach – opis usługi, techniczne szczegóły komunikacji, gotowość do użycia, powiązania z innymi usługami. Pozwala to na szybsze odnajdywanie przez potrzebnych usług. Rejestr pełni jednak rolę **pasywną**. Klient sam musi wyszukać usługę, która go interesuje. |  |
| Broker  Między rejestrem a usługami umieszcza się obiekt pośredniczący – Broker, który wyszukuje żądaną usługę i przesyła klientowi. Ścisłe, punktowe powiązanie między usługami. Z każdą nową usługą złożoność rozwiązania rośnie. |  |
| Szyna  ESB – Enterprise Service Bus. Najbardziej złożone rozwiązanie. Architektura ESB zakłada model centralnej szyny danych, jako model połączeń między usługami. Do szyny ESB podłączane są wszystkie usługi, Broker oraz Rejestr. Szyna pośredniczy w komunikacji pomiędzy usługami. Zaletą tego rozwiązania w stosunku do poprzedniego jest mała ilość połączeń między usługami (brak bezpośrednich połączeń). W przypadku n usług, liczba połączeń z wykorzystaniem ESB wynosi n, a w przypadku połączeń bezpośrednich n(n−1)/2. |  |

**Architektura SOA**



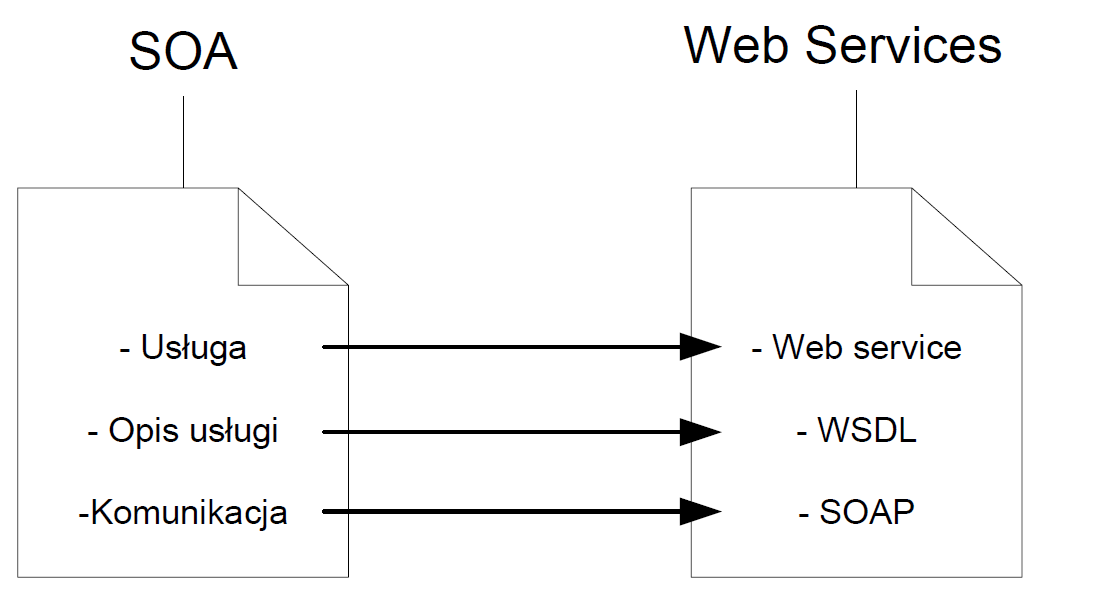
Architektura SOA jest warstwowa (wzorzec architektoniczny Layers). Warstwy poziome (1-5) stanowią warstwy funkcjonalne, a warstwy pionowe (6-7) służą do zarządzania bezpieczeństwem i jakością usług. Warstwy poziome 1-2 stanowią środowisko wykonawcze. Warstwa 2 stanowi wsparcie logiki biznesowej (3) (np. dodatkowe przetwarzanie danych) a warstwa 1 stanowi dostęp do danych. Warstwa środkowa (3) to zbiór autonomicznych usług stanowiących logikę biznesową. W warstwie 4 odbywa się procedura łączenia usług w formę procesu biznesowego nazywana aranżacją (orchestration). Do budowania logiki procesu używane są specjalistyczne języki, takie jak BPEL (Business Process Execution Language). Warstwa 5, to warstwa prezentacji, którą stanowi np. aplikacja kliencka.

# Web Services

Do implementacji rozproszonych komponentów usługowych wykorzystuje się technologie CORBA, DCOM, EJB i Web Services, przy czym właśnie ta ostatnia jest najbardziej znana.

**Web Services:**

* Technologia konstrukcji rozproszonych komponentów usługowych (usług), stanowiących podstawę dla realizacji aplikacji biznesowych w architekturze zorientowanej na usługi (SOA)
* Usługi Web Services umożliwiają aplikacjom wymianę danych.
* Usługi Web Services są dyskretnymi jednostkami kodu; każda z nich odpowiada za ograniczony zbiór zadań.
* Są oparte na XML, uniwersalnym języku internetowej wymiany danych; można je wywoływać niezależnie od systemu operacyjnego i języka programowania.
* Implementacja środowiska aplikacyjnego w technologii Web Services jest możliwa w dowolnym z języków programowania



* Najważniejsze elementy:
  + Web Service: Usługa Web Service to zwarty, samodokumentujący się komponent programowy, który może być przez swojego twórcę zarejestrowany w sieci komputerowej, a następnie przez twórcę aplikacji-konsumenta odkryty i wywołany w trybie zdalnego wykonania.
  + SOAP – protokół komunikacyjny służący do przekazywania zdalnych wywołań
  + WSDL – język opisu interfejsu usługi służący do dystrybucji parametrów połączeń sieciowych
  + UDDI – specyfikacja bazy danych służącej do rejestracji udostępnianych komponentów usługowych
* Podstawowe kanony implementacji środowisk Web Services obejmują:
  1. Implementację komponentu usługowego w dowolnym języku programowania,
  2. Sporządzenie pliku opisu interfejsu komponentu (WSDL),
  3. Zgłoszenie komponentu do centralnego rejestru (UDDI),
  4. Wyszukanie komponentu w centralnym rejestrze (UDDI),
  5. Wygenerowanie kodu komunikacyjnego dla klienta (Web Service Proxy),
  6. Implementację klienta Web Service.

**WCF (Windows Communication Fundation):**

* Technologia Microsoft umożliwiająca implementację zaawansowanych usług sieciowych. Stworzona, aby wspomóc programowanie aplikacji rozproszonych oraz ułatwić implementację systemów zaprojektowanych w architekturze SOA.
* Zestaw klas do tworzenia usług sieciowych
* Cechy (wspiera):
* Transakcyjność
* Hosting – proces instalacji usługi w środowisku docelowym. Wyróżnia się następujące techniki hostowania:
  + Samohostowanie – procesem hostującym jest sama aplikacja. Cyklem życia usługi zarządza programista
  + Hosotwanie przez usługę Windows – cykl życia usługi kontrolowany przez SO.
  + Hostowanie w IIS – wspiera automatyczną aktywację usługi (podczas pierwszego żądania klienta), zarządzana przez IIS, możliwość wygodnej konfiguracji
* Bezpieczeństwo – poufność (szyfrowanie wiadomości), autentykacja i autoryzacja, integralność
* Kolejkowanie (np. MSMQ)

**Web Service Software Factory (WSSF)**

* Technologia do modelowania usług webowych. Umożliwia szybkie napisanie „zgrabnego” web service’u, który może być później wykorzystany w dowolnej aplikacji Windows.
* Generuje kod i wymusza na nas dość „elegancką” strukturę usługi
* Modelowanie kontraktu usługi, kontraktu danych i hosta

# SOAP

**SOAP (Simple Object Access Protocol):**

* Jest podstawowym protokołem wymiany komunikatów w architekturze SOA.
* **Tworzy podstawową infrastrukturę komunikacyjną dla usług webowych.**
* Prosty protokół komunikacyjny oparty na języku XML, umożliwiający przekazywanie wywołań zdalnych komponentów Web Services
* Może współdziałać z dowolnym niskopoziomowym sieciowym mechanizmem transportowym, np. HTTP, HTTPS, SMTP, JMS, RMI (zwykle HTTP).
* Niezależny od platformy
* Używa gramatyki XML do:
  + Określania nazw metod
  + Definiowania typów parametrów i zwracanych wartości
  + Opisu typów
  + Określa sposób wysyłania/odbierania komunikatów/odpowiedzi oraz kodowania
* Jest standardem W3C.
* Cechy:
  + Rozszerzalność,
  + Możliwość zastosowania wielu różnych protokołów sieciowych (HTTP, SMTP lub MSMQ)
  + Niezależność od zastosowanego modelu programistycznego

**Komunikaty SOAP**

* Podstawowa jednostka informacji przesyłana pomiędzy węzłami SOAP
* Pakiet danych przesyłanych od źródła do celu
* Podstawowa jednostka komunikacji między węzłami SOAP

**Infrastruktura komunikacyjna (=Struktura komunikatu SOAP)**

* Istotna część specyfikacji SOAP poświęcona jest infrastrukturze komunikacyjnej. Infrastruktura komunikacyjna SOAP definiuje zestaw elementów XML do „opakowywania” dowolnych komunikatów XML w celu umożliwienia przekazywania ich pomiędzy systemami. Infrastruktura składa się z następujących podstawowych elementów XML:
  + **Envelope (koperta)** – określa szkielet opisujący, co znajduje się w komunikacie i jak go przetwarzać
  + **Header (nagłówek)** – zawiera informacje precyzujące sposób traktowania wiadomości. Takie jak: uwierzytelnianie, streszczenie wiadomości, informacje o routingu – dotyczy to np.: miejsc przeznaczenia wiadomości oraz kolejności w jakiej musi ona tam dotrzeć, transakcje – konieczność wykonania pewnych czynności w ramach transakcji nadawcy, informacje o odpłatności.
  + **Body (treść)** - reprezentuje właściwą treść komunikatu; to pojemnik, który może zawierać dowolną liczbę elementów (danych) dowolnej przestrzeni nazw, które chcemy przesłać w komunikacie. Zawiera dane przeznaczone dla ostatecznego odbiorcy.
  + **Fault (błąd)** - umieszczony wewnątrz elementu Body — może informować o błędach

**Węzeł SOAP**

* Jednostka, która implementuje reguły przetwarzania komunikatów SOAP. Reguły dotyczą nadawania, odbierania, przetwarzania i przekazywania komunikatów.
* Węzeł początkowy – inicjuje przesyłanie komunikatu SOAP
* Węzeł końcowy – przyjmuje i przetwarza zawartość komunikatu
* Pośrednik – dowolny węzeł między początkowym a końcowym. W przypadku błędu pośrednik przekazuje komunikat do węzła początkowego.
* Ścieżka komunikatu – zestaw węzłów pośrednich, przez które przechodzi komunikat.

**Model przetwarzania SOAP:**

* SOAP definiuje model przetwarzania, który opisuje reguły przetwarzania komunikatów SOAP przesyłanych od nadawcy SOAP do odbiorcy SOAP.
* Wiadomość SOAP (SOAP Message) jest przesyłana od węzła początkowego nadawcy (SOAP Sender) poprzez węzły pośredniczące (SOAP Node) do węzła końcowego odbiorcy (SOAP Receiver), przy użyciu dowiązanego protokołu transportowego (SOAP Binding), z ewentualnym wykorzystaniem dodatkowych rozszerzeń (SOAP Feature) opisanych za pomocą bloków wewnątrz wiadomości (SOAP Module).
* **Wzorce przesyłania wiadomości:**
  + Komunikacja jednokierunkowa (Simplex) – proste jednokierunkowe przekazanie komunikatu, gdzie nadawca nie otrzymuje odpowiedzi (jeden tylko wysyła, a drugi tylko odbiera).
  + Komunikacja dwukierunkowa (Duplex) – oba mogą wysyłać i odbierać
  + Żądanie-Odpowiedź (Request-Reply) – jednokierunkowa, podobna do wywoływania operacji – może być asynchroniczna lub synchroniczna.
* **Kanał komunikacyjny** – medium służące do wymiany komunikatów**.** Scenariusz:
  + Klient ustanawia kanał do usługi
  + Usługa akceptuje żądanie otwarcia kanału od klienta
  + Klient wysyła wiadomość przez kanał
  + Usługa wysyła odpowiedź przez kanał
  + Klient zamyka kanał i zwalnia zasoby
* Podczas przetwarzania komunikatu węzeł SOAP pełni względem niego co najmniej jedną rolę, która ma wpływ na sposób przetwarzania nagłówków SOAP. Gdy węzeł SOAP otrzymuje komunikat do przetworzenia, najpierw musi ustalić, jaką rolę ma wobec niego pełnić. Decyzję tę może podjąć na podstawie informacji znajdujących się w komunikacie SOAP. Role (SOAP 1.2):
  + **next** ­ Każdy węzeł pośredni SOAP oraz końcowy odbiorca SOAP musi przyjmować tę rolę i może dodatkowo przyjmować inne role SOAP
  + **none** ­ Węzły SOAP nie mogą przyjmować tej roli
  + **ultimateReceiver** ­ Węzeł SOAP musi przyjąć tę rolę aby zostać końcowym odbiorcą SOAP. Węzły pośrednie SOAP nie mogą przyjmować tej roli
* Gdy węzeł określi już, jakie role będzie pełnić, musi przetworzyć wszystkie obowiązkowe nagłówki (oznaczone wartością **mustUnderstand="1"**), skierowane do tych ról. Węzeł SOAP może także przetworzyć nagłówki opcjonalne (oznaczone wartością **mustUnderstand="0"**), kierowane do ról pełnionych przez węzeł.

# Web Services Description Language (WSDL)

* Opracowany przez Microsoft i IBM, oparty na XML język do definiowania usług sieciowych. Służy do opisu technicznych parametrów połączenia sieciowego aplikacji-klienta z komponentem Web Service.
* Opisy WSDL mogą być umieszczane w rejestrze UDDI – kombinacja WSDL i UDDI ma się przyczynić do promocji rozwiązań usług sieciowych w Internecie.
* **WSDL jest wykorzystywany do stworzenia opisu usługi oraz tego jak się z nią skomunikować - definicja tzw. punktów końcowych usługi - opis punktów dostępu do usług sieciowych.**
* WSDL to język znaczników XML: Znacznik <definitions> otacza całą zawartość dokumentu. Znaczniki <service> wraz ze znacznikami <port> definiują adresy punktów dostępowych dla usługi. Znaczniki <portType> służą do deklaracji funkcji biznesowych oferowanych przez usługę. Znaczniki <binding> określają metody kodowania parametrów wywołania i parametrów zwrotnych usługi.

# Punkty końcowe

* Receptory, które odbierają żądania z zewnątrz, a następnie je przetwarzają je na język specyficzny dla technologii.
* Opisywane przez 3 elementy (ABC):
  + **A (dress)** - **[DOKĄD?] - Adres** 
    - Każdy punkt końcowy usługi ma unikalny adres. Adres wskazuje na: Lokalizację oraz Schemat transportu (HTTP/HTTPS, TCP, IPC, Peer network, MSMQ).
  + **B (binding)** - **[JAK?] – Wiązanie**
    - Mechanizm opisujący szczegóły komunikacyjne na danym punkcie końcowym. Dotyczą one protokółu transportowego (zabezpieczenia, transakcje) oraz sposobu kodowania i kompresji wiadomości
    - Dla danego protokołu definiuje, w jaki sposób komunikaty SOAP są przesyłane za pomocą tego protokołu.
    - Pozwala oddzielić instalację usługi w danym środowisku od implementacji usługi
    - Usługa publikuje decyzje związane z wyborem wiązań w metadanych
  + **C (ontract)** - **[CO?]**
    - zbiór specyfikacji, definiujących interfejs usługi. Musi być udostępniony przez każdą usługę.
    - Typu kontraktów:
      1. **Kontrakt usługi [ServiceContract]** (standardy: CLR – Common Language Runtime, WSDL)
* Zawiera informacje dla klienta o tym co usługa oferuje
* Może być określony dla klasy (której instancje będą usługami) lub dla interfejsu
* Usługa musi mieć co najmniej jeden taki kontrakt
* Możliwe parametry np. Kontrakt zwrotny, Poziom ochrony (Sign, Nne, EncryptAndSign), Tryb sesji (Allowed, Not Allowed)
  + - 1. **Kontrakt operacji – [OperationContract]** (standardy: CLR, XSD – XML Schema Definition)
         * Określa, które metody usługi będą udostępnione
         * Możliwe parametry np. Poziom ochrony, IsTerminating/IsInitiating (zaczyna kończy sesję)
      2. **Kontrakt danych – [DataContract]**
         * Opisuje typy danych podlegające wymianie między usługą i klientem
         * Stosowany do klas, typów wyliczeniowych, struktur
         * Określa co ma być (de)serializowane
         * Każdy atrybut kontraktu musi być jawnie opisany z wykorzystaniem atrybutu [DataMember]
      3. **Kontrakt wiadomości – [MessageContract]** (standardy: CLR, SOAP)
         * Dostarcza dodatkowych możliwości w stosunku do kontraktu danych w zakresie formatowania parametrów wiadomości SOAP
         * Atrybuty [MessageHeader] [MessageBodyMember]
         * Możliwe parametry np. Poziom ochrony, IsWrapped, WrapperName, WrapperNamespace
      4. **Kontrakt błędów – [FaultContract(typeOf(…))]**
         * Mechanizm obsługi wyjątków w środowisku sieciowym
         * Definiowany w kontekście kontraktu operacji – do opisu jakiego typu wyjątki może ona generować
* **Zachowania** – określają charakterystykę zachowania punktu końcowego lub całej usługi w czasie działania (np. Transactions, Concurrency: Multiple, Single, Reentrant, Error handling, Instancing: PerCall, PerSession, Single.