

# XML w elektronicznej wymianie danych i integracji aplikacji

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią –  
2007/08

## XML w integracji aplikacji

- ▶ Cel: umożliwienie wymiany danych pomiędzy aplikacjami:
  - ▶ aplikacje/komponenty/moduły posługują się różnymi formatami wewnętrznymi,
  - ▶ wspólny mianownik: XML.
- ▶ Zastosowania:
  - ▶ komunikacja między klientem a serwerem,
  - ▶ komunikacja między elementami systemu rozproszonego,
  - ▶ integracja komponentów aplikacji,
  - ▶ konfigurowanie aplikacji i jej komponentów,
  - ▶ ...

## Zastosowania

- ▶ Wewnętrz-firmowe:
  - ▶ w ramach jednego projektu / jednej instytucji,
  - ▶ komunikacja między różnymi modułami / aplikacjami,
  - ▶ być może w architekturze rozproszonej,
  - ▶ rozwiązania tworzone pod kątem projektu,
  - ▶ możliwość stosowania rozwiązań standardowych (np. JAXB).
- ▶ Globalne:
  - ▶ usługi dostępne w Internecie,
  - ▶ współpraca różnych partnerów,
  - ▶ ważna standaryzacja,
  - ▶ typowe rozwiązanie – Web services.

## Web services

- ▶ Idea – „witryna WWW czytana przez maszyny”.
- ▶ Pojęcie używane także do opisu architektury systemu złożonego z takich komponentów.
- ▶ Zazwyczaj dotyczy systemów o poniższych cechach (w nawiasach typowe/standardowe technologie):
  - ▶ komunikacja w wysokiej warstwie sieci (HTTP),
  - ▶ węzły/usługi opisane (WSDL),
  - ▶ komunikaty między węzłami (XML, SOAP),
  - ▶ możliwość rejestrowania i wyszukiwania usług (UDDI).

## Web services – możliwe zastosowania

- ▶ Udostępnianie/sprzedaż użytecznych danych:
  - ▶ rozkłady lotów linii lotniczych,
  - ▶ dane o pogodzie,
  - ▶ aktualny czas wg zegara atomowego.
- ▶ Zdalne usługi:
  - ▶ wyszukiwanie,
  - ▶ pobieranie aktualnej wersji oprogramowania.
- ▶ Operacje biznesowe między partnerami:
  - ▶ zamawianie,
  - ▶ sprawdzanie stanu realizacji zamówienia.

## Web services – standaryzacja

- ▶ SOAP (pierwotnie *Simple Object Access Protocol*):
  - ▶ początki: 1998,
  - ▶ wersja 1.2: rekomendacje W3C, czerwiec 2003 (errata: kwiecień 2007).
- ▶ Web Services Description Language:
  - ▶ wersja 1.1: notatka W3C, 2001,
  - ▶ wersja 2.0: rekomendacje W3C, czerwiec 2007.
- ▶ Universal Description Discovery and Integration:
  - ▶ projekt organizacji OASIS,
  - ▶ część standardu WS-I Basic Profile.
- ▶ Standardy WS-\*:
  - ▶ różne standardy, zazwyczaj nie rekomendacje W3C, m.in.:
  - ▶ Web Services Interoperability – ramy dla działania Web serwisów:  
WS-I Basic Profile, Simple Soap Binding Profile, . . . ,
  - ▶ WS-Eventing, WS-Addressing, WS-Routing, . . . – standardy IBM.
- ▶ Business Process Execution Language (OASIS) – opis semantyki i łączenie serwisów w super-serwisy.

# SOAP – protokół komunikacji

- ▶ Protokół komunikacji.
- ▶ Format komunikatów (zastosowanie XML).
- ▶ Różnice w stosunku do RPC, CORBA, DCOM itp.:
  - ▶ reprezentacja danych niezależna od platformy (tekstowa),
  - ▶ typy danych niezależne od platformy,
  - ▶ niższa efektywność.

## Komunikat SOAP

- ▶ Dokument XML, jedna wiadomość:
  - ▶ przestrzeń nazw  
`http://www.w3.org/2001/12/soap-envelope`,
  - ▶ element główny: `Envelope`.
- ▶ Składniki dokumentu (elementy pod głównym):
  - ▶ `header` – opcjonalny,
  - ▶ `body` – obowiązkowy.
- ▶ Ograniczenia:
  - ▶ brak DTD (i referencji do encji),
  - ▶ brak instrukcji przetwarzania.

# Nagłówek SOAP

- ▶ actor – opcjonalny identyfikator strony komunikacji (URI),
- ▶ mustUnderstand – czy zrozumienie jest konieczne (0/1).

## Przykład z W3Schools

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Header>
    <m:Trans
xmlns:m="http://www.w3schools.com/transaction/"
soap:actor="http://www.w3schools.com/appml/"
soap:mustUnderstand="1">234</m:Trans>
  </soap:Header>

  ...
</soap:Envelope>
```

# Ciało komunikatu SOAP

- ▶ encodingStyle – sposób kodowania danych (URI),

## Zapytanie – przykład z W3Schools

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
    </m:GetPrice>
  </soap:Body>

</soap:Envelope>
```

# Ciało komunikatu SOAP

## Odpowiedź – przykład z W3Schools

```
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

## Informacja zwrotna o błędzie

- ▶ Element(y) fault w body.
- ▶ Podelementy:
  - ▶ faultcode
  - ▶ faultstring
  - ▶ faultactor
  - ▶ detail
- ▶ Standardowe kody błędów:
  - ▶ VersionMismatch
  - ▶ MustUnderstand
  - ▶ DataEncodingUnknown
  - ▶ Sender
  - ▶ Receiver

# WSDL – opis serwisu

- ▶ Dokument XML, opis jednej lub kilku usług:
  - ▶ przestrzeń nazw  
`http://schemas.xmlsoap.org/wsdl/`,
  - ▶ element główny: `definitions`.
- ▶ Składniki dokumentu (elementy pod głównym):
  - ▶ `serviceType` – opis usługi, zawiera wiele `portType`,
  - ▶ `portType` – typ „portu”, czyli zbiór operacji, usługa może posiadać wiele portów, `portType` może występować w `serviceType` oraz bezpośrednio w `definitions`,
  - ▶ `message` – opis typu komunikatu,
  - ▶ `types` – definicje typów,
  - ▶ `binding` – opis sposobu komunikacji.
- ▶ XML Schema używane do definiowania typów.

## WSDL – wiadomości, typ portu

### Przykład z W3Schools

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

## WSDL – wiązanie (z SOAP)

- ▶ `style` – `rpc` lub `document`,
- ▶ `transport` – protokół transportowy (URI).
- ▶ `soapAction` – akcja SOAP odpowiadająca operacji.

### Przykład z W3Schools

```
<binding type="glossaryTerms" name="b1">
<soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
      soapAction="http://example.com/getTerm"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>
```

## Wyszukiwanie usług

- ▶ Idea:
  - ▶ dostawcy usług rejestrują je w katalogu,
  - ▶ użytkownik znajduje usługę w katalogu,
  - ▶ najlepiej: opis usługi i wyszukiwanie semantyczne.
- ▶ Universal Description Discovery and Integration (UDDI):
  - ▶ usługa katalogowa, wiele powiązanych węzłów,
  - ▶ dostępna jako Web service (poprzez SOAP),
  - ▶ opisy usług w WSDL.

# Super usługi i kooperacja usług

- ▶ Terminologia:
  - ▶ **choreografia** (*choreography*) – opis działania układu usług jako całości,
  - ▶ **orkiestracja** (*orchestration*) – opis działania pojedynczego komponentu i jego kooperacji ze światem,
- ▶ Standaryzacja:
  - ▶ WS-Choreography (W3C),
  - ▶ Business Process Execution Language (BPEL, OASIS) – opis orkiestracji pojedynczej usługi,
- ▶ (Nowe) usługi zbudowane w oparciu o istniejące usługi:
  - ▶ możliwość rekursji,
  - ▶ *Service Oriented Architecture*.

## AJAX

- ▶ AJAX - **A**synchronous **J**avaScript and **X**ML:
  - ▶ strona WWW „działa” w przeglądarce użytkownika dzięki JavaScript,
  - ▶ komunikacja z serwerem po HTTP,
  - ▶ komunikaty kodowane w XML,
  - ▶ coraz szerzej stosowane (m.in. wiele usług Google).
- ▶ Zalety:
  - ▶ interakcja z użytkownikiem bez przeładowywania strony,
  - ▶ przesyłane tylko niezbędne dane.
- ▶ Wady:
  - ▶ narzut na przesłanie i uruchomienie skryptu,
  - ▶ (czasami) nieintuicyjne zachowanie stron WWW.
- ▶ Odstępstwa w praktyce:
  - ▶ inne „scripty”,
  - ▶ dane nie zawsze w XML.

## AJAX – wsparcie

- ▶ Wsparcie do używania AJAX-a:
  - ▶ pisanie kodu JavaScript uciążliwe,
  - ▶ problemy z obsługą różnych przeglądarek,
  - ▶ rozwiązanie: gotowe fragmenty kodu generowane przez narzędzia.
- ▶ Przykłady bibliotek:
  - ▶ DOJO,
  - ▶ DWR,
  - ▶ AJAX Tags (biblioteka znaczników JSP).

## Elektroniczna wymiana danych (EDI)

- ▶ Przedsiębiorstwa używają własnych formatów danych.
- ▶ Wymiana danych między przedsiębiorstwami:
  - ▶ dostawca (sprze)daje klientowi narzędzia,
  - ▶ partner dostosowuje się do formatu silniejszego partnera,
  - ▶ tworzone ad-hoc narzędzia tłumaczące,
  - ▶ standard.
- ▶ Możliwości wykorzystania standardu:
  - ▶ narzędzia tworzone zgodnie ze standardem,
  - ▶ narzędzia tłumaczące do standardu w interfejsach.

## Standaryzacja EDI przed XML

- ▶ EDIFACT – United Nations Standard Messages Directory for Electronic Data Interchange For Administration, Commerce and Transport.
- ▶ ANSI Accredited Standards Committee X12 sub-group.

### EDIFACT – fragment komunikatu

```
TDT+20+57EP12+1++HLC:172:20+++8407319:146::HAMBURG EXPRESS:DE'  
RFF+VM:DIHE'  
RFF+VON:51WP11'  
FTX+TRA+++CEE-CHINA EUROPE EXPRESS LOOP 4'  
FTX+AAI+++MASTER PETER SUNSHINE'
```

## Standaryzacja EDI przed XML

### EDIFACT – fragment specyfikacji

```
1 Inland transport  
2 At the statistical territory limit  
10 Pre-carriage transport  
11 At border  
12 At departure  
13 At destination  
20 Main-carriage transport  
21 Main carriage - first carrier  
22 Main carriage - second carrier  
23 Main carriage - third carrier  
30 On-carriage transport
```

# XML EDI

## Tradycyjne EDI

- ▶ Format dokumentów zapisany w specyfikacji
- ▶ Zwięzłe komunikaty, tylko niezbędne dane.
- ▶ Scentralizowana, trudna zmiana standardu.
- ▶ Zmiany standardu pociągają uciążliwe zmiany oprogramowania.
- ▶ Implementowanie od podstaw.

## XML EDI

- ▶ „Samoopisujący się” format dokumentów.
- ▶ Rozwlekłe komunikaty – narzut na „samoopisywanie się”.
- ▶ Możliwość tworzenia własnych odmian standardów.
- ▶ Większość problemów ze zmianą standardu bierze na siebie parser XML.
- ▶ Możliwość korzystania z gotowych narzędzi.

## Elastyczność XML EDI

- ▶ Elastyczny format komunikatów (elementy opcjonalne, wybór podelementu).
- ▶ XSLT jako język opisu formatowania.
- ▶ Nowe zastosowanie EDI:
  - ▶ podstawowa funkcjonalność – wymiana danych między aplikacjami przedsiębiorstw,
  - ▶ nowe perspektywy: kontakt z klientami wyposażonymi tylko w przeglądarki,
  - ▶ E-Commerce.
- ▶ Zgodność z ideą Web services.

# Standaryzacja XML EDI

- ▶ XML zbyt elastyczny.
- ▶ Standaryzacja na poziomie ramowym:
  - ▶ możliwa wymiana informacji dowolnego typu,
  - ▶ informacje jednego typu tak samo reprezentowane,
  - ▶ przykład: Electronic Business XML – ebXML.
- ▶ Standardy branżowe (konkretne zestawy komunikatów):
  - ▶ SWIFT,
  - ▶ RosettaNet,
  - ▶ Automotive Industry Action Group,
  - ▶ Health Level Seven,
  - ▶ Open Travel Alliance,
  - ▶ ...

## ebXML

- ▶ ebXML:
  - ▶ zbiór specyfikacji definiujących sposób prowadzenia biznesu i wymiany danych przez Internet,
  - ▶ zaakceptowane 14 maja 2001 r.,
  - ▶ oczekiwane implementacje i wsparcie w istniejących systemach,
  - ▶ wsparcie przez inne inicjatywy standaryzacyjne.
- ▶ Electronic Business XML Working Group:
  - ▶ założona we wrześniu 1999 r.,
  - ▶ ok. 150 specjalistów,
  - ▶ patronat OASIS i UN/CEFACT.

## ebXML – podejście do standaryzacji

- ▶ Meta-model pozwalający na opracowywanie modeli specyficznych dla zastosowań:
  - ▶ zbiór podstawowych schematów, elementów XML oraz procesów biznesowych,
  - ▶ sposób definiowania słowników danych,
  - ▶ nie definiuje konkretnych, docelowych komunikatów.
- ▶ Metainformacje:
  - ▶ informacje o wersjach,
  - ▶ metadane odpowiadające nagłówkom z istniejących systemów EDI.
- ▶ Ramy architektury technicznej:
  - ▶ sposoby implementacji repozytoriów, serwisów, itp.,
  - ▶ integracja z istniejącymi technologiami EDI.