

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Java. Usługi WWW. Vademecum profesjonalisty

Autorzy: Steve Graham, Simeon Simeonov,
Toufic Boubez, Doug Davis, Glen Daniels, et al.
Tłumaczenie: Grzegorz Kowalski, Cezary Welsyng
ISBN: 83-7197-991-6

Tytuł oryginału: [Building Web Services with Java:
Making Sense of XML, SOAP](#)

Format: B5, stron: 544



Era e-biznesu, w którą wkracza światowa gospodarka, pociąga za sobą konieczność integracji złożonych systemów informatycznych. Usługi WWW (webservices) mają na tym polu do odegrania ważną rolę. Dzięki nim aplikacje mogą komunikować się z innymi aplikacjami poprzez Internet za pomocą standardowych protokołów, niezależnie od tego, w jakim języku zostały napisane i na jakiej platformie je uruchomiono.

Książka „Java. Usługi WWW. Vademecum profesjonalisty” przeznaczona jest dla programistów mających pewne doświadczenie w pisaniu aplikacji działających w Internecie. Jej celem jest zapoznanie Czytelnika z pojęciem usług WWW oraz wszystkimi elementami niezbędnymi do ich wykorzystania w biznesie. Poznasz założenia technologii usług WWW i schemat zależności pomiędzy nowymi standardami, takimi jak Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) oraz Universal Description Discovery and Integration (UDDI). Dzięki przykładowi kodu szybko nauczysz się implementować usługi WWW w języku Java.

- Dowiesz się, jakie są ogólne założenia architektury usług WWW
- Poznasz język XML będący podstawą innych standardów, wykorzystywanych do budowy usług WWW
- Zaznajomisz się ze standardem SOAP i poznasz jego zastosowania w e-biznesie
- Stworzysz własne usługi WWW w oparciu o Apache Axis i Javę
- Nauczysz się opisywać usługi WWW, tak by mogły być automatycznie wyszukiwane przez aplikacje
- Poznasz najważniejsze platformy, na których buduje się usługi sieciowe: J2EE, .NET, a także moduły SOAP::Lite (Perl) i platformę GLUE

„Java. Usługi WWW. Vademecum profesjonalisty” to książka, która nie tylko przedstawia całą dzisiejszą wiedzę na ten temat, ale także prezentuje praktyczne sposoby jej wykorzystania. Jeśli chcesz być na bieżąco ze światowymi trendami w integrowaniu złożonych aplikacji biznesowych – musisz ją przeczytać.



Spis treści

Podziękowania.....	9
O Autorach	11
Wprowadzenie	13
Rozdział 1. Wprowadzenie do usług sieciowych	19
Czym jest usługa sieciowa?.....	19
Perspektywa biznesowa.....	21
Perspektywa techniczna	22
Wykorzystanie usług sieciowych.....	22
Integracja aplikacji korporacyjnych.....	23
B2B	24
Trendy w e-biznesie	25
Dlaczego potrzebujemy usług sieciowych?	26
Zakres problemu.....	28
Rdzenne technologie.....	28
Dynamika przemysłu	29
Architektury zorientowane na usługi.....	32
Stosy technologii zapewniających współdziałanie usług sieciowych	34
Stos połączenia.....	35
Stos opisu.....	36
Stos wyszukiwania	39
Połączenie stosów technologii.....	39
Podsumowanie.....	41
Rozdział 2. Elementarz XML.....	43
Pochodzenie XML-a	44
Dwie twarze XML-a — treść kontra strukturalizacja danych.....	46
Dokumenty XML zorientowane na treść.....	46
Dokumenty XML zorientowane na strukturalizację danych	47
Czas życia dokumentu	48
Instancje dokumentów XML.....	49
Nagłówek dokumentu	49
Elementy	50
Atrybuty	52
Dane znakowe.....	55
Uproszczona wersja zamówienia.....	57
Przestrzenie nazw XML	58
Mechanizm przestrzeni nazw	59
Składnia przestrzeni nazw	60
Atrybuty z prefiksem przestrzeni nazw	62
Definicje typu dokumentu.....	63
Poprawność składniowa i strukturalna	64
Struktura dokumentu	65
Czy DTD wystarcza?.....	66

XML Schema	67
Podstawy XML Schema	68
Wiązanie dokumentu ze schematem	69
Typy proste	69
Typy złożone	73
Schemat zamówienia	75
Podstawy wielokrotnego wykorzystania schematów	78
Zaawansowane techniki wielokrotnego wykorzystania schematów	84
To jeszcze nie wszystko	91
Przetwarzanie dokumentów XML	91
Podstawowe operacje	91
Przetwarzanie dokumentów o silnie ustrukturalizowanych danych	93
Implementacja metody checkInvoice na podstawie interfejsu SAX	96
Implementacja metody checkInvoice na podstawie interfejsu DOM	102
Testowanie kodu	107
Podsumowanie	109
Zasoby	111
Rozdział 3. Simple Object Access Protocol (SOAP)	113
Rozwój protokołów XML	114
Protokoły XML pierwszej generacji	114
Simple Object Access Protocol (SOAP)	116
Powstanie protokołu SOAP	117
Co SOAP powinien proponować?	118
Czym naprawdę jest SOAP?	119
Prowadzenie interesów z firmą SkatesTown	120
Współdziałanie z systemem magazynowym	122
Usługa sieciowa do kontroli zapasów	124
Wybór platformy dla usług sieciowych	124
Z perspektywy dostawcy usługi	124
Z perspektywy klienta usługi	126
Testowanie usługi	127
SOAP w działaniu	128
Model koperty SOAP	131
Koperta SOAP	132
Wersje protokołu SOAP	132
Nagłówki SOAP	133
Treść komunikatu SOAP	135
Wykorzystanie rozszerzeń SOAP	135
Z perspektywy klienta usługi	135
Z perspektywy dostawcy usługi	137
Testowanie usługi	140
SOAP w działaniu	140
Pośredniki SOAP	142
Potrzeba istnienia pośredników	142
Pośredniki w protokole SOAP	143
Podsumowanie	144
Obsługa błędów w protokole SOAP	147
Schemat przetwarzania komunikatu SOAP	148
Kodowanie danych w protokole SOAP	149
Wykorzystanie różnych metod kodowania	149
Reguły kodowania danych w protokole SOAP	150
Wybór metody kodowania danych	156

	Projektowanie systemów rozproszonych opartych na usługach sieciowych.....	161
	Przesyłanie komunikatów.....	162
	Przekazywanie komunikatów a RPC.....	166
	Zdalne wywoływanie procedur przez SOAP.....	168
	Usługa sieciowa do wysyłania zamówień.....	171
	Schematy zamówienia i faktury.....	171
	Z perspektywy klienta usługi.....	175
	Z perspektywy dostawcy usługi.....	177
	Testowanie usługi.....	178
	SOAP w akcji.....	178
	Wiązania protokołu SOAP.....	180
	Ogólne uwagi.....	180
	HTTP(S).....	182
	Komunikaty SOAP z załącznikami.....	183
	Wiązanie SOAP do protokołu SMTP.....	184
	Inne protokoły.....	185
	Podsumowanie.....	185
	Co dalej?.....	186
	Zasoby.....	187
Rozdział 4.	Tworzenie usług sieciowych.....	189
	Czym jest Axis i dlaczego właśnie z niego będziemy korzystać?.....	189
	Architektura platformy Axis.....	190
	Komponenty Axis.....	190
	Określanie łańcucha dla usługi.....	200
	Parsowanie XML-a.....	201
	Instalacja serwera Axis.....	201
	Konfiguracja platformy Axis.....	201
	Metody konfiguracji.....	205
	Bezpieczeństwo.....	207
	Proste usługi sieciowe.....	207
	Programowanie po stronie klienta.....	208
	Zaawansowane aspekty wprowadzania usług sieciowych.....	210
	Usługi zorientowane na przetwarzanie dokumentów.....	211
	Kodowanie i dekodowanie danych.....	214
	Tworzenie procedur obsługi komunikatów.....	216
	Wyspecjalizowane procedury nawrotu — Interfejsy.....	217
	Błędy.....	218
	Wzorce komunikacji.....	219
	Tworzenie i wprowadzanie pośrednika.....	220
	SOAP 1.2.....	221
	Monitoring.....	221
	Podsumowanie.....	222
Rozdział 5.	Zastosowania SOAP w e-biznesie.....	223
	Bezpieczeństwo usług sieciowych.....	224
	Przykładowy scenariusz.....	225
	SSL i podstawowe uwierzytelnianie za pomocą HTTP.....	226
	Podpis cyfrowy.....	237
	Szyfrowanie dokumentów XML.....	242
	Usługa notarialna.....	246
	Autoryzacja.....	247
	Asercje bezpieczeństwa.....	251
	Infrastruktura klucza publicznego i zarządzanie kluczami.....	252
	Od czego zacząć przy wdrażaniu rozwiązań zapewniających bezpieczeństwo?.....	257

Integracja systemów przedsiębiorstwa.....	258
Serwer SOAP na podstawie J2EE	258
Przetwarzanie transakcji.....	260
ACID i dwufazowe zatwierdzenie	266
Nieawodne przesyłanie komunikatów	270
Model bezpieczeństwa J2EE.....	278
Jakość usług.....	281
Serwer SOAP na potrzeby przedsiębiorstwa.....	281
Szeroki dostęp.....	282
Zarządzanie systemem	283
Zaawansowane zarządzanie bezpieczeństwem.....	285
Podsumowanie.....	286
Zasoby.....	287
Rozdział 6. Opisywanie usług sieciowych	291
Do czego potrzebne są opisy usług sieciowych?.....	291
Rola opisów w architekturze zorientowanej na usługi	292
Dobrze zdefiniowana usługa.....	293
Opis funkcjonalny.....	293
Opis niefunkcjonalny	294
Opis agregacji (aranżacji, kompozycji)	294
Wieża opisu usług — podsumowanie	295
Historia języków definicji interfejsu.....	296
Standard WSDL.....	300
Model informacyjny WSDL	300
Elementy języka WSDL.....	303
Element PortType.....	309
Element Operation.....	310
Element Message.....	314
Element Binding.....	318
Element Port.....	325
Element Service.....	326
Element Definitions	327
Element Documentation	327
Zastosowanie elementu Import.....	328
Mechanizm rozszerzeń w WSDL.....	331
Języki WSDL i Java	333
Generowanie kodu na podstawie specyfikacji WSDL	333
Kierunki rozwoju standardów opisu usług	355
Język WSEL.....	355
Język WSFL.....	355
Podsumowanie.....	357
Rozdział 7. Wyszukiwanie usług sieciowych	359
Znaczenie wyszukiwania usług.....	359
Rola rejestrów.....	360
Wyszukiwanie usług w czasie projektowania i podczas działania	360
Różne mechanizmy wyszukiwania usług.....	361
Zmiana scenariusza.....	363
Standard UDDI.....	364
Model korzystania z UDDI.....	365
Koncepcja struktury tModel w rejestrze UDDI.....	374
Publikowanie informacji o firmie w rejestrze UDDI	387
Publikowanie informacji o usługach w rejestrze UDDI	393
Wyszukiwanie informacji w rejestrze UDDI.....	403

	Pobieranie szczegółowych informacji na temat firm i usług w rejestrze UDDI.....	411
	Podsumowanie specyfikacji UDDI 1.0	412
	Prywatne rejestry UDDI.....	412
	Po co firmie prywatny rejestr UDDI?	413
	Pięć rodzajów prywatnych rejestrów UDDI	415
	Co nowego w wersji 2.0?.....	420
	Zestawienie zmian w UDDI 2.0	420
	Własne taksonomie.....	420
	Modelowanie relacji pomiędzy wpisami businessEntity	423
	Zmiany w API zapytań	426
	Zmiany w API publikacji.....	433
	Inne zmiany	436
	WSDL w UDDI.....	439
	Zapisywanie w UDDI elementu businessService opartego na dokumencie WSDL....	439
	Bardziej złożone dokumenty WSDL i odpowiadające im wpisy UDDI	442
	Podsumowanie — przykład dynamicznego wyszukiwania dokumentu WSDL w UDDI	446
	Podsumowanie.....	455
Rozdział 8.	Zgodność operacyjna, narzędzia i warstwa pośrednia.....	457
	Zgodność operacyjna — istota usług sieciowych.....	457
	Wspólnota twórców implementacji SOAP	458
	Laboratorium zgodności operacyjnej.....	459
	Konsorcjum W3C — pojawienie się standardu SOAP.....	460
	Szersze spojrzenie na usługi sieciowe.....	461
	Kto tworzy systemy na podstawie SOAP?	462
	Inne języki i środowiska.....	463
	SOAP::Lite — usługi sieciowe w języku Perl.....	464
	Środowisko usług sieciowych .NET — krótkie wprowadzenie.....	466
	GLUE — jeszcze jedno rozwiązanie dla usług sieciowych w języku Java.....	473
	Podsumowanie.....	476
	Zasoby.....	476
Rozdział 9.	Przyszłe koncepcje	479
	Uniwersalne przetwarzanie informacji.....	479
	Wizja wszechobecných usług sieciowych.....	480
	Ontologie i semantyczny Internet.....	484
	Model opisu zasobów	484
	Ontologie.....	486
	RDF a usługi sieciowe.....	486
	Agenty programowe	487
	Agenty programowe a usługi sieciowe.....	488
	Przetwarzanie danych typu P2P.....	489
	P2P a usługi sieciowe.....	490
	Siatkowe przetwarzanie danych.....	490
	Siatkowe przetwarzanie danych a usługi sieciowe.....	492
	Wbudowane usługi sieciowe	492
	Wizja połączonych technologii.....	492
	Zasoby.....	493
	Słowniczek.....	495
	Skorowidz.....	515

Rozdział 1.

Wprowadzenie do usług sieciowych

W tym rozdziale wprowadzimy podstawową terminologię wykorzystywaną w dalszej części książki. Zdefiniujemy pojęcie **usługi sieciowej** i opiszemy sytuacje, w których usługi te odgrywają ważną rolę. Pokażemy prosty model, zwany architekturą zorientowaną na usługi, pozwalający uporządkować zastosowania technologii usług sieciowych. Przedstawimy także wzajemne relacje między różnymi technologiami, takimi jak: **SOAP** (*Simple Object Access Protocol*) oraz **WSDL** (*Web Services Description Language*) oraz **UDDI** (*Universal Description Discovery and Integration*) w formie trzech stosów zapewniających współdziałanie usług sieciowych.

W kolejnych rozdziałach zajmiemy się dokładnym omówieniem wprowadzonych tu pojęć.

Czym jest usługa sieciowa?

Trzymasz w rękach książkę opisującą budowanie usług internetowych. Jednak nie możemy Ci od razu powiedzieć, jak je tworzyć. Najpierw należy wyjaśnić, co rozumiemy pod pojęciem *usługi sieciowej*.

Termin usługi sieciowe zdobył dużą popularność w minionym roku. Wielu producentów oprogramowania (małych i dużych) podejmuje inicjatywę rozwoju i adopcji tej technologii w swoich produktach (zobacz ramka „Dynamika rynku usług sieciowych”). Różne organizacje są zaangażowane w rozwój standardów związanych z usługami sieciowymi. Chociaż można dostrzec rosnącą zgodność różnych interpretacji tego pojęcia, to nie istnieje jedna, ogólnie przyjęta definicja usługi sieciowej. Przypomina to sytuację z początków programowania obiektowego — nie zostało ono wchłonięte przez główny nurt metodologii tworzenia oprogramowania aż do chwili jednoznacznego zdefiniowania pojęć *dziedziczenia*, *enkapsulacji* i *polimorfizmu*.

Kilku głównych producentów platform dla usług sieciowych opublikowało własne definicje usługi. Definicja sformułowana przez firmę IBM jest zawarta w dokumencie <http://www4.ibm.com/software/solutions/Webservices/pdf/WSCA.pdf>.

„Usługa sieciowa jest interfejsem opisującym kolekcję operacji, do których, na podstawie standaryzowanych komunikatów w XML-u, istnieje dostęp przez sieć. Usługi sieciowe wykonują określone zadanie lub zestaw zadań. Usługa sieciowa jest opisana standardowym, formalnym dokumentem XML, zwanym opisem usługi, który zawiera wszystkie informacje potrzebne do interakcji z usługą, włącznie z opisem formatu komunikatów (dzięki którym są wykonywane operacje), protokołów transportowych i lokalizacji.

Interfejs z założenia ukrywa szczegóły implementacji usługi, dzięki czemu można z niej korzystać w ten sam sposób niezależnie od platformy sprzętowej i systemowej, na której usługa działa oraz od języka programowania, w jakim usługa została napisana. Takie podejście pozwala i zachęca do tego, aby aplikacje oparte na usługach sieciowych były luźno powiązane, zbudowane z komponentów i niezależne od wykorzystanych technologii. Usługi sieciowe mogą działać samodzielnie lub we współpracy z innymi usługami w celu przeprowadzenia złożonej operacji lub transakcji biznesowej”.

Microsoft podaje kilka definicji usługi sieciowej. Pierwszą z nich można znaleźć pod adresem <http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000442>:

„Usługa sieciowa jest jednostkowym elementem logiki aplikacji dostarczającym dane i usługi innym aplikacjom. Aplikacje komunikują się z usługami sieciowymi z wykorzystaniem popularnych w sieci Internet protokołów i formatów danych, jak: HTTP, SML i SOAP niezależnie od sposobu implementacji konkretnej usługi. Usługi sieciowe łączą najlepsze elementy programowania opartego na komponentach oraz sieci WWW i stanowią fundament modelu programistycznego Microsoft .NET”.

Dругa definicja Microsoftu jest dostępna pod adresem http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnWebsrv/html/Websvcs_plaform.asp:

„Usługa sieciowa jest programowalną logiką aplikacji dostępną poprzez standardowe protokoły sieci Internet. Usługi sieciowe łączą najlepsze elementy programowania działającego na podstawie komponentów oraz sieci WWW. Tak jak komponenty, usługi sieciowe dają funkcjonalność czarnej skrzynki, z której można korzystać, nie przejmując się wewnętrzną implementacją. W odróżnieniu od używanych do tej pory technologii komponentowych, dostęp do usług sieciowych nie jest realizowany przez protokoły specyficzne dla danego modelu obiektowego jak DCOM (*Distributed Component Object Model*), RMI (*Remote Method Invocation*) lub IIOP (*Internet Inter-ORB Protocol*). Komunikacja z usługami sieciowymi istnieje na podstawie popularnych w Internecie protokołów i formatów danych jak HTTP (*Hypertext Transfer Protocol*) i XML (*Extensible Markup Language*). Co więcej, interfejs usługi jest ściśle zdefiniowany przez akceptowane i generowane komunikaty. Aplikacje korzystające z usług sieciowych mogą być zaimplementowane w dowolnym języku programowania i na dowolnej platformie, o ile tylko są w stanie wysyłać i odbierać komunikaty w formacie zdefiniowanym przez interfejsu usługi”.

Sun podaje następującą definicję w dokumencie <http://www.sun.com/software/sunone/faq.html#2>:

„Usługi sieciowe są komponentami programowymi, które mogą być w dowolnym momencie odszukiwane i składane w grupy w celu odpowiedzi na problem (żądanie) użytkownika. Język Java™ oraz XML to doskonałe technologie do realizacji usług sieciowych”.

Jak widać, usługi sieciowe są rozumiane tak samo, nie istnieje jednak ustalona definicja tego pojęcia. Wielu programistów twierdzi, że nie może zdefiniować, czym jest usługa sieciowa, ale ci sami rozpoznają ją, gdy zobaczą.

Na potrzeby tej książki ustaliliśmy, że usługa sieciowa jest komponentem programowym **niezależnym od implementacji oraz platformy**. Komponent ten może być:

- **opisany** w języku opisu usług sieciowych;
- **opublikowany** w rejestrze usług;
- **odszukany** za pomocą standardowego mechanizmu (w czasie wykonania lub projektowania);
- **wywołany** poprzez zdefiniowany interfejs (API), zwykle zdalnie;
- **zgrupowany** z innymi usługami sieciowymi.

Należy pamiętać, że usługi sieciowe nie muszą koniecznie egzystować w sieci WWW, co mogłaby sugerować niefortunnie dobrana nazwa. Usługa sieciowa może być umieszczona w dowolnej sieci, Internecie lub sieci wewnętrznej. Niektóre usługi można wywołać poprzez zwyczajne wywołanie metody w obrębie tego samego procesu w systemie operacyjnym lub z wykorzystaniem pamięci dzielonej między powiązаныmi ze sobą procesami na jednej maszynie. W rzeczywistości usługi sieciowe mają niewiele wspólnego z siecią WWW zorientowaną na HTML i przeglądarki WWW. Czasami zdarza się, że nazwy wybierane w przemyśle informatycznym są pozbawione sensu — po prostu zaczynają żyć własnym życiem.

Nie można także zapomnieć, że dla programu korzystającego z usługi sieciowej szczegóły dotyczące implementacji oraz platformy, na której działa usługa, są nieistotne. Usługa jest dostępna poprzez zadeklarowany interfejs oraz mechanizm wywołania (protokół sieciowy, schematy kodowania danych itd.). Jest to sytuacja analogiczna do powiązania przeglądarki z serwerem WWW, pomiędzy którymi istnieje niewielki związek. Przeglądarka nie zwraca uwagi na to, czy ma do czynienia z serwerem Apache Tomcat, Microsoft IIS czy IBM Websphere. Wspólne wymagania są ograniczone do rozumienia protokołu HTTP oraz języka HTML, a także ograniczonego zestawu typów MIME. Podobnie serwer WWW — nie przejmuje się tym, jakiemu klientowi przesyła dane. Dzięki minimalnym powiązaniom między komponentami, usługi sieciowe mogą tworzyć system luźno związanych ze sobą komponentów.

Perspektywa biznesowa



Z biznesowego punktu widzenia technologia usług sieciowych daje przede wszystkim możliwość integracji — integracji aplikacji w przedsiębiorstwie lub integracji aplikacji między partnerami biznesowymi (np. w łańcuchu dostaw). Scenariusz przedstawiony

w tej książce (szczególnie w rozdziale 7.) ilustruje takie właśnie podejście. Integracja aplikacji pozwala na oszczędzenie czasu i kosztów przy odbieraniu zamówień, udzielaniu informacji o ich statusie, przetwarzaniu zleceń wysyłki itd. Ważną zaletą jest to, że integracja aplikacji jest możliwa bez ścisłego związywania się z pojedynczym partnerem biznesowym. W sytuacji gdy inny dostawca zaoferuje niższą cenę, korzystniejsze warunki dostawy lub lepszą gwarancję jakości, firmowy system obsługi zamówień może być łatwo skonfigurowany do współpracy z tym dostawcą; zmiana konfiguracji jest równie prosta jak wczytanie innej strony do przeglądarki internetowej. Z chwilą gdy usługi sieciowe oraz XML jako format dokumentów zyskują popularność, dynamiczna integracja partnerów biznesowych w takim stylu stanie się częstą praktyką.

Kiedy integracja systemów będzie tak łatwa, organizacja uzyska znacznie lepszy kontakt z dostawcami, klientami oraz innymi partnerami biznesowymi, czego wynikiem będzie oszczędność kosztów, elastyczność profilu przedsiębiorstwa, lepsza obsługa klientów, większa ich lojalność itd. Tak jak IT jest podstawą sprawnego działania organizacji, tak technologia usług sieciowych będzie fundamentalną dla integracji systemów — integracji aplikacji działających w wewnętrznej sieci firmy lub integracji systemów partnerów biznesowych przez Internet albo rozbudowane wirtualne sieci prywatne.

Z perspektywy biznesowej usługa sieciowa jest procesem biznesowym lub fragmentem takiego procesu udostępnianym przez sieć partnerom biznesowym wewnętrznym i (lub) zewnętrznym dla osiągnięcia celu biznesowego.

Perspektywa techniczna

Z technicznego punktu widzenia usługa internetowa jest po prostu kolekcją jednej lub kilku powiązanych ze sobą operacji dostępnych przez sieć i zdefiniowanych przez opis usługi. Na tym poziomie abstrakcji pomysł usług sieciowych nie jest niczym nowym. Z pomocą usług sieciowych specjaliści IT próbują rozwiązać podstawowy problem przetwarzania rozproszonego, który jest znany od lat — problem lokalizacji i dostępu do zdalnych systemów. Zasadnicza różnica polega na tym, że dzisiejsze stanowisko oparte jest na otwartych technologiach (XML i protokoły internetowe) oraz standardach, nad których kształtem czuwają konsorcja takie jak **W3C** (*World Wide Web Consortium*) , które kieruje rozwojem specyfikacji SOAP i WSDL. Co więcej, przyjęte rozwiązania zwykle bazują na **wyszukiwaniu na podstawie możliwości** , gdzie wyszukiwanie dotyczy usług danego typu, a nie pojedynczej usługi o danej nazwie lub identyfikatorze obiektu.

Wykorzystanie usług sieciowych

Nadrzędnym hasłem związanym z usługami sieciowymi jest integracja aplikacji. Usługi sieciowe to zbiór technologii umożliwiających dostęp do funkcjonalności biznesowej, jak np. przetwarzanie zamówień kupna. Zwykle funkcjonalność biznesowa jest już zaimplementowana w postaci starych systemów przetwarzania transakcji, istniejących aplikacji WWW, komponentów EJB itp. Technologia usług sieciowych polega na umożliwieniu dostępu do aplikacji oraz na ich integracji; nie jest to technologia implementacji.

Technologia usług sieciowych jest wykorzystywana przez organizacje w dwóch klasach zastosowań — integracji aplikacji korporacyjnych (EAI, *Enterprise Application Integration*) oraz integracji aplikacji partnerów biznesowych przez Internet (B2B, *Business-to-Business*). W każdej z tych kategorii mogą się pojawiać rozwiązania o różnym stopniu złożoności, począwszy od najprostszych, jak sprawdzenie numeru karty kredytowej, aż do systemów obsługi skomplikowanych transakcji biznesowych, w które jest zaangażowanych wiele stron, jak system obsługi dostaw i zamówień. Usługi internetowe mogą być wywołane przez zwykłe aplikacje na komputery PC, systemy mainframe, przeglądarki WWW, a nawet urządzenia przenośne, jak telefony komórkowe lub cyfrowe asystenty osobiste (PDA). Niezależnie od konkretnych aplikacji, usługi sieciowe będą wykorzystywane w integracji systemów. Zintegrowane, elastyczne i luźno powiązane systemy będą mogły być łatwo rekonfigurowane w celu dostosowania do zmian zachodzących w procesach biznesowych przedsiębiorstwa.

Integracja aplikacji korporacyjnych


Integracja aplikacji korporacyjnych cały czas jest obszarem, gdzie wielkie firmy konsultingowe podpisują wielomilionowe kontrakty na pomoc klientom w uporządkowaniu gąszczu aplikacji, które w zamierzeniu nigdy nie miały ze sobą współpracować.

Obecnie wiele systemów w przedsiębiorstwach funkcjonuje w postaci ogromnego, monolitycznego silosa aplikacji. Sama modyfikacja takich systemów jest w praktyce często niewykonalna, co dopiero mówić o ich integracji z innymi aplikacjami. Aplikacje te operują zwykle na własnych formatach danych, a czasami (ze względów historycznych, często związanych z wydajnością) korzystają nawet z własnych protokołów komunikacji. Co więcej, wiele systemów, szczególnie w dużych organizacjach, działa na różnych platformach. Zmuszenie systemów do kooperacji to prawdziwe wyzwanie. W przypadku wielu organizacji, głównie powstałych w wyniku fuzji kilku odrębnych wcześniej firm, koszty poniesione na integrację systemów mogą znacząco wpłynąć na kondycję finansową przedsiębiorstwa.

Usługi sieciowe to zbiór technologii stanowiących narzędzia, dzięki którym można „opakować” istniejące systemy informatyczne w usługi sieciowe i wówczas przeprowadzić integrację. Aplikacje napisane w dowolnych językach programowania i działające na dowolnych platformach mogą korzystać z aplikacji udostępnianych w formie usług sieciowych. Dzięki takiemu podejściu cała złożoność istniejących systemów może być ukryta za standardowymi protokołami na podstawie XML-a. Można również zrezygnować z projektów zakładających współpracę między parami aplikacji na rzecz, bazującej na usługach sieciowych, grupowej interakcji systemów. Można się spodziewać, że dzięki rozwojowi standardów, technologii i narzędzi umożliwiających wysokopoziomową współpracę aplikacji, wewnętrzna integracja systemów małych i dużych przedsiębiorstw na całym świecie stanie się łatwa, przez co będzie można elastycznie modelować procesy biznesowe, a także, jeżeli zajdzie taka potrzeba, wykorzystywać outsourcing usług.

W wielu firmach technologia usług sieciowych będzie po raz pierwszy wykorzystana do wewnętrznej integracji aplikacji, ponieważ jest to najważniejsze zadanie działu IT. Elastyczne systemy umożliwią stworzenie elastycznych modeli biznesowych. Elastyczne modele biznesowe pozwolą lepiej dostosowywać się do zmian zachodzących w środowisku biznesowym.

B2B

Kolejnym motorem rozwoju usług sieciowych jest nieustanna ewolucja strategii B2B. B2B polega na integracji systemów biznesowych dwu lub więcej firm w celu implementacji procesów biznesowych zachodzących między kilkoma partnerami, jak obsługa łańcucha dostaw. Niektórzy eksperci są zdania, że integracja łańcucha dostaw będzie kluczowym zastosowaniem usług sieciowych w wyniku standaryzacji popularnych formatów na podstawie XML-a, a związanych z procesami biznesowymi występującymi w łańcuchu dostaw. Aplikacje B2B mogą być elementarne, jak zautomatyzowane sprawdzanie poprawności numeru karty kredytowej lub bardzo skomplikowane, jak obsługa łańcucha dostaw dla firmy z listy Fortune 100. Wyzwania łączące się z budowaniem aplikacji B2B wraz z ogromnym potencjałem rynkowym takich systemów spowodowały szybki rozwój nowych technologii, dzięki którym w ciągu pięciu lat przenieśliśmy się ze świata aplikacji B2C (*Business-to-Consumer*)  do świata usług sieciowych i SOAP.

B2C, B2B a usługi sieciowe

Aplikacje dostępne online, bazujące na HTML-u, są udostępniane szerokiemu gronu osób. Klasycznym przykładem aplikacji WWW klasy B2C jest internetowa księgarnia Amazon. Korzystanie z niej polega na uruchomieniu przeglądarki, wczytaniu strony firmy, wprowadzeniu pewnych danych do formularzy, wysłaniu ich i otrzymaniu czytelnych wyników. Jedynym sposobem zautomatyzowania tego procesu jest symulacja czynności wykonywanych przez człowieka, co wymaga przeprowadzenia odwrotnej inżynierii aplikacji WWW, w celu poznania sposobu przesyłu danych między kolejnymi stronami, automatycznego przekazania informacji między tymi stronami i w końcu zinterpretowania odpowiedzi zwróconej w postaci dokumentu HTML. Takie podejście było popularne we pierwszych latach istnienia sieci WWW (1995 – 1997) i jest ono bardzo podatne na błędy. Dowolna zmiana w aplikacji — nawet taka, która dotyczy wyłącznie interfejsu użytkownika i nie zmienia przekazywanych danych — może spowodować wystąpienie błędów. Problemy powstają, gdyż w większości aplikacji ich logika nie jest dobrze oddzielona od sposobu pokazania danych. Jedynym sposobem integracji aplikacji sieciowych jest zastosowanie podejścia B2B.

Aplikacje B2B zasadniczo różnią się od aplikacji B2C, ponieważ są one projektowane pod kątem tego, aby ich klientami były inne aplikacje. Tabela 1.1 zawiera zestawienie różnic dla aplikacji napisanych w Javie. Obydwie klasy aplikacji są niezależne od wewnętrznych technologii, którymi najczęściej są zwykłe klasy Javy lub komponenty EJB (współdziałanie usług sieciowych z komponentami EJB jest omówione w rozdziale 5. — „Zastosowania SOAP w e-biznesie”). Tutaj jednak podobieństwa się kończą. Logika aplikacji B2C jest kontrolowana przez serwlety lub strony JSP (*Java Server Pages*) umieszczone w kontenerze serwletów. Podstawą aplikacji B2B jest zwyczajny kod Javy (często w postaci EJB) ulokowany w kontenerze usług sieciowych. Aplikacje B2C komunikują się z przeglądarką poprzez HTTP. Aplikacje B2B mogą wykorzystywać w tym celu dowolny otwarty protokół internetowy, jak HTTP, SMTP, FTP albo przemysłowe rozwiązania, jak EDI. Dane dla aplikacji B2C są przekazywane zwyczajnym protokołem HTTP. Dane wejściowe przychodzą w postaci parametrów żądania GET (zakodowane w identyfikatorze URL) lub parametrów żądania POST z formularzy. Przesyłane mogą być tylko ciągi znaków. Wszystkie inne typy danych, nawet liczby, muszą być zakodowane w formie

napisów. Dane wyjściowe są przemieszane ze znacznikami języka HTML na stronach. Aplikacje B2B — dla kontrastu — wykorzystują XML jako format wymiany danych. XML doskonale się sprawdza w zastosowaniach B2B, ponieważ jest niezależny od platformy i języka programowania, może w nim reprezentować dowolne struktury danych, łatwo go przetwarzać oraz można weryfikować poprawność dokumentu XML niezależnie od jego przetwarzania. Aplikacje B2C muszą mieć jakiś interfejs użytkownika (zwykle stworzony w HTML-u, chociaż niekiedy używa się apletów Javy), ponieważ ich odbiorcami są ludzie. Aplikacje B2B nie mają interfejsu użytkownika, ponieważ ich klientami są inne aplikacje.

Tabela 1.1. Porównanie napisanych w Javie aplikacji B2C i B2B

Obszar	Aplikacja B2C	Aplikacja B2B
Logika biznesowa	Klasy Javy i komponenty EJB	Klasy Javy i komponenty EJB
Mechanizm interakcji ze środowiskiem	Serwlety i strony JSP	Moduł obsługi usług sieciowych
Protokół komunikacyjny	HTTP	HTTP, SMTP, FTP, TCP/IPEDI, JMS, RMI/IOP...
Dane wejściowe	Parametry żądań GET/POST	XML
Dane wyjściowe	HTML	XML
Interfejs użytkownika	HTML + skrypty	brak
Klient	Człowiek korzystający z przeglądarki	Inna aplikacja

Trendy w e-biznesie


Jest jasne, że rozwój biznesu jest obecnie uwarunkowany rozwojem tzw. nowej ekonomii. Firmy muszą się dostosowywać do zmian zachodzących dynamicznie na rynku. W ciągu kilku ostatnich lat integracja aplikacji stała się głównym wyzwaniem przed jakim stanęły przedsiębiorstwa. Tradycyjne rozwiązania nie są już wystarczające, co wychodzi na jaw wraz ze wzrostem skali oraz liczby przeprowadzanych transakcji.

W ostatniej dekadzie współdziałanie komponentów heterogenicznych systemów rozproszonych było głównym zagadnieniem inżynierii oprogramowania, a w szczególności integracji aplikacji korporacyjnych. Niestety, wizja płynnej integracji pozostaje wciąż w sferze marzeń. Niedoskonałość istniejących architektur uniemożliwia realizację tego założenia. Niedoskonałość ta ma źródło w ścisłym powiązaniu komponentów systemów, co wprowadza zależności na każdym poziomie działania aplikacji. Jedną z najważniejszych lekcji, jaką odebraliśmy jako projektanci i architekci oprogramowania, jest informująca, że aplikacje powinny być w stanie automatycznie zlokalizować zasoby (programowe lub inne) wtedy gdy są one potrzebne, bez dodatkowej interwencji człowieka. Taka możliwość uwalnia pracowników od zajmowania się systemami IT i pozwala im skoncentrować się na klientach i właściwej działalności firmy. Projektanci systemów również mogą się dzięki temu skupić na implementacji logiki biznesowej oprogramowania, zamiast tracić czas na rozwiązywanie problemów technicznych ze współdziałaniem aplikacji. Koncepcja niejawnej, niewidocznej integracji jako głównej korzyści biznesowej

jest podstawowym czynnikiem skłaniającym do wykorzystania technologii usług sieciowych (bardziej niż jakiegokolwiek argumenty techniczne). Innymi słowy, nadszedł czas na integrację *just in time!*

Trendy w projektowaniu aplikacji przesuwają się od sztywnych struktur w kierunku elastycznych rozwiązań. Trendy we współpracy między partnerami biznesowymi przesuwają się od umów statycznych w kierunku dynamicznych. Trendy w integracji B2B — od integracji technologii w kierunku integracji procesów biznesowych. Podobne zmiany zachodzą w modelach programistycznych i architektonicznych, co umożliwi realizację tych trendów i przejście od ściśle powiązanych aplikacji do luźno powiązanych usług.

W dziedzinie technologii główne zmiany zaszły w zakresie zwiększenia uniwersalności oraz współdziałania aplikacji poprzez otwarte i szeroko akceptowane standardy. Pierwszy znaczący krok został wykonany dwie dekady temu w związku z uznaniem TCP/IP za otwartą platformę sieciową. Umożliwiło to rozwinięcie się ważnego i powszechnie spotykanego modelu przetwarzania klient-serwer. Kolejny krok to pojawienie się WWW wraz z HTTP i językiem HTML, które razem stanowiły pierwszy naprawdę uniwersalny, otwarty i przenośny interfejs użytkownika. Następnie Java stała się prawdziwie otwartym i przenośnym językiem programowania, a ostatecznie uzupełnił ją XML, wnosząc otwarty i przenośny standard wymiany danych. Kolejnym krokiem w ewolucji tych standardów jest integracja. W jaki sposób wszystkie te składniki łączą się, aby wspomóc ewolucję e-biznesu? Odpowiedzią są usługi sieciowe.

Odejście od mechanizmu zdalnego wywołania procedur (RPC, *Remote Procedure Call*) w stronę modelu przetwarzania opartego na przekazywaniu komunikatów lub **ukierunkowanego na dokumenty**  odzwierciedla pewien aspekt luźno powiązanych systemów. W podejściu zorientowanym na przetwarzanie dokumentów interfejs usługi sieciowej staje się o wiele prostszy i bardziej elastyczny. Interfejs RPC, wymagający przekazywania ustalonego zbioru parametrów w zadanej kolejności, jest dość niewygodny. Wprowadzenie nieznacznych zmian w strukturze informacji, np. dodanie pola określającego datę ważności karty kredytowej, powoduje konieczność utworzenia nowego interfejsu, opublikowania go oraz zrozumienia przez klienta usługi. W podejściu bazującym na przetwarzaniu dokumentów nowa informacja może być dodana do schematu dokumentu zdefiniowanego przez interfejs usługi sieciowej. Programy korzystające ze starego schematu niekoniecznie będą miały problemy po dodaniu nowego elementu XML (jest to cecha przestrzeni nazw XML, które są omówione w rozdziale 2. „Elementarz XML-a”). Z tego punktu widzenia interfejsy usług sieciowych są bardziej elastyczne, co pozwala tworzyć systemy o lepszych możliwościach adaptacyjnych.

Dlaczego potrzebujemy usług sieciowych?

Na początku tego rozdziału powiedzieliśmy, że motywacją do opracowywania technologii komunikacji przez Internet między aplikacjami jest rozwiązanie problemów, przed jakimi stoi obecnie przetwarzanie rozproszone, co w szczególności dotyczy integracji B2B. Od 1999 r. następuje szybki rozwój technologii usług sieciowych bazującej na XML-u, co ma być sposobem na realizację tych wyzwań. W gąszczu artykułów prasowych, nowych edycji produktów oraz ogłaszanych standardów wielu ludzi zastanawia się, czy usługi sieciowe to właściwa droga. W końcu dysponujemy już wieloma mechanizmami

Dynamika rynku usług sieciowych

Większa część dużych producentów oprogramowania oraz wiele małych firm IT zaakceptowało ideę usług sieciowych w takiej czy innej formie. Niektóre z nich mogą popierać tę technologię jedynie werbalnie, zastanawiając się, czy to tylko chwilowy trend i wykorzystując modną nazwę w celach czysto marketingowych. Inne firmy postanowiły, że ich przyszłość będzie rozwijała się na podstawie technologii usług sieciowych. Oto krótkie przedstawienie inicjatyw związanych z rozwojem usług sieciowych podjętych przez kilku głównych graczy na arenie IT:

- **IBM: Dynamiczny e-biznes** — IBM dostarcza bogaty zbiór technologii włącznie ze stosem SOAP wchodzącym w skład WebSphere'a (wywodzącym się z Apache SOAP 2.2), narzędziami WSDL-a w produkcie *Web Services Toolkit* oraz implementacją UDDI. Wiele dużych produktów IBM-u wykorzystuje w jakimś stopniu rozwiązania oparte na usługach sieciowych.
- **Microsoft: .NET** — Można powiedzieć, że Microsoft postawił na sukces platformy .NET. Choć .NET bazuje na technologii usług sieciowych, to cała inicjatywa jest o wiele szersza i wprowadza nowy język programowania (C#) oraz wspólną platformę wykonania programów, na której można budować implementacje różnych języków programowania. Technologii .NET przyjrzymy się z bliska w rozdziale 8.
- **Sun: SunOne (*Open Net Environment*)** — Sun wprowadził pojęcie **inteligentnych usług sieciowych** (*Smart Web services*), które są w stanie zrozumieć kontekst, w jakim zostały wdrożone lub wywołane (np. tożsamość użytkownika, typ urządzenia, którym posługuje się klient, politykę poufności itd.). W technologii Suna istnieje standard definiujący sposób dzielenia kontekstu oraz architektura *SunONE*, w której można wdrażać takie rozwiązania. Podejście firmy Sun do usług sieciowych jest podobne do podejścia firm konkurencyjnych w tym, że Sun opiera się na standardach XML, SOAP, WSDL i UDDI. Poszerza również te technologie poprzez wprowadzanie rozwiązań wywodzących się ze standardu ebXML. Szczegóły o sposobie zespolenia tych technologii nie są jasne. Jednym z istotnych elementów inicjatywy rozwoju usług sieciowych jest sponsorowanie przez firmę programu *Java Community Process* oraz tworzenie fragmentów specyfikacji języka Java dotyczących usług sieciowych.
- **Oracle: Oracle 9i Web Services Broker** — Nastawienie firmy Oracle do usług sieciowych także bazuje na standardach SOAP, WSDL i UDDI. Oracle podkreśla rolę swej technologii bazodanowej jako rejestru usług (brokera) zapewniającego bezpieczeństwo oraz inne dodatkowe funkcje poprzez pośrednictwo między klientem a dostawcą usługi.
- **Macromedia: Macromedia platform** — Firma Macromedia włączyła technologię usług sieciowych do swojej platformy do tworzenia aplikacji dla przedsiębiorstw. Jej aplikacje klienckie mogą wyświetlać dane pobrane z usług internetowych. Serwery aplikacyjne umożliwiają tworzenie usług przez programistów na dowolnym poziomie zaawansowania, a dostarczone narzędzia ułatwiają budowanie aplikacji wykorzystujących technologię usług sieciowych.

Fakt włączenia się wielu dużych producentów oprogramowania w rozwój usług sieciowych jest ekscytujący. Wiąże się to jednak, niestety, z ryzykiem niezgodności implementacji. Jeśli usługi sieciowe pochodzące od różnych producentów nie będą potrafiły kooperować, technologia nie zostanie szeroko zaaprobowana. Na szczęście firmy poświęcają dużo wysiłku temu, aby nie popełnić błędu niezgodności implementacji.

W rozdziale 8. przyjrzymy się kilku istniejącym implementacjom infrastruktury usług sieciowych, począwszy od produktów wielkich firm, aż do małych producentów oprogramowania.

rozproszonego przetwarzania informacji, z których niektóre można by rozwinąć do tego stopnia, aby zaspokajały potrzeby e-biznesu. Po co budować nowy stos technologii na fundamencie usług sieciowych?

Jest to bardzo dobre pytanie i trudno udzielić na nie zwięzłej odpowiedzi. „Ponieważ usługi sieciowe wykorzystują XML” nie jest właściwą odpowiedzią. To prawidłowa obserwacja, jednak nieodpowiadająca na najważniejsze pytanie, dlaczego wykorzystanie XML-a

wprowadza tak zasadniczą zmianę. Istnieją trzy główne powody, dla których obecne rozwiązania przetwarzania rozproszonego są gorsze od technologii usług sieciowych w rozwiązywaniu problemów e-biznesowych:

- Zakres rozwiązywanych problemów.
- Wybór dostępnych technologii.
- Dynamika powstawania nowych standardów i rozwiązań technicznych.

Zakres problemu

Tradycyjne mechanizmy przetwarzania rozproszonego ewoluowały zwykle od architektur technicznych, a u ich podstaw nie leżały problemy integracji aplikacji, np. technologia CORBA została opracowana jako rozwiązanie problemu implementacji złożonych, rozproszonych systemów obiektowych. Przyjmowano wówczas, że jest to właściwe podejście do organizowania komunikacji między aplikacjami. Jak powiedzieliśmy wcześniej, mechanizm RPC zwykle nie sprawdza się najlepiej w takich zastosowaniach. Zapotrzebowanie na luźno powiązane aplikacje oraz automatyzację procesów biznesowych ujawniło zyski płynące z prostej wymiany komunikatów niosących dane (zwykle dokumenty biznesowe) między partnerami w transakcjach e-biznesowych, co jest nazywane podejściem ukierunkowanym na dokumenty. Specyfikacje dotyczące przetwarzania rozproszonego traktują wymianę komunikatów jako model przetwarzania, jednak RPC i wymiana komunikatów nigdy nie uzyskały równego stopnia ważności, aż do chwili nadejścia usług sieciowych.

Rzeczywisty rozwój usług sieciowych nie jest związany z żadną predefiniowaną architekturą, ale z problemem integracji aplikacji. Jest to bardzo istotne rozróżnienie. Wybór zakresu problemu określa zagadnienia, na jakich jest skupiony rozwój technologii. Technologie związane z usługami sieciowymi zostały zaprojektowane od podstaw pod kątem integracji aplikacji. W efekcie stwarza to możliwości wykraczające poza zakres standardowych technik przetwarzania rozproszonego:

- Pomoc dla RPC oraz przesyłania dokumentów.
- Transport zakodowanych informacji pochodzących z aplikacji oraz dokumentów biznesowych.
- Działanie oparte na otwartych protokołach internetowych, jak HTTP i SMTP.

Innymi słowy, usługi sieciowe lepiej nadają się do realizacji tych zadań niż technologie, którymi dysponowaliśmy do tej pory, ponieważ zostały zaprojektowane właśnie pod tym kątem. COM, CORBA, RMI to ciągle świetne technologie komunikacji między rozproszonymi obiektami w sieci korporacyjnej, jednak integracja aplikacji e-biznesowych pozostanie domeną usług sieciowych.

Rdzenne technologie

Ze względu na to, że usługi sieciowe są wykorzystywane przy rozwiązywaniu problemów o znacznie szerszym zakresie niż tradycyjne technologie rozproszone, opierają się na bardziej elastycznych rozwiązaniach. Co więcej, używając usług sieciowych, możemy wykorzystać całe nasze doświadczenie w zakresie łączenia oraz integracji aplikacji, jakie

zdobyliśmy podczas pracy z systemami rozproszonymi. Te dwa czynniki sprawiają, że usługi sieciowe oferują lepsze podstawy do rozwiązywania problemów e-biznesowych aniżeli tradycyjne techniki rozproszone.

W punkcie „Stosy technologii zapewniających współdziałanie usług sieciowych” zapoznamy Cię z tytułowym pojęciem. Stosy te definiują warstwowy podział technologii zapewniających funkcjonalność usług sieciowych. Dzięki temu można warstwa po warstwie porównać stanowisko wykorzystujące usługi sieciowe z tradycyjnymi metodami przetwarzania rozproszonego, aby przekonać się, dlaczego technologie, na których opierają się usługi sieciowe, są lepsze w rozwiązywaniu istniejących problemów. Zamiast przechodzić cały długi proces, skupmy się na dwóch kluczowych możliwościach — reprezentacji struktur danych oraz opisu tych struktur.

Kodowanie danych jest zasadniczym słabym punktem tradycyjnych podejść do przetwarzania rozproszonego, szczególnie tych, które są niezależne od języka programowania. Oczywiście zazwyczaj dysponują one mechanizmami do reprezentacji prostych typów danych (liczby, napisy, wartości logiczne, daty itd.), podstawowych tablic oraz struktur wraz z ich właściwościami, jednak odwzorowanie złożonych typów danych, na jakich operują aplikacje, jest bardzo trudne z pomocą tych narzędzi. Wprowadzenie dodatkowych, własnych typów danych było w praktyce niemożliwe, gdyż wymagałoby zmian w specyfikacji. Sprawy komplikuje jeszcze bardziej fakt binarnego kodowania danych, przez co aplikacje muszą się np. martwić o to, czy liczby są kodowane w standardzie z malejącym czy z rosnącym porządkiem bitów.

W technologii usług sieciowych dane są zapisywane w formacie XML. Tekstowy format dokumentów XML pozwala zapomnieć o problemach związanych z porządkiem bajtów, a szeroki dostęp narzędzi do przetwarzania XML-a umożliwia łatwe wkroczenie w świat usług sieciowych. Dzięki hierarchicznej strukturze XML-a (osiągniętej przez zagnieżdżanie elementów) można zmienić pewien zagnieżdżony fragment dokumentu, nie martwiąc się o wpływ tej zmiany na pozostałą treść. Siła ekspresji, jaką dają atrybuty oraz zagnieżdżanie elementów, pozwala na bardziej naturalną reprezentację złożonych struktur danych w XML-u niż w czysto binarnych formatach, tradycyjnie wykorzystywanych np. w technologiach COM i CORBA. Krótko mówiąc, XML ułatwia operowanie na danych dowolnych typów.

Wybór XML-a spowodował uwypatnienie kolejnej zalety usług sieciowych — możliwości opisu typu danych i późniejszej weryfikacji, czy otrzymane dane są zgodne ze specyfikacją. Wykorzystuje się do tego metajęzyki związane z XML-em, np. XML Schema. Binarne kodowanie danych używane dotychczas w architekturach rozproszonych nie oferowało żadnych takich mechanizmów walidacji, zrzucając odpowiedzialność za poprawność danych na programistę aplikacji, co bardzo komplikowało proces tworzenia oprogramowania.

Dynamika przemysłu

Impet jest bardzo ważnym elementem dynamiki rozwoju oprogramowania. Wielkie problemy otwierają drogę ku wielkim możliwościom. Chęć wykorzystania tych możliwości jest źródłem dynamicznego rozwoju inicjatyw podjętych w celu rozwiązania problemu. Ten właśnie impet jest główną siłą przemysłu. Dzięki niemu pojawiają się innowacyjne

rozwiązania na szeroką skalę. Wyzwanie stawiane przez integrację aplikacji e-biznesowych jest ogromne, dlatego właśnie wszyscy czołowi gracze branży IT obecnie pracują nad nim (porównaj punkt „Dynamika rynku usług sieciowych”). Potrzeby klientów, wymagania stawiane przez rynek oraz chęć dołączenia do elitarnej grupy firm zajmujących się najnowocześniejszymi technologiami skłoniły wiele przedsiębiorstw do głębokiego zaangażowania w usługi sieciowe. Należy się spodziewać wielu pozytywnych efektów. Zastanówmy się — ostatnio, gdy każdy z kluczowych producentów infrastruktury informatycznej pracował nad tym samym zbiorem zagadnień, były początki rozwoju e-biznesu, gdy powstawały rozwiązania służące budowaniu aplikacji WWW. W efekcie stworzono nowy model projektowania aplikacji, w którym wykorzystuje się przeglądarkę internetową jako uniwersalnego klienta oraz serwer aplikacyjny jako uniwersalny system zaplecza. Można więc mieć nadzieję, że wynikiem wspólnej pracy znamienitych umysłów, działających pod auspicjami takich organizacji jak W3C i OASIS, będzie stworzenie dobrego rozwiązania problemu integracji e-biznesowej.

Weterani przemysłu IT często utożsamiają wspomniany impet z rozdmuchaną reklamą. Czy chcemy więc powiedzieć, że usługi sieciowe odniosą sukces, ponieważ są tak rozreklamowane? Absolutnie nie! Niezwykle dynamiczny rozwój tej technologii jest faktem, a ona sama odbiega od dotychczasowych trendów w technologiach przetwarzania rozproszonego. Fundamentalna różnica polega na tym, że wiele firm z branży IT może się **jednocześnie** zaangażować w tworzenie uzupełniających się standardów.

Jednoczesne działanie jest kluczowe dla zapewnienia dobrej dynamiki rozwoju i powstawania innowacyjnych rozwiązań. W dotychczasowych przedsięwzięciach, skupiających się na opracowaniu technologii przetwarzania rozproszonego, nie można było osiągnąć takiego zrównowżenia, ponieważ były one prowadzone przez pojedynczego producenta (np. COM Microsoftu) albo przez dużą, powolną w działaniu organizację, jak OMG (*Object Management Group*), która jest autorem standardu CORBA. W obydwu przypadkach szybki postęp prac uniemożliwiałoby scentralizowane zarządzanie standardami. Każda zmiana musiała być aprobowana przez ciało posiadające prawo do danego standardu. Standardy COM i CORBA należały odpowiednio do Microsoftu i OMG. Taka sytuacja nie sprzyja osiągnięciu dużego tempa rozwoju, mimo wysokości budżetu przeznaczanego na promocję danej technologii. Producent, który odczuwa, że ma bardzo niewielki wpływ na rozwój technologii, prawdopodobnie nie poświęci się pracy nad nią. Innymi słowy, możesz korzystać z technologii COM, jeżeli jednak sądzisz, że nie masz szans na wywarcie wpływu na Microsoft w zakresie rozwoju COM-a, to raczej nie będziesz poświęcał wiele czasu na rozmyślanie, w jaki sposób można udoskonalić tę technologię. Projekty z otwartym dostępem do kodu źródłowego jak system operacyjny Linux i oprogramowanie tworzone przez Apache Software Foundation rozwijają się dynamicznie, ponieważ pracujący przy nich ludzie mają bezpośredni wpływ na kształt końcowego produktu. Usługi sieciowe także rozwijają się dynamicznie, ponieważ standardy są jednocześnie opracowywane przez W3C, OASIS, UDDI oraz wiele innych organizacji standardyzacyjnych. Co więcej, jak do tej pory najwięksi producenci chętnie angażują się w prowadzenie otwartych projektów.

Z perspektywy technicznej jest interesujące, że XML w zasadzie wiąże się z możliwością równoległego prowadzenia procesu standaryzacji usług sieciowych. W XML-u istnieją pewne udogodnienia (przestrzenie nazw i schematy), dzięki którym ewolucja standardów bazujących na XML-u może zachodzić w sposób zdecentralizowany, bez

późniejszych problemów z łączeniem częściowych rozwiązań w całość. Jeżeli na przykład grupa A jest właścicielem pewnego standardu, a grupa B próbuje stworzyć jego rozszerzenie, to grupa B może to zrobić w następujący sposób:

- Rozszerzenie może być opublikowane niezależnie od samego standardu.
- Rozszerzenie może istnieć obok standardowych mechanizmów.
- Aplikacje, które nie są w stanie zinterpretować rozszerzenia, nie będą działać błędnie w jego obecności.
- Aplikacje, które wymagają rozszerzenia, nie będą działać bez niego.

W pracach nad technologią usług sieciowych połączono właściwy zakres problemu (integracja aplikacji e-biznesowych) z właściwymi technologiami (standardy na podstawie XML-a) oraz możliwością równoległego działania i wprowadzania innowacji. Z tego właśnie powodu usługi sieciowe osiągną sukces.

Historia przetwarzania rozproszonego

Przetwarzanie rozproszone skupiało się na zagadnieniu rozłożenia obliczeń pomiędzy różne systemy, które wspólnie pracowały nad rozwiązaniem danego problemu. Najczęściej używaną abstrakcją przetwarzania rozproszonego był RPC. Mechanizm zdalnego wywołania procedur pozwalał na wywołanie zdalnej funkcji w taki sam sposób jak lokalnej. Rozproszone systemy obiektowe wymagały mechanizmu RPC zorientowanego na obiekty (ORPC). W takiej architekturze potrzebny był dodatkowy kontekst, dzięki któremu można wywołać metody na rzecz konkretnych instancji obiektów. Historia RPC oraz rozproszonych systemów obiektowych jest dość skomplikowana. Poniższe zestawienie zawiera niektóre z najważniejszych wydarzeń:

- 1987
 - Firma Sun Microsystems opracowała system *Open Network Computing* (ONC) na podstawie RPC jako podstawowy mechanizm komunikacyjny dla swego sieciowego systemu plików NFS (*Network File System*).
 - Firma Apollo Computer stworzyła system *Network Computing System* (NCS) na podstawie RPC na potrzeby systemu operacyjnego *Domain*.
- 1989
 - Organizacja *Open Software Foundation* (OSF, obecnie *The Open Group*) opublikowała dokument *Request for Technology* (RFT) dla systemu RPC. OSF otrzymała dwie propozycje. Pierwsza z nich, pochodząca od firmy HP/DEC, bazowała na systemie NCS (HP przejął Apollo). Kolejne rozwiązanie oparte na ONC przedstawił Sun. OSF wybrała NCS jako mechanizm RPC dla swojego projektu *Distributed Computing Environment* (DCE).
 - Powstała grupa *Object Management Group* (OMG) w celu opracowania specyfikacji definiujących (niezależną od platformy i języka) implementację technologii przetwarzania rozproszonego (w czasie pisania tej książki konsorcjum liczy około 650 członków). OMG zajęła się pracami nad specyfikacją rozproszonej architektury obiektowej o nazwie *Common Object Request Broker Architecture* (CORBA).
- 1990
 - Microsoft oparł swoje prace nad mechanizmem RPC na zmodyfikowanej wersji DCE/RPC.
- 1991
 - Organizacja OSF wprowadziła DCE 1.0.
 - Ukazał się standard CORBA 1.0 wraz z wiązaniem dla języka C. Zyskał popularność termin *Object Request Broker* (ORB) oznaczający oprogramowanie będące infrastrukturą dla budowy rozproszonych systemów obiektowych.

- 1996
 - Microsoft przedstawił architekturę *Distributed Component Object Model* (DCOM), która była ściśle związana z wcześniejszymi pracami Microsoftu nad technologiami komponentowymi, jak *Object Linking and Embedding* (OLE), *COM* (znany również jako *OLE2*) oraz *ActiveX* (lekkie komponenty stosowane w aplikacjach WWW). Podstawowe funkcje oferowane przez DCOM bazują na technologii RPC opracowanej przez Microsoft. DCOM jest protokołem ORPC.
 - Ukazał się standard CORBA 2.0, w którym znacznie rozbudowano rozproszony model obliczeniowy oraz wprowadzono wysoko poziomowe usługi, z których mogły korzystać rozproszone obiekty. Częścią specyfikacji był protokół *Internet Inter-ORB Protocol* (IIOP). IIOP pozwala na współpracę wielu ORB-ów niezależnie od producenta oprogramowania. IIOP jest protokołem ORPC.
- 1997
 - Sun wypuścił pakiet JDK 1.1 zawierający mechanizm *Remote Method Invocation* (RMI). RMI definiuje model przetwarzania rozproszonego z wykorzystaniem obiektów Javy. RMI jest podobny do mechanizmów CORBA i DCOM, jednak działa tylko z obiektami języka Java. RMI jest oparty na protokole ORPC o nazwie *Java Remote Method Protocol* (JRMP).
 - Microsoft ogłosił powstanie COM+, następcy architektury DCOM. Możliwości COM+ zbliżyły go do modelu przetwarzania rozproszonego zdefiniowanego przez specyfikację CORBA.
- 1999
 - Firma Sun opublikowała specyfikację J2EE (*Java 2 Platform Enterprise Edition*). Na platformie Java 2 zintegrowano technologie RMI z IIOP, co umożliwiło współdziałanie systemów na podstawie Javy oraz architektury CORBA.
 - Specyfikacja *Simple Object Access Protocol* (SOAP) po raz pierwszy ujrzała światło dzienne. Narodziła się era usług sieciowych.

Chociaż RPC i rozproszone obiekty to tradycyjne podejścia do konstrukcji systemów rozproszonych, nie są one oczywiście jedynymi. Inne, bardzo ważne podejście bazuje na przekazywaniu komunikatów niosących dane lub dokumenty. Zamiast skupiać się na rozproszonych obliczeniach poprzez odpowiednie wywołanie zdalnych fragmentów kodu, w modelu przekazywania komunikatów przyjęto odmienne podejście: komunikujące się aplikacje wykonują niezależnie od siebie obliczenia i wymieniają się komunikatami zawierającymi tylko dane. Zostało to spopularyzowane przez integratorów systemów, którzy próbowali doprowadzić do kooperacji między wysoce heterogenicznymi systemami. W większości systemy te były tak różne, że nie można było zrealizować wymagania precyzyjnej integracji za pomocą mechanizmu RPC. Zamiast tego osiągnięto sukces poprzez przesyłanie samych danych pomiędzy systemami. Komercyjne znaczenie aplikacji opartych na przekazywaniu komunikatów niezmiennie wzrasta od momentu, gdy IBM w 1993 r. opracował produkt MQSeries. Odpowiednikiem tego systemu z Microsoftu jest *Microsoft Message Queuing Server* (MSMQ). Specyfikacja J2EE definiuje zbiór interfejsów programistycznych zebranych pod nazwą *Java Messaging Service* (JMS). Nie została podjęta żadna próba zdefiniowania standardowego protokołu komunikacji między serwerami komunikatów.

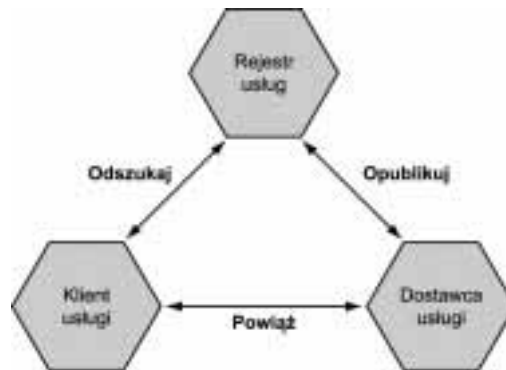
Jedną z podstawowych zalet technologii usług sieciowych jest to, że protokoły, na których bazuje, obsługują równie dobrze modele RPC oraz modele przekazywania komunikatów. W rozdziale 3. znajduje się podrozdział poświęcony temu właśnie zagadnieniu.

Architektury zorientowane na usługi

Na wczesnym etapie ewolucji technologii usług sieciowych zauważyliśmy pewną prawidłowość. Występowała ona za każdym razem, gdy wykorzystywaliśmy usługi sieciowe w integracji aplikacji. Nazwaliśmy ten wzorzec architekturą zorientowaną na usługi

(SOA, *Service Oriented Architecture*). Jest to prosta koncepcja, którą można zastosować w wielu sytuacjach wykorzystania usług internetowych. Na rysunku 1.1 przedstawiono główne role oraz operacje wykonywane w architekturze SOA.

Rysunek 1.1.
Architektura
zorientowana
na usługi



W architekturze zorientowanej na usługi można zawsze wyróżnić trzy role — **klienta usługi** 📖, **dostawcę usługi** 📖 oraz **rejestr usług** 📖:

- Dostawca usługi odpowiada za przygotowanie **opisu usługi** 📖, opublikowanie go w jednym rejestrze lub więcej oraz odbieranie komunikatów wywołujących usługę od jednego klienta lub większej liczby klientów. Dostawcą usługi może więc być dowolna firma, która udostępni usługę w sieci. Możesz traktować dostawcę usługi jako serwer w powiązaniu klient-serwer między klientem usługi a jej dostawcą.
- Zadaniem klienta usługi jest pobranie opisu usługi z rejestru i wykorzystanie go w celu nawiązania połączenia i wywołania usługi udostępnionej przez dostawcę. Każdy, kto wykorzystuje funkcjonalność usługi sieciowej może być uważany za klienta tej usługi. Klienta usługi można traktować jako klienta w powiązaniu klient-serwer między klientem usługi a jej dostawcą.
- Rejestr usług służy do przechowywania, przeszukiwania i udostępniania opisów usług oferowanych przez różnych dostawców. Rolą rejestru usług jest kojarzenie klienta usługi z dostawcą usługi. Po sparowaniu klienta z dostawcą, rejestr nie jest już potrzebny na obrazku — pozostała część interakcji zachodzi bezpośrednio między klientem usługi a jej dostawcą.

W każdej z wymienionych ról może występować dowolny program lub węzeł sieciowy. W pewnych okolicznościach ta sama aplikacja może pełnić kilka funkcji, na przykład dostawcy usług dla końcowych odbiorców oraz klienta usług dostarczonych przez innych.

W architekturze SOA pojawiają się także trzy operacje — **opublikuj** 📖, **odszukaj** 📖 oraz **powiąż** 📖. Operacje te definiują kontrakty między rolami SOA:

- Opublikowanie usługi polega na jej zarejestrowaniu lub, innymi słowy, ogłoszeniu jej istnienia. Jest to rodzaj kontraktu między rejestrem usług a dostawcą. Kiedy dostawca usługi umieszcza jej opis w rejestrze usług, wszystkim potencjalnym klientom tej usługi udziela szczegółowych informacji na temat jej funkcji. Szczegóły interfejsu służącego do publikacji zależą od implementacji rejestru usług. W najprostszym przypadku rola rejestru usług jest odgrywana przez samą sieć,

a opublikowanie polega po prostu na umieszczeniu opisu usługi w drzewie katalogów serwera aplikacji WWW. W przypadku innych implementacji rejestru usług, jak np. UDDI, zdefiniowany jest bardzo wyrafinowany interfejs do publikacji.

- Odszukiwanie jest niemal lustrzanym odbiciem operacji publikowania. Jest to kontrakt pomiędzy klientem usługi a rejestrem usług. Za pomocą tej operacji klient usługi określa kryteria wyszukiwania, jak typ usługi, inne jej aspekty, jak gwarancja jakości usługi itd. Rejestr usług wyszukuje wśród przechowywanych opisów wszystkie usługi spełniające podane kryteria. Możliwości konfiguracji wyszukiwania zależą oczywiście od implementacji rejestru usług. Najprostsze rejestry mogą np. oferować jedynie wyszukiwanie oparte na bezparametrowym żądaniu HTTP GET. Oznacza to, że operacja wyszukiwania zawsze zwróci w wyniku opisy wszystkich opublikowanych usług i zadanie wybrania odpowiedniej usługi spadnie na klienta. Oczywiście UDDI oferuje potężne możliwości w zakresie wyszukiwania.
- Operacja powiązania tworzy połączenie o charakterze klient-serwer między klientem usługi a jej dostawcą. Może ona być skomplikowana i przeprowadzana w sposób dynamiczny, połączona np. z wygenerowaniem na bieżąco pośrednika po stronie klienta na podstawie opisu usługi, który będzie wykorzystywany do wywołania usługi. Można również wykorzystywać model statyczny, gdzie programista na stałe zapisuje w programie sposób wywołania usługi sieciowej.

Podstawowym elementem architektury SOA jest opis usługi, który jest publikowany przez dostawcę usługi w rejestrze usług. To właśnie ten opis jest pobierany przez klienta jako wynik operacji wyszukiwania. To opis usługi przekazuje klientowi wszystkie informacje, jakich potrzebuje do tego, aby połączyć się i wywołać funkcje usługi. Opis usługi zawiera również informacje na temat wartości zwracanych w wyniku wywołania funkcji usługi.

W różnych przypadkach wdrażania architektury zorientowanej na usługi w każdej roli można wykorzystywać różne technologie. W rozdziale 7. omawiamy kilka możliwych implementacji rejestru usług i przedstawiamy szczegółowo technologię UDDI. Rozdział 6. dotyczy opisów usług oraz ich wykorzystania w automatycznym tworzeniu pośrednika usługi sieciowej po stronie klienta oraz szkieletu po stronie serwera służącego do przekazywania żądań do usługi. W rozdziale 3. i 4. skupiamy się na użyciu protokołu SOAP w operacji wiązania, natomiast rozdział 5. dotyczy przystosowania operacji wiązania do potrzeb e-biznesowych.

Stosy technologii zapewniających współdziałanie usług sieciowych

Usługi sieciowe są otoczone obszernym zestawem technologii: XML, SOAP, WSDL, UDDI, WSEL, WSFL i inne. Jak można się zorientować w tym, czym one są i jak do siebie pasują? Wyjaśnienie tego jest jednym z celów niniejszej książki.

Aby wkomponować te technologie w pewne ramy, odniesiemy się do trójki stosów. Ten podział został po raz pierwszy zaproponowany przez W3C, IBM i Microsoft w marcu 2001 r. (<http://www.w3.org/2001/03/WSWS-popa/paper51>). Technologie związane z usługami sieciowymi zostały zgrupowane w postaci trzech stosów:

- Stos połączenia.
- Stos opisu.
- Stos wyszukiwania.

Zawartość stosów przedstawiona w tej książce pokazuje podział nieco odbiegający od oryginalnie zaproponowanego przez W3C (częściowo ze względu na rozwój standardów od marca 2001 r.).

Stos połączenia

Rysunek 1.2 przedstawia stos połączenia.


Rysunek 1.2.
Stos połączenia



W stosie połączenia są zgrupowane technologie definiujące sposób przesłania komunikatu od klienta usługi do jej dostawcy. Na samym dole stosu znajduje się protokół sieciowy. Usługi sieciowe mogą wykorzystywać różne standardowe protokoły Internetu, jak HTTP, HTTPS, SMTP, FTP itd., a także wyrafinowane protokoły spotykane w systemach korporacyjnych, jak RMI/IIOP i MQSeries.

Dane są kodowane w XML-u. Istnieje także możliwość tworzenia w komunikatach odwołań do danych w innym formacie, co daje całkowitą elastyczność w zakresie typów danych występujących w komunikatach. W technologii usług internetowych specyfikacja rodzaju danych jest zapisywana w XML Schema. Dotyczy to zarówno własnych schematów użytkownika, w przypadku przesyłania dokumentów XML, jak i predefiniowanych schematów opisujących np. zasady kodowania w protokole SOAP, czym zajmujemy się w rozdziale 3.

Powyżej warstw protokołu sieciowego oraz schematu kodowania danych znajduje się warstwa komunikatów XML. Usługi sieciowe wykorzystują tu standard SOAP we wszystkich jego odmianach dotyczących kodowania danych, stylu interakcji oraz wiązania do protokołów. SOAP służy do umieszczania dokumentów XML w kopertach. Wszystko razem stanowi opartą na standardach, solidną podstawę architektury usług sieciowych.

Kolejną warstwą umieszczoną koncepcyjnie jeden poziom ponad mechanizmem SOAP opakowywania komunikatów jest mechanizm wprowadzający rozszerzenia do zwykłych kopert pod nazwą **nagłówków SOAP** (*SOAP headers*)  Dzięki nagłówkom SOAP ortogonalne rozszerzenia, jak np. podpis elektroniczny, można związać z treścią komunikatu przesyłanego w kopercie SOAP. W rozdziale 3. szczegółowo omawiamy mechanizm SOAP oraz funkcjonalność nagłówków.

Warstwy tego stosu są dobrze zdefiniowane, czy to przez standardowe protokoły sieciowe, czy też przez samą specyfikację SOAP. Stos ten grupuje zbiór najszerzej akceptowanych i najlepiej rozwiniętych technologii związanych z usługami sieciowymi.

Po prawej stronie rysunku 1.2 znajdują się trzy pionowe kolumny reprezentujące stowarzyszone technologie mające wpływ na różne warstwy stosu połączenia. Na przykład bezpieczeństwo może się pojawiać na każdym poziomie, np. SSL na poziomie protokołu sieciowego oraz podpis elektroniczny na poziomie rozszerzeń SOAP. Pojawienie się jednego standardu obejmującego wszystkie kwestie bezpieczeństwa usług sieciowych jest wątpliwe. Rozdział 5. zawiera szczegółowe informacje w kontekście obecnych technologii związanych z usługami sieciowymi, a dotyczące podpisów elektronicznych w formacie XML oraz kryptografii XML. Pozostałe kolumny zostały opisane jako Jakość usług oraz Zarządzanie. Jest to tylko garść aspektów, które mogą się pojawiać na różnych poziomach stosu połączenia. Jeszcze nie ma dla nich ustalonych standardów, ale prace trwają.

Stos opisu

Stos połączenie dotyka tylko podstawowych możliwości usług sieciowych. Nawet w najprostszych przypadkach użycia usług sieciowych widać potrzebę możliwości współpracy na wyższym poziomie.

Rozważmy następującą sytuację (dokładnie prześledzimy ten przykład w rozdziale 3.). Firma dostarczyła usługi sprawdzania zaopatrzenia, dzięki czemu klienci mogą sprawdzać, czy w magazynie znajduje się określona liczba produktów o danym numerze. Parametrami wywołania usługi sieciowej są napis reprezentujący numer urządzenia oraz liczba całkowita oznaczająca liczbę sztuk, na jakie jest zapotrzebowanie. Jeżeli firma dysponuje pożądaną liczbą produktów, usługa zwraca wartość logiczną `true`; w przeciwnym przypadku wynikiem jest `false`.

Z punktu widzenia protokołu SOAP interakcja z usługą jest łatwa, jednak sprawy się komplikują, gdy weźmiemy pod uwagę liczbę informacji, jaka musi być dzielona przez klienta i dostawcę usługi. Minimalny zasób danych, którymi musi dysponować klient usługi, to:

- Adres usługi sieciowej.
- Wiedza o tym, że komunikaty będą przekazywane po HTTP.
- Wiedza o tym, że należy korzystać z SOAP 1.1.
- Wiedza o tym, że dane powinny być kodowane w standardzie SOAP.
- Informacja, że żądania powinny mieć postać wywołań RPC z dwoma parametrami typu napisowego i liczbowego oznaczającymi odpowiednio numer produktu i liczbę sztuk.
- Informacja, że odpowiedzi będą efektem wywołań RPC o typie logicznym i będą oznaczały wynik sprawdzenia podanego warunku.

Dodajmy do tego bezpieczeństwo, opłaty, obsługę błędów oraz inne funkcje konieczne do budowy poważnych systemów opartych na usługach sieciowych, a zobaczymy, że wymagany zasób wspólnych informacji jest jeszcze większy. W jaki sposób klient usługi

może zdobyć te informacje? Tradycyjnie, usługi sieciowe publikowały opis swoich możliwości w formie dokumentów czytelnych dla człowieka. Projektanci systemów czytali te opisy i konfigurowali aplikacje klienckie pod kątem komunikacji ze wskazanymi usługami.

Takie rozwiązanie może działać, ale z wielu powodów nie jest skalowalne:

- Wymaga kosztownej (pod względem czasowym i pieniężnym), ręcznej konfiguracji przez wyszkolony, a więc nieliczny personel.
- Jest narażone na błędy, ponieważ nie wykorzystuje formalnych specyfikacji usług.
- Wyklucza automatyczne wyszukiwanie i korzystanie z usług sieciowych — konfiguracja aplikacji klienckiej wymaga a priori wiedzy o dostępnych usługach.
- Nie istnieją żadne mechanizmy informowania o zmianach oraz odtwarzania po awarii; za każdym razem, gdy usługa się w jakiś sposób zmienia, istnieje ryzyko, że w aplikacjach klienckich pojawią się błędy.

Z tych właśnie powodów liderzy przemysłu IT opracowują standardy wchodzące w skład stosu opisu. Rysunek 1.3 przedstawia stos opisu usługi.

Rysunek 1.3.
Stos opisu usługi





Kluczowym elementem architektury zorientowanej na usługi jest sam opis usługi. Publikowana informacja powinna opisywać różne aspekty usługi sieciowej, dlatego stos opisu ma kilka poziomów. Głównym celem tworzenia opisu usługi jest to, żeby dostarczyć klientowi informację o tych cechach usługi, które są dla niego istotne. W rozdziale 6. dokładnie omawiamy każdą z technologii opisu usług sieciowych.

W świecie usług sieciowych podstawowym formatem opisu jest XML. XML Schema jest formalizmem służącym do definiowania typów danych zawartych w opisie, a wszystkie technologie opisu usług umieszczone na stosie posługują się XML-em. Usługi sieciowe zawdzięczają wiele ze swej funkcjonalności metajęzykowi XML.

Następne dwie warstwy na stosie — implementacja usługi oraz interfejs usługi — są opisane w języku WSDL (*Web Services Description Language*). Specyfikacja WSDL została przekazana W3C jako zgłoszenie zapotrzebowania na nowy standard i obecnie trwają prace zmierzające w kierunku standaryzacji tego języka. WSDL jest językiem definiowania interfejsów usług sieciowych. Zrozumienie tego jest nieodzowne do zrozumienia idei całej technologii. Za pomocą WSDL-a projektant opisuje zestaw operacji, jakie może wykonać

usługa, włącznie z typem obiektów przekazywanych jako parametry wejściowe i wyjściowe tych operacji oraz schematami wiązania do protokołów sieciowych i schematami kodowania danych. Na tym poziomie zdefiniowany jest interfejs usługi. Implementacja usługi określa adres w sieci, pod jakim znajduje się usługa. Dokument WSDL pozwala automatycznie wygenerować kod klienta usługi sieciowej, bazując na informacji zawartej w jego treści.

Sam język definicji interfejsów to jednak za mało. Na decyzję klienta o użyciu danej usługi sieciowej mają wpływ jeszcze inne aspekty. Jeżeli dwóch różnych dostawców oferuje implementacje usługi o tym samym interfejsie, np. serwis giełdowy, to z punktu widzenia opisu interfejsu usługi są prawie nierozróżnialne — mają tylko inne adresy sieciowe. Dostawcy mogą jednak dawać zupełnie różne warunki w zakresie polityki bezpieczeństwa, kosztów użycia, czasu odpowiedzi itd. Taka nieoperacyjna charakterystyka może mieć wielkie znaczenie dla klienta usługi. Rolą definicji węzła końcowego jest nałożenie na opis WSDL kolejnej warstwy informacyjnej dotyczącej tych cech usługi, na które ma wpływ jej środowisko implementacji. Prace na tym polu są dopiero w fazie początkowej, a nazwa języka **WSEL** (*Web Services Endpoint Language*)  jest jeszcze mało znana. Innym przykładem tworzenia tego rodzaju opisów może być związana ze standardem ebXML specyfikacja **CPP** (*Collaboration-Protocol Profile and Agreement Specification*).

Na szczycie stosu opisu usługi znajduje się warstwa stanowiąca cichą obietnicę niezauważalnej, automatycznej integracji usług — warstwa komponowanie usług. Definiując **komponowanie usług** , projektant określa sposób współdziałania grupy usług w celu realizacji zaawansowanych funkcji. Opis komponowania usług może na przykład dotyczyć współpracy usługi wysyłania zamówień kupna, usługi powiadamiania oraz obsługi zwrotów dającej w efekcie implementację złożonego serwisu obsługi zamówień. Na tym poziomie istnieje wiele różnych stanowisk, jak rozwiązać problem. IBM zaproponował język **WSFL** (*Web Services Flow Language*), a Microsoft opracował *Xlang*. Nie powstał tutaj jeszcze żaden formalny standard.

Komponowanie usług sieciowych stawia poważne wyzwania zarówno natury technicznej, jak i biznesowej. Po pierwsze, niewidoczna integracja usług wymaga opracowania zaawansowanych technologii. Najważniejszy jest opis działania usługi, zdefiniowany poprzez podanie reguł tworzenia sekwencji wywołań oraz reguł dotyczących wysyłania i odbierania komunikatów. Następnym pojawiającym się problemem jest odwzorowywanie procesów biznesowych na interakcje między usługami. Dodatkowym utrudnieniem jest tu wymaganie, żeby niektóre powiązania były tworzone w czasie wykonania. Bez tych możliwości trudno jest zaimplementować typowe procesy biznesowe, jak występowanie w imieniu klienta, świadczenie usług brokerskich oraz pośrednictwo w usługach finansowych. Z perspektywy biznesowej problemy wcale nie są mniejsze. Integracja usług jest problemem przepływu pracy i w związku z tym mogłaby wprowadzać zależności między różnymi aspektami modeli biznesowych przedsiębiorstw. Z tego powodu najtrudniejsza do przeprowadzenia jest integracja niosąca potencjalnie największe korzyści, czyli taka, która przekracza granice przedsiębiorstwa.




Stos wyszukiwania

Umiejąc opisywać usługi sieciowe, jesteśmy w o wiele bardziej komfortowej sytuacji niż wcześniej, ale ciągle rozwiązaliśmy tylko część problemu integracji usług sieciowych. Opis usługi podaje informację o tym, jak się z nią połączyć i jak z niej korzystać. Ale w jaki sposób mielibyśmy zdobyć ten opis? Jest oczywiste, że potrzebujemy jakiegoś mechanizmu wyszukiwania usług sieciowych. Wymagany jest zatem katalog lub wyszukiwarka usług. Dostawcy usług będą potrzebowali mechanizmu publikacji, żeby byli w stanie udostępniać informację o oferowanych przez siebie usługach i modyfikować ją w trakcie ewoluowania usług. Klienci usług będą potrzebowali do wyszukiwania dobrze zdefiniowanego interfejsu. To właśnie rola opisanego wcześniej rejestru usług w architekturze zorientowanej na usługi.

Rysunek 1.4 przedstawia trzeci ze stosów technologii — stos wyszukiwania. Zawiera on technologie związane z wyszukiwaniem usług sieciowych.

Rysunek 1.4.
Stos wyszukiwania



Dolną warstwą na stosie jest poziom inspekcji. **Inspekcja**  jest techniką odszukiwania opisu usługi na podstawie szczegółowych informacji dotyczących tej usługi (np. identyfikator usługi bądź URL). Znowu należy zaznaczyć, że nie istnieje tu żaden ustalony standard. IBM ma ADS , a Microsoft — DISCO .

Poziom katalogu reprezentuje funkcjonalność związaną z wyszukiwaniem usług sieciowych i partnerów biznesowych na podstawie opisu możliwości usługi. W odróżnieniu od wcześniejszych technik przetwarzania rozproszonego, które korzystały z ogólnie znanych nazw do zdalnego lokalizowania usług, usługi sieciowe dają możliwość wyszukiwania na podstawie zestawu operacji wykonywanych przez usługę. Standard UDDI jest proponowaną technologią katalogową dla usług sieciowych.

Rozdział 7. jest poświęcony szczegółowemu wyjaśnieniu technik wyszukiwania, w szczególności standardowi UDDI.

Połączenie stosów technologii

Czy pojedyncza usługa sieciowa musi korzystać ze **wszystkich** tych technologii, aby mogła być w pełni uznawana za usługę sieciową? Z pewnością nie.

Analizując stosowi połączenia, można powiedzieć, że żaden konkretny protokół sieciowy, nawet HTTP, nie jest wymaganym elementem usługi sieciowej — można korzystać z dowolnej liczby protokołów. Niektóre usługi nawet nie potrzebują istnienia protokołu sieciowego. Technologie takie jak WSIF (*Web Services Invocation Framework*) (<http://www.alpha-works.ibm.com/tech/wsif>) dają możliwość implementacji usługi sieciowej jako zwykłej metody w języku Java, gdzie klient i dostawca usługi rezydują w obrębie tej

samej wirtualnej maszyny Javy. Przemieszczając się w górę stosu, odkrywamy, że nawet SOAP nie jest wymaganą technologią w usługach sieciowych. Jako usługę sieciową można potraktować komponent, do którego dostajemy się poprzez standardowe żądanie POST protokołu HTTP. W tych przypadkach wspólnym elementem decydującym o tym, że możemy traktować opisane rozwiązania jako usługi internetowe, jest użycie języka WSDL do opisu usług.

Czy więc obecność opisu jest warunkiem koniecznym uznawania komponentu za usługę sieciową? Tak jak wcześniej odpowiedź brzmi „nie”. Wiele usług, szczególnie tych stworzonych przed pojawieniem się standardu SOAP, nie posiada odpowiadającego mu opisu usługi. Komponenty te są uważane za usługi sieciowe, trzeba jednak pamiętać, że bez istnienia opisu na usłudze nie można wykonać operacji publikacji, wyszukiwania i powiązania w architekturze zorientowanej na usługi. Wraz z upowszechnianiem się standardu WSDL będzie coraz mniej usług pozbawionych opisu. Wielu projektantów doszło do wniosku, że można zdefiniować usługę sieciową jako „cokolwiek, co można opisać w języku WSDL”.

Czy usługa sieciowa musi się pojawić w rejestrze UDDI, aby zyskać miano usługi sieciowej? Oczywiście nie. Wiele usług dostępnych w sieci nie jest opublikowanych w żadnym rejestrze UDDI ani nie obsługuje mechanizmów wyszukiwania ADS oraz DISCO.

Zgodzisz się więc, że dana usługa sieciowa nie musi wykorzystywać **wszystkich** opisanych technologii, aby być uważaną za usługę sieciową. Czy jednak istnieje **jakakolwiek** technologia wspólna dla **każdej** usługi sieciowej? Jeżeli zapoznasz się z powyższymi argumentami, odpowiedź na to pytanie również będzie negatywna. Czy SOAP jest wymagany? Nie. Może WSDL? Nie. UDDI? Też nie. Nie istnieje pojedyncza technologia, która jednoznacznie klasyfikowałaby komponent jako usługę sieciową. Z tego powodu trudno jest zdefiniować, czym jest usługa sieciowa.

Oprócz tworzenia wspaniałych specyfikacji przemysł informatyczny pracuje również nad budową oprogramowania spełniającego ustanowione standardy, aby móc rozwiązywać rzeczywiste problemy biznesowe. W tej książce do uruchamiania przykładów korzystamy z platformy Apache Axis. To zaawansowana platforma do wdrażania usług sieciowych o wysoce skalowalnej i rozszerzalnej architekturze. Jest ona szczegółowo opisana w rozdziale 4.

Istnieją również inne implementacje pochodzące od różnych producentów. Rozdział 8. przybliży najlepsze z obecnie dostępnych narzędzi, wraz z opisem ich możliwości oraz wzajemnej zgodności.

Interoperacyjność jest fundamentalnym założeniem technologii usług sieciowych. Czy w sieci WWW moja przeglądarka przejmuje się tym, jakiego serwera WWW używasz? Nie. Tak samo jest z usługami sieciowymi. Każdy klient powinien być w stanie wywołać dowolną, standardową (bez własnych rozszerzeń) usługę sieciową operującą na każdej platformie. Wszystko jeszcze przed nami, ale trwają prace nad osiągnięciem tego stanu, ponieważ każdy jest świadomy, że to jedyna droga do sukcesu usług sieciowych (oraz dynamicznej integracji aplikacji).

Podsumowanie

W tym rozdziale przedstawiliśmy definicję usług sieciowych i wskazaliśmy, w jaki sposób technologia ta pomoże w rozwoju biznesu. Zbudowaliśmy też koncepcyjny model — architekturę zorientowaną na usługi — którego można używać w rozważaniach nad problemami związanymi z usługami sieciowymi. Pokrótkie omówiliśmy także paletę technologii stowarzyszonych z usługami sieciowymi i przedstawiliśmy pomysł na ich ustrukturalizowanie poprzez pojęcie stosów.

Pozostała część książki bazuje na dotychczasowym materiale. W rozdziale 2. zajmujemy się standardem, z którego wyrasta technologia usług sieciowych — metajęzykiem XML. Rozdział 3. nawiązuje do dyskusji z poprzedniego — zajmuje się stosem połączenia, a w szczególności protokołem SOAP, który jest preferowanym mechanizmem dostępu do usług sieciowych. W rozdziale 4. omawiamy implementację SOAP w projekcie Apache Axis. Materiał z rozdziału 5. poszerza informacje o SOAP i Axis, opisując sposób realizacji innych wymagań e-biznesowych w usługach sieciowych, np. bezpieczeństwo. Rozdział 6. jest poświęcony stosowi opisu, przy czym skupiamy się na tym, skąd klient usługi czerpie informacje o rodzaju i adresie docelowym przesyłanych komunikatów. W rozdziale 7. zajmujemy się stosem wyszukiwania, a w szczególności standardem UDDI. Rozdział 8. zawiera omówienie innych platform do budowy systemów opartych na usługach sieciowych. Zamykamy tę książkę rozdziałem 9. „Przyszłe koncepcje”, w którym omawiamy przyszłe kierunki rozwoju usług sieciowych.