Web Services – Wprowadzenie

Andrzej Chwedoruk

(Referat przygotowany na seminarium magisterskie z obiektowości)

Kwiecień 2005

Czym są webservices?

Webservices to nowa jakość o której mówi się, że zrewolucjonizuje sposoby dystrybucji danych w sieci Internet. Głównym założeniem platform webservices jest nie tylko udostępnianie informacji ale generalnie usług.

Co to jest WebServices ?

Pojęcie "Web Service" *w ogólności* oznacza ideę rozproszonej aplikacji webowej.

Myślimy o komunikacji między aplikacjami poprzez internet. Słyszymy i czytamy w wielu miejscach:

"usługa brana z sieci".

"zorientowane na usługi podejście projektanckie bazujące na idei budowania aplikacji poprzez odkrywanie i udostępnianie serwisów sieciowych oraz integracji aplikacji na zasadzie just-in-time."

Jest to bardzo ogólna (i nieco bełkotliwa) definicja. Tysiąc rzeczy możemy nazwać web serwisem (przynajmniej pomijając to ostatnie: just-in-time). Niemniej jednak, kiedy się o tym mówi, to zazwyczaj ma się na myśli poniższe, konkretne podejście.

Pod terminem Usługi Web należy rozumieć zbiór małych funkcjonalnych klocków pracujących w sieci i udostępniających określone usługi innym systemom czy też innym webservices. Takimi klockami mogą być np. usługi pogodowe, translatory językowe czy mechanizmy przeliczające waluty, a wykorzystywane mogą być przez inne usługi udostępniające usługi bardziej złożone.
Należy pamiętać, że webservices zwykle nie będą stanowić samodzielnych aplikacji, a jedynie stanowić interfejs komunikacyjny pomiędzy klientami z sieci a systemami biznesowymi.
Wyjątkową siłą webservices jest wykorzystanie istniejących i szeroko stosowanych technologii tj. protokołu HTTP i języka XML. HTTP jest jednym z najbardziej rozpowszechnionych protokołów w sieci WEB, co umożliwia natychmiastowe wykorzystanie tej paltformy do przesyłania komunikatów.
XML dostarcza metajęzyk za pomocą którego porozumiewają się klienci z usługami oraz poszczególne komponenty.
Idąc dalej, przy budowie i wykorzystaniu webservices nie ma znaczenia w jakiej technologii jest napisana usługa, na jakim systemie operacyjnym pracuje. Pełną interoperatywność zapewnia protokół SOAP, który ma aspiracje stać się następcą technik typu CORBA, RMI czy DCOM.

O Web Services można również myśleć jako o rozwinięciu programowania obiektowego i przeniesienie go z zamkniętego środowiska, jakim jest pojedynczy komputer w środowisko sieciowe. Należy sobie wyobrazić, że obiekty i ich metody, zamiast znajdować się w jednym programie i być wykonywane w jednym procesorze

(czy w zestawie procesorów), są umieszczone na rozproszonych serwerach. Jednakże nie należy kojarzyć Web Services z przetwarzaniem rozproszonym. Każdy obiekt to usługa. Z technicznego punktu widzenia usługa, to instancja do której można wysłać zapytanie (czy rozkaz) i spodziewać się zwrotnej odpowiedzi (rezultatu zapytania). Usługa udostępnia na zewnątrz (do sieci) swój interfejs. Z naszego punktu widzenia (tj. ze strony informatyka chcącego zrobić użytek z usługi) nie jest istotne jak usługa zbudowana jest wewnątrz – tj. na jakiej maszynie działa, pod jakim systemem, w jakim języku została napisana. Ze wszystkimi usługami możemy porozumiewać się w jednakowy sposób (protokół SOAP, omówiony później), ich interfejs programistyczny (API) jest znany (opisany językiem WSDL), usługę możemy wyszukać dzięki usłudze katalogowej (UDDI). Ponieważ usługi mają możliwość komunikowania się ze sobą, w rzeczywistości dana usługa może być zbudowana z innych usług (ale jak pamiętamy wewnętrzna struktura usługi nie jest dla nas istotna i jest dla nas zupełnie przezroczysta).

Wykorzystując technologię Web Services, możemy łatwo, a więc tanio, zintegrować wewnątrz firmy systemy informatyczne pochodzące od różnych producentów i pracujące na różnych serwerach albo zorganizować pracę grupową (workflow).

Teraz możemy przenieść myślenie o Web Services na nieco wyższy poziom abstrakcji. Możemy przenieść je na poziom procesów biznesowych. Każda firma może być widziana w Internecie poprzez swoje usługi sieciowe, czyli końcówki jej wewnętrznych systemów i procedur. Łącząc na poziomie technicznym usługi sieciowe naszej firmy z innymi firmami, de facto na poziomie zarządzania firmą, łączymy swoje procesy biznesowe. Dla firmy to oznacza, możliwość szybkiego rozwoju poprzez łatwe nawiązywanie współpracy z innymi firmami, a więc możliwość specjalizowania się, ograniczenie kosztów (np. poprzez korzystanie z zewnętrznych systemów informatycznych i opłacaniu ich ryczałtem w przeciwieństwie do ponoszenia kosztów instalacji systemu wewnątrz firmy, etc.).

WS działa trochę jak RPC. Natomiast, co tu jest szczególne, to to, że będą się pojawiały metajęzyki opisu przesyłanych danych itd. Całość ma być przenośna i mocno ogólna. Chcemy skodyfikować zasady tworzenia luźno powiązanych systemów bazujących na rozproszonych usługach (w przeciwieństwie do pojedynczej implementacji).

Implementacja powinna pozwalać na zastosowanie wielowarstwowych mechanizmów bezpieczeństwa oraz udogodnienia umożliwiające konfigurowanie środowiska (np. mechanizmów autoryzacji, naliczania opłat itd.).

Chronologicznie i z grubsza dziać się ma tak:

- dostarczyciele usług - **rejestrują się** u pośredników (UDDI).

- klient - **pyta** pośrednika, kto mu da daną usługę. (SOAP).

- pośrednik (broker) - lokalizuje mu tę usługę i podaje opis jej użycia (WSDL).

- klient **podłącza się** do dostarczyciela i korzysta z usługi (SOAP)

Jest tu dość śliska sprawa dotycząca pośredników, rejstrów usług, centralizacji i czystości całego systemu -- Microsoft w roli głównego powiernika informacji o wszystkich usługodawcach na świecie.
(Na szczęście) można też się u nikogo nie rejestrować, klient może po prostu podłączyć się i pracować z danym serwerem.
System zawsze używa XML. Prawie zawsze transmisja danych jest po HTTP.
Takie "bardziej" pełne rozwiązania będą używały: XML + HTTP + SOAP + WSDL + UDDI.
I implementacja w Javie/J2EE.

Na czym bazują Webservices?

Warstwa komunikacji :

Protokół SOAP – Simple Object Access Protocol
Protokół SOAP jest protokołem jednokierunkowym. Zapytania do usługi wysyła się wiadomością SOAP. Wiadomość jest napisana w XMLu. Zastosowanie XMLa gwarantuje elastyczność budowania zapytań, tj. ich prostą modyfikację w oprogramowaniu klienckim, możliwość osadzania w zapytaniu innych dokumentów XML oraz łatwość debugowania (format XML jest czytelny dla człowieka). W wiadomości SOAP zawarte są takie informacje jak: co jest treścią wiadomości, kto i co ma za zadanie z nią uczynić, zbiór zasad reprezentowania danych oraz sposób osadzenia wiadomości w innych protokołach.

Protokół SOAP nie ma narzuconego sposobu transportowania go. Może być transportowany bezpośrednio przez protokół TCP, protokoły SMTP lub HTTP albo przez inny system, np. Jabber.

Wiadomość SOAP składa się z nagłówka i ciała podzielonych na bloki. Wysłana wiadomość SOAP wysyłana jest do odbiorcy przez wskazaną listę pośredników – węzłów SOAP. W każdym węźle odbywa się przetwarzanie wiadomości. W blokach nagłówka zapisane są zlecenia dla pośredników. Zlecenia nie są zdefiniowane przez żaden standard, dlatego może zdarzyć się, że pośrednik nie będzie potrafił przetworzyć wiadomości w sposób mu przypisany. Jeśli takie zlecenie posiadało atrybut, że musi być zrozumiane i wykonane przez pośrednika, przesyłanie przetwarzania wiadomości i dalszego jej przekazywania jest zatrzymane, a do nadawcy wraca wiadomość o błędzie. Pośrednik po wykonaniu swojego zadania usuwa z nagłówka blok, w którym zadanie zostało mu zlecone. Może także dopisać coś do niego. Następnie wiadomość przesyłana jest do kolejnego, wskazanego węzła SOAP.

Inaczej - SOAP to protokół definiujący formaty komunikatów, sposoby wysyłania komunikatów i odbierania odpowiedzi, kodowania danych w języku XML oraz gramatykę XML służącą do: określania nazw metod, definiowania typów parametrów i zwracanych wartości oraz opisu typów. SOAP określa także mechanizmy

wywoływania zdalnych procedur (RPC) przy pomocy protokołu komunikacyjnego HTTP. Żądania SOAP są wysyłane jako żądania HTTP POST, choć do komunikacji może być wykorzystany jakikolwiek protokół transportowy.
Cały proces wymiany komunikatów rozpoczyna się od zapytania SOAP wysłanego jako HTTP POST z typem zawartości text/xml i polem SOAPAction zawierającym nazwę metody SOAP. W komunikacie jest także miejsce na dane w postaci XML. Odbiornik na serwerze (np. servlet) kontroluje pole SOAPAction i w zależności od jego zawartości podejmuje odpowiednie działania. Na podstawie treści XML odnajdywana jest odpowiednia usługa która wykona metodę i zwróci rezultat. Następnie rezultat jest zwracany klientowi jako dokument XML ze standardowym nagłówkiem HTTP i typem zawartości text/xml.
Podczas wymiany dużej ilości komunikatów znaczącą wartością staje się wydajność. W przeciwieństwie do technologii typu CORBA, RMI czy DCOM gdzie dane transportowane są w postaci binarnej (brak metadanych), SOAP przesyła komunikaty czystym i otwartym XML. SOAP wraz z samoopisującymi się danymi w XML umożliwia łatwe i szybkie przystosowanie każdego systemu do wykorzystania stosowanej implemenntacji SOAP, co w przypadku systemów binarnych sprawia duże problemy w związku z różnymi sposobami kodowania informacji. Czas potrzebny na zakodowanie/odkodowanie komunikatu SOAP jest nieporównywalnie mały do czasu przesłania takiego komunikatu przez sieć.

SOAP to specyfikacja protokołu do wywoływania metod serwerów, serwisów, komponentów oraz obiektów. SOAP kodyfikuje wykorzystywane w praktyce zastosowanie XML-a oraz HTTP jako mechanizmów wywoływania metod. Specyfikacja SOAP-a określa kilka rodzajów nagłówków HTTP ułatwiających filtrację firewall-om i proxy. Definiuje także słownik znaczników XML-owych wykorzystywanych do reprezentacji parametrów, zwracanych wartości oraz rzucanych przez metody wyjątków.

Najczęściej wykorzystuje HTTP. Ale nie musi.

roboczy szkic (working draft) SOAP w wersji 1.2 -- 9 lipca 2001.
http://www.w3.org/TR/soap12/

Czemu nie jakieś typu DCOM/CORBA, tylko coś nowego?

Wbrew pozorom nie uważam, że jest to kretyńskie pytanie. Myślę, że jest coś na rzeczy w kwestii popularności XML-a (złośliwe), ale też jego tekstowości, ogólności, no i jest też kwestia wielu stubów z CORBY itp. rzeczy - nałożenie tego na HTTP nie jest takie oczywiste.
Teksty marketingowe, których pełno, mówią też o problemach z bezpieczeństwem w tego typu systemach (kiepski argument), więc rzekomo z tego powodu wiele firewalli i serwerów proxy blokuje przepływ tego typu informacji (równie kiepski - czemu nie można zrobić CORBy na porcie 80).

struktura SOAP-komunikatu

- **envelope** - opakowanie; jakaś tam przestrzeń nazw
- **header** (opcjonalny) -- dodatkowe informacje
- **body** - zawiera parametry wywołania / odpowiedź / informacje o błędach

Prosty dokument SOAP z żądaniem ceny mydła:

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope"
 env:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
  <env:Body>
    <m:GetPrice xmlns:m="Some-URI">
      <Item>Soap</Item>
    </m:GetPrice>
  </env:Body>
</env:Envelope>
```

Ogólne zasady:

- *MUSI* być kodowany przy użyciu XML-a
- *MUSI* mieć element Envelope
- *MOŻE* mieć element Header
- *MUSI* mieć element Body
- *MUSI* używać odpowiednich przestrzeni nazw
- *NIE MOŻE* zawierać referencji do DTD
- *NIE MOŻE* zawierać instrukcji procesora XML-a

Obsługa błędów:

Informacje o błędach przetwarzania zawarte są wewnątrz elementu Fault. Element musi pojawić się wewnątrz elementu Body i może wystąpić tylko raz w komunikacie SOAP.

Elementy zawarte wewnątrz Fault:

- **faultcode**: WYMAGANY kod błędu, do użycia przez oprogramowanie
- **faultstring**: WYMAGANY czytelna dla człowieka nazwa błędu
- **faultactor**: (Aplikacje nie będące końcowym odbiorcą komunikatu) - kto wygenerował błąd
- **detail**: (Jeśli nie dało się przetworzyć zawartości elementu Body) - specyficzne informacje o błędzie

Kody błędów - faultcodes:

- **VersionMismatch**: Niepoprawna przestrzeń nazw dla elementu Envelope.
- **MustUnderstand**: Jeden z elementów Header z atrybutem mustUnderstand ustawionym na "1" nie został zrozumiany.
- **Client**: Komunikat nie został poprawnie sformatowany albo zawierał niepoprawne dane.
- **Server**: Wystąpił problem z serwerem tak, że wiadomość nie mogła być przetworzona.

Przykładowy komunikat:

```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope"
 env:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
  <env:Body>
    <env:Fault>
      <faultcode>env:MustUnderstand</faultcode>
      <faultstring>SOAP Must Understand Error</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Zanurzenie w HTTP - żądanie:

```
POST /StockQuote HTTP/1.1
Host: www.stocksserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<env:Envelope
 xmlns:env="http://www.w3.org/2001/06/soap-envelope">
   <env:Body>
     <m:GetStockQuote xmlns:m="Some-URI"
      env:encodingStyle="http://www.w3.org/2001/06/soap-encoding">
        <symbol>SUNW</symbol>
     </m:GetStockQuote >
   </env:Body>
</env:Envelope>
```

Odpowiedź:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<env:Envelope xmlns:env=http://www.w3.org/2001/06/soap-envelope">
   <env:Body>
           <m:GetStockQuoteResponse xmlns:m="Some-URI"
        env:encodingStyle="http://www.w3.org/2001/06/soap- encoding">
              <price>5.00</price>
       </m:GetStockQuoteResponse>
   </env:Body>
</env:Envelope>
```

Bezpieczeństwo

SOAP jako, że używa HTTP jako protokołu transportu, jest bezstanowy.
Nie implementuje bezpieczeństwa. Transport przez HTTP pozwala jednak na zastosowanie SSL-a na poziomie aplikacji.
Pole SOAPAction nagłówka HTTP pozwala firewallom na filtrowanie komunikatów SOAP i np. zupełnie zabronić wywołań SOAP. Firewalle mogą filtrować pakiety SOAP bazując na nazwie obiektu, poszczególnej metody albo brać pod uwagę oba te kryteria.

Warstwa opisu usług :

WSDL – Web Service Definition Language

Język w którym opisany jest interfejs usługi. WSDL przedstawia w jaki sposób można korzystać z usługi, a więc jakie przyjmuje rozkazy, jakie zwraca odpowiedzi, jakich formatów danych używa.

WSDL jest nadzbiorem języka SDL, umożliwia twórcom usługi Web opisanie co potrafi usługa, gdzie się znajduje i jak ją wywołać. Usługa Web może być pytana o listę udostępnianych metod. Odpowiedź powinna zawierać opis w zrozumiałym formacie. Język WSDL jest przydatny przy automatyzacji komunikacji pomiędzy usługami webservices umożliwiając współgranie usług.

W myśl języka WSDL usługę pojmujemy jako pewien zbiór portów. Każdy port jest skojarzony ze swoim protokołem i ma określony typ. Typ portu związany jest z obsługiwanymi operacjami. Komunikacja z usługą odbywa się za pomocą zdefiniowanych wiadomości zawierających zdefiniowane typy danych. Usystematyzowanie tego myślenia znajduje odzwierciedlenie w postaci formatu XML, o którą opiera się WSDL. WSDL definiuje te wszystkie elementy (typy danych, wiadomości, typy portów, powiązania z protokołami, porty, usługę). Może też zawierać dokumentację słowną dla programistów, którzy będą programowali użycie usługi.

Metajęzyk WSDL
Specyfikacja definiuje następnie język opisu interfejsu SOAP serwera zwany WSDL.
Format WSDL jest również XML-owy.
Przypominam: po stworzeniu serwisu możemy opublikować jego opis i umieścić odwołanie do tego opisu w repozytorium (UDDI, patrz dalej) tak, aby umożliwić odnalezienie go przez potencjalnych użytkowników.
Wszystkie operacje w ramach Web Services (rejestracje, wyszukiwania, przekazywania danych) są realizowane konsekwentnie po protokole SOAP.
Kiedy ktoś zechce użyć danego serwisu, zagląda do odpowiedniego pliku w formacie WSDL i znajduje tam lokalizację serwisu, udostępniane funkcje oraz metody ich wywoływania.
Klient używa opisu w WSDL-u do sformowania żądania SOAP do żądanego serwisu.

Warstwa komunikacji usług :

UDDI – Universal Description, Discover and Integration

Usługa katalogowa dla usług sieciowych. W katalogu znajdują się profile różnych firm, tj. ich opis, informacje kontaktowe oraz to jakie usługi sieciowe udostępniają. W katalogu UDDI znajdują się specyfikacje usług podane w języku WSDL. Katalog UDDI jest jednocześnie przykładem usługi sieciowej, komunikacja z katalogiem odbywa się poprzez SOAP (a dla człowieka poprzez aplikację, np. serwis WWW). W obecnej chwili utrzymywaniem katalogów UDDI zajmują się firmy Microsoft i IBM.

Inaczej UDDI (Universal Description, Discovery and Integration Service) to usługa udostępniająca klientom mechanizmy dynamicznego wyszukiwania innych usług Web. UDDI stanowi interfejs umożliwiający dynamiczne połączenie się z usługą udostępnianą przez innego usługodawcę. Rejestry UDDI zawierają:
- informacje o webservices na bazie nazwy usługodawcy, jego adresu, kategorii biznesowej czy informacji technicznej itp.,
- operacje dotyczące usługi WEB, tj, rejestracji, wyszukiwania i korzystania z usługi
- szczegóły udostępniane przez niskopoziomowe API

UDDI posiadają dwa rodzaje klientów: usługodawców publikujących swoje usługi oraz klientów pragnących skorzystać z tych usług. Warstwa UDDI leży nad protokołem SOAP, przez co komunikaty UDDI stanowią obiekty w komunikatach SOAP.

Podział przechowywanych danych:
White Pages: ogólne dane każdej firmy
Yellow Pages: ogólna klasyfikacja
Green Pages: techniczne dane

Warstwa transportu :

Usługi sieciowe (Web Services) komunikują się ze sobą poprzez wymienianie wiadomości SOAP. XML-owa postać protokołu SOAP jest jego silną zaletą, ale nie jedyną. Drugą istotną cechą jest fakt, że SOAP jest protokołem warstwy aplikacji i do transportu potrzebuje być osadzony w innym protokole. Specyfikacja SOAP wprowadza dowolność co do wyboru protokołu podrzędnego. SOAP może być skojarzony z protokołami takimi jak TCP, UDP, HTTP, SMTP, ale nie tylko. Wybór protokołu podrzędnego wiąże się z wyborem pewnych cech i mechanizmów, które będą dotyczyły transportu wiadomości SOAP.

Service	Service	Service	Service	Service
SOAP Message	SOAP Message	SOAP Message	SOAP Message	SOAP Message
Protocol Binding	Protocol Binding	Protocol Binding	Protocol Binding	Protocol Binding
HTTP	SMTP	SIP	TCP	UDP
TCP	TCP	UDP		

W miejsce klasycznych protokołów Internetowych można wstawić inny system transportu gwarantujący bardziej pożądane cechy, przykładem takiego systemu jest Jabber.

Wymagania dla Web Services

Podstawy Web Services to jak przedstawiono powyżej: protokół SOAP, WSDL, UDDI. Jednakże te cegiełki nie tworzą jeszcze pełnego systemu. Web Services w dużej mierze pozostają nadal koncepcją. World Wide Web Consortium (W3C), pod którego patronatem pomysł Web Services się rozwija, dopiero niedawno powołał grupę, która ma stworzyć oficjalne wzorce dla systemów Web Services. Nie da się ukryć, że usługi sieciowe tworzone są dla potrzeb Internetu komercyjnego, także to właśnie od biznesu płyną postulaty dotyczące Web Services, które zostaną za chwilę przedstawione.

Bezpieczeństwo

Nie od dziś wiadomo, że możliwości swobodnej transmisji, jakie daje Internet, jest wykorzystywana na bardzo wiele złych sposobów, przed którymi należy się bezwzględnie zabezpieczać: kradzież informacji przemysłowej, uniemożliwianie prowadzenia działalności elektronicznej, celowe działania destrukcyjne, podszywanie się, etc. Dlatego system, który ma być traktowany poważnie przez system, musi być systemem, który dba o bezpieczeństwo. Problem jest o tyle większy, że teraz przez Internet mają przedostawać się dane, które dotychczas krążyły wyłącznie wewnątrz lokalnych sieci firmowych lub w intranecie po łączach dzierżawionych, nie po publicznych. Teraz zostaną przedstawione wymogi bezpieczeństwa, metody zabezpieczania dokumentów i transmisji oraz inne aspekty bezpieczeństwa w systemach Web Services.

Zarządzanie

Zarządzanie jest silnie powiązane ze sprawą bezpieczeństwa. Usługi sieciowe poprzez katalog UDDI wystawione są na widok publiczny, a specyfikacja ich interfesjsu jest sprawą jawną. W środowisku, gdzie każdy może zpreparować i wysłać do usługi poprawną wiadomość SOAP, bezwzględnie potrzebne jest zarządzanie dostępem i prawami do usługi (listy kontroli dostępu, ACL). Zwłaszcza, gdy za świadczenie usługi będą pobierane opłaty.

Z tym zagadnieniem związany jest temat zarządzania kluczami publicznymi. Przy ogromnym wymaganiu na bezpieczeństwo, w sieci pojawi się mnóstwo kluczy publicznych, którymi będą podpisane całe masy dokumentów. Klucze generowane przez firmy, certyfikaty pochodzące o zaufanych stron, doczepiane do dokumentów lub porozrzucane po różnych lokalizacjach.

Trzecim istotnym zagadnieniem dla zarządzania, jest właśnie pobieranie opłat. Jest to również zagadnienie rozległe. Po pierwsze dlatego, że różne usługi będą miały różnych odbiorców. Odbiorcą usługi może być współpracująca firma, a więc rozliczenie firma-firma, może też być nią indywidualny użytkownik Internetu, czyli de facto osoba anonimowa. Rozliczanie opłat za korzystanie z usług sieciowych ma jeszcze jeden aspekt, ze względu na fakt, że dana usługa może być zbudowana z innych usług.

Niezawodność i wydajność

Założeniem Web Services jest, że jedna firma, będzie budowała swoją usługę w oparciu o usługi innych firm. Tworzy się więc siatka powiązań. Jedno nie działające ogniwo może zaważyć o całej funkcjonalności usługi. W siatce powiązań rozpiętej na rozległej sieci Internet bardziej niż w przypadku integracji wewnętrznych systemów komputerowych trzeba zwrócić uwagę na sprawny transport wiadomości między systemami. Na przeszkodzie mogą stanąć wolne łącza, tymczasowe niedostępności zdalnych systemów, opóźnienia w dostarczaniu wiadomości, przeszkody architektoniczne.

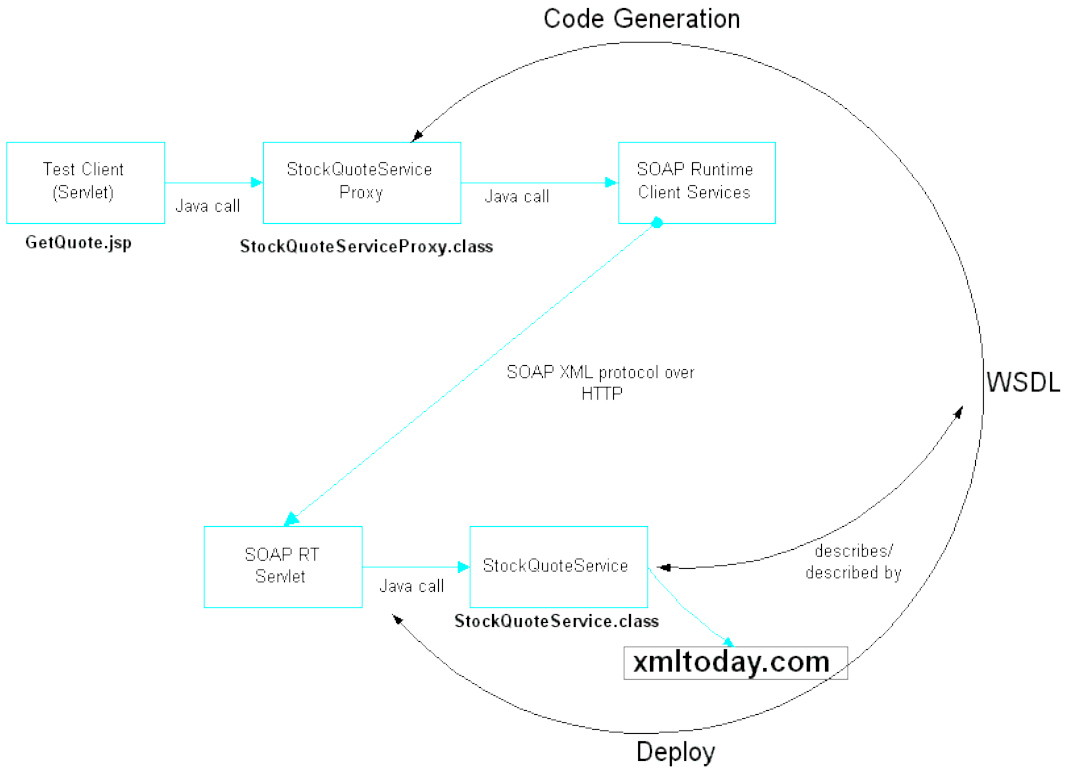
Mobilność i dostępność

Podstawowy protokół SOAP nie jest stricte związany z żadną konkretną techniką transportową. Niektórzy chcieliby wykorzystać ten fakt i zastosować system transportu, który uwolniłby usługi sieciowe od sztywnego przydziału adresu IP, czy mało dynamicznej usługi DNS, wprowadzając możliwość występowania usługi tam, gdzie nie jest możliwe przydzielenie stałego publicznego adresu IP, np. gdy serwer usługi stoi za firewallem, NATem lub ma przydzielany numer z DHCP. Takie problemy zdają się nie występować z dużymi serwerami internetowymi, jednakże należy pamiętać, że oprogramowanie typu Web Services może też się znajdować w oprogramowaniu klienckim. W raporcie zostanie zaprezentowany system transportu obchodzący ograniczenia wynikające z architektury sieci Internet.

Jak zacząć z WebServces?

Wydaje się, że najpopularniejsza w zastosowaniu jest *Java*. Mamy w niej API XML-owe:
(Sun) JAXP (processing), JAXB (binding) , JAXM (messaging), JAXR (refistries) , JAX-RPC.
Istnieje szereg komercyjnych IDE do rozwijania web serwisów. Wśród nich mamy np. odpowiedni moduł VisualAge IBM'a (dawniej znany jako produkt stand-alone - WSDE - Web Services Development Environment).
Przy użyciu WSDE można łatwo zrobić sobie prosty web serwis, praktycznie nie znając szczegółów SOAP-a, WSDL-a, ani nawet XML-a. Dostarczamy narzędziu klasę typu JavaBean i wskazujemy metody, które chcemy udostępnić. WSDE generuje z tego gotowy do opublikowania opis interfejsu WSDL. Możemy teraz podpiąć ten interfejs do naszego serwera SOAP; jak również wygenerować sobie szkielet klienta; ściślej - klasę stanowiącą "proxy" zwykłe-wywołanie-metody tłumaczącą to na niższy poziom, czyli wywołania/odczyt wyniku SOAP.

Przykład: Dostarczamy klasę StockQuoteService (która będzie skądś wiedziała, jakie są kursy akcji - tu robi to ściągając je z xmltoday.com). IDE generuje opis WSDL, podłącza go pod servlet nasłuchujący/odpowiadający po SOAP na naszym serwerze oraz generuje kod klasy StockQuoteServiceProxy.



Przykład 2:
Tworzymy własny Web Service przy użyciu Visual Studio .NET

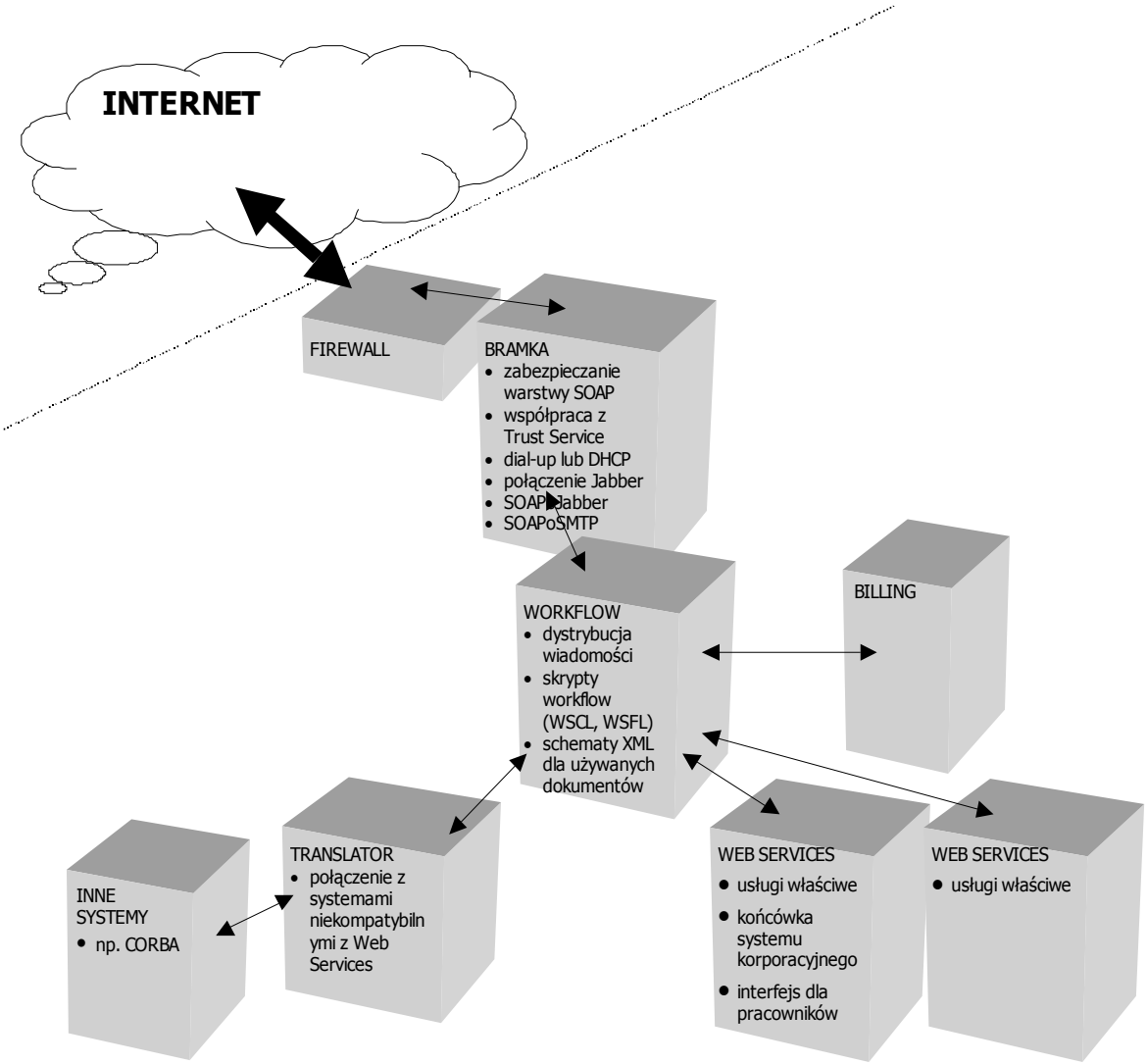
Web Service będzie zestaw 6 losowych numerów LOTTO pomiędzy 1 a 49, z liczbami zmieniającymi się przy każdym dostępie do usługi.

Krok po kroku:
(niedługo zostanie dołączony opis)

Architektura WebServices

Architektura opisuje trzy role: **dostarczyciela** serwisu, **klienta** serwisu oraz **pośrednika**. Obrazuje także trzy główne operacje: **publikacji**, **wyszukiwania** oraz **wiązania**.

Lokalna sieć firmowa znajdująca się na brzegu sieci



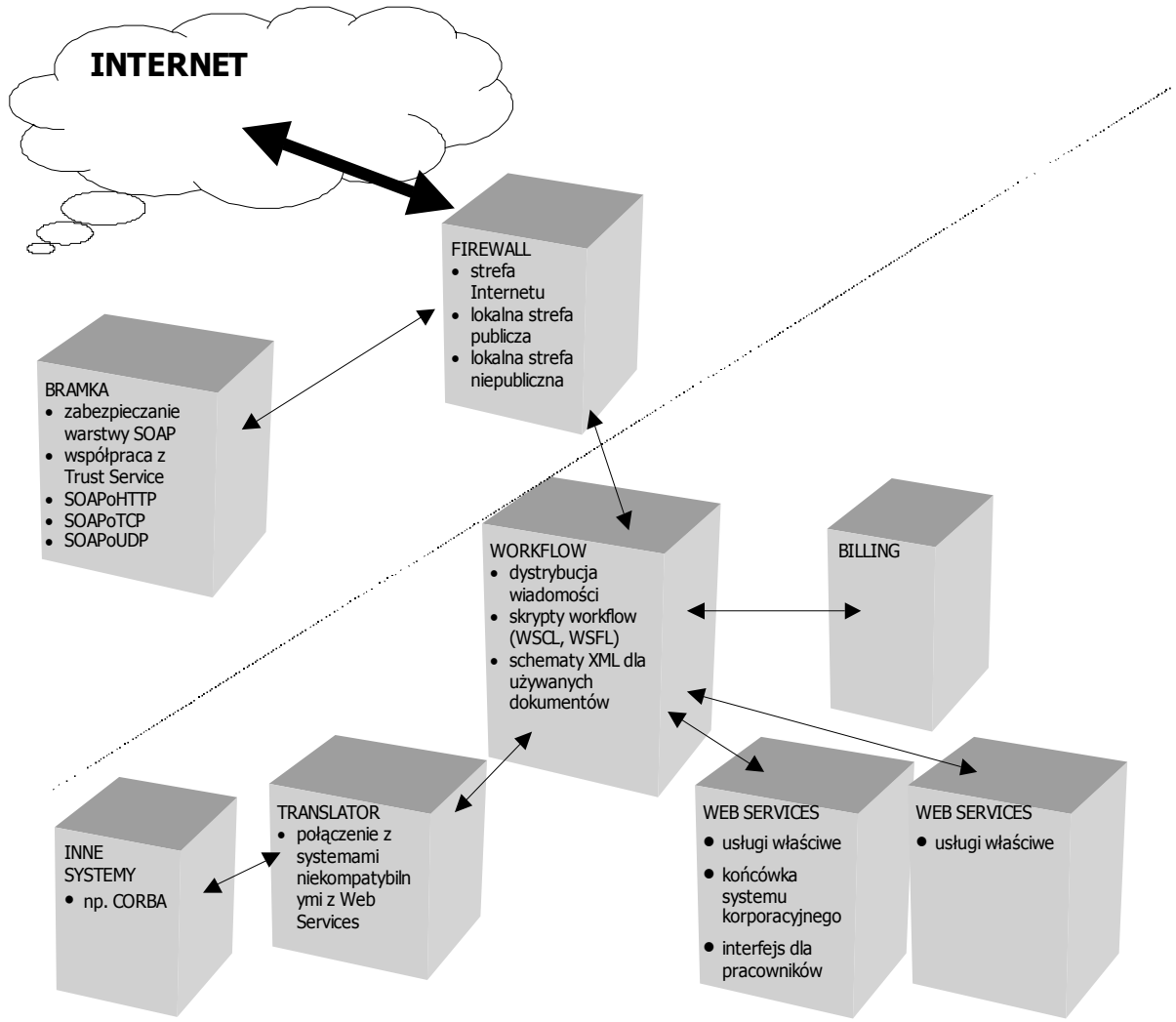
Przykład architektury Web Services w sieci firmowej. Przypuśćmy, że firma z jakiś względów (koszta, ograniczenia, lokalizacja) nie posiada stałego łącza i wykonuje cykliczny dostęp wdzwaniany do sieci (ew. posiada stałe łącze, ale bez stałego publicznego numeru IP). Do realizowania usług sieciowych wystarcza cykliczne pobieranie zgłoszeń SOAP i przetwarzanie ich. Zgłoszenia SOAP spoczywają na serwerze pocztowym w Internecie lub na serwerze Jabber, skąd są pobierane z chwilą nawiązania połączenia.

Urządzenie pełniące funkcję firewalla chroni sieć przed atakami, jednak dla wiadomości SOAP jest zupełnie przezroczyste. Wiadomości SOAP trafiają do bramki, która jest węzłem pośrednim SOAP, do którego wiadomości zostały skierowane. Utrzymanie bezpieczeństwa warstwy SOAP jest domeną tej bramki. Bramka podpisuje do późniejszej autoryzacji dokumenty wychodzące oraz sprawdza wiarygodność dokumentów przychodzących. W wybranych przypadkach wspiera się zewnętrzną usługą zaufania (Trust Service), która potrafi dostarczyć klucze publiczne nadawcy lub też sprawdzić od razu jego podpis. Bramka bezpieczeństwa zdejmuje odpowiedzialność za bezpieczeństwo z działających w sieci aplikacji.

Sercem sieci jest system workflow, który nawiguje wątkami konwersacyjnymi zachodzącymi między usługami według zadanych mu schematów. Schematy są łatwe w modyfikacji, może nimi zajmować się niżej wykwalifikowana kadra IT. Modyfikacje nie wymagają przerabiania samych usługi.

Do systemu workflow dołączone są docelowe usługi sieciowe, system billingowy oraz translator komunikatów SOAP na protokoły zrozumiałe systemom nie budowanym z myślą o działaniu w środowisku Web Services. Usługi docelowe są de facto stykiem z systemami korporacyjnymi, po ‘drugiej stronie’ których znajdują się zasoby ludzkie tej firmy.

Lokalna sieć firmowa z pełnym dostępem do Internetu



Ten przykład różni się od poprzedniego typem dostępu do Internetu. Firewall i bramka bezpieczeństwa znajdują się w strefie adresacji publicznej. Dzięki temu bramka bezpieczeństwa jako pierwszy pośrednik SOAP jest adresowalna i osiągalna z Internetu. Bramka z wnętrzem sieci lokalnej komunikuje się w celu wymiany wiadomości SOAP poprzez urządzenie firewall. Dzięki rozróżnieniu trzech stref w urządzeniu (strefa Internetu, strefa lokalna publiczna, strefa lokalna niepubliczna), administrator może skonfigurować ruch w sieci tak, że komunikacja w warstwie SOAP między Internetem, a wnętrzem sieci firmowej zawsze będzie przebiegała przez bramkę bezpieczeństwa.