JAVA czy .NET?

Grzegorz Grudziński
gsg@tls-technologies.com

28 listopada 2001

Streszczenie

Referat omawia dwa konkurencyjne podejścia do budowy rozproszonych aplikacji biznesowych: Java 2 Enterprise Edition i Microsoft .NET. Obie technologie mają podobne ambicje i zbliżone obszary zastosowania. Wiele je rożni, wiele łączy. W wielu przypadkach wybór którejś z nich wymaga starannego rozważenia za i przeciw.

Chcemy wyjaśnić, na czym obie technologie polegają, i podać zbiór pytań, na które trzeba odpowiedzieć, zanim wybierze się jedną z nich.

1 Wstęp

Celem niniejszego tekstu jest porównanie dwóch ważnych technologii budowy systemów rozproszonych: Java 2 Enterprise Edition oraz Microsoft .NET. Porównanie ma na celu przedstawienie tych cech obu technologii, które są decydujące przy dokonywaniu wyboru: której z nich użyć w poważnym, zapewne komercyjnym, zastosowaniu?

Celem dokumentu nie jest natomiast przedstawienie szczegółowych różnic technicznych, które między nimi występują. Tematem, którego nie będziemy w zasadzie poruszać, jest jedno z (potencjalnie) ważnych zastosowań obu technologii, jakim jest tworzenie Web Services. Naszym zdaniem zagadnienie to jest marginalne przy wyborze między J2EE i .NET, które mają o wiele szerszy zakres zastosowań.

2 Krótkie omówienie obu technologii

2.1 Java 2 Enterprise Edition

2.1.1 Ogólne omówienie

Java 2 Enterprise Edition (w skrócie J2EE) jest zestawem specyfikacji, stworzonych przez firmę Sun Microsystems wraz z wieloma grupami roboczymi, których członkowie reprezentują wiele firm obecnych na światowym rynku oprogramowania. Podstawą dla wszystkich tych specyfikacji jest język programowania Java.

Java 2 Enterprise Edition powstała dla ułatwienia budowy, wdrażania i zarządzania skomplikowanymi systemami, zbudowanymi z użyciem architektury wielowarstwowej (*ang. multi-tiered*). Systemy budowane z użyciem J2EE są najczęściej wyposażone w interfejs użytkownika wykorzystujący przeglądarkę WWW (choć nie jest to koniecznością ani regułą).

Java 2 Enterprise Edition sama w sobie nie jest produktem — jest zbiorem specyfikacji, dla których stworzono wiele implementacji (komercyjnych i niekomercyjnych, Open Source i closed source). Firma Sun Microsystems dla niektórych elementów J2EE udostępnia implementacje wzorcowe, które mogą być wykorzystywane bezpośrednio, jednak w zamyśle powinny służyć ułatwieniu testowania innych implementacji standardu.

2.1.2 Składniki

Na Java 2 Enterprise Edition składają się między innymi:

- język programowania Java i maszyna wirtualna dla tego języka
- technologia budowy dynamicznych stron WWW (Java Server Pages, serwlety)
- technologia komponentów rozproszonych (Enterprise Java Beans)
- zestaw protokołów komunikacyjnych (IIOP, RMI)
- standardowe interfejsy programistyczne:
  - Java Messaging System (JMS) — do komunikacji asynchronicznej i przesyłania komunikatów
  - Java Transaction API (JTA) — do realizacji rozproszonych transakcji
  - Java Database Connectivity (JDBC) — do komunikacji z bazami danych

- Java Connector Architecture — do komunikacji z różnego rodzaju istniejącymi systemami (np. systemami ERP)
- Java Naming and Directory Interface (JNDI) — do komunikacji z usługami katalogowymi
- ... i wiele innych

- dodatkowe usługi (obsługa analizy i generacji dokumentów XML-owych, komunikacja przez protokół SOAP itp.)

2.2 Microsoft .NET

2.2.1 Ogólne omówienie

Microsoft.NET jest zestawem produktów i strategii, które mają ułatwić budowę systemów rozproszonych w architekturze wielowarstwowej. Systemy budowane z użyciem Microsoft.NET są często z interfejsem opartym o przeglądarkę WWW.

Microsoft.NET jest w dużym stopniu rozwinięciem poprzedniej architektury i strategii Microsoftu o nazwie Windows DNA. Windows DNA obejmuje wiele znanych technologii i produktów Microsoftu, a w szczególności model komponentów i komunikacji między nimi COM+, Microsoft Message Queue (MSMQ), Microsoft Transaction Server (MTR), bazę danych Microsoft SQL Server. Microsoft.NET zastępuje te technologie i produkty nowymi wersjami i wprowadza nowe technologie.

2.2.2 Składniki

Na Microsoft.NET składają się między innymi:

- niezależny od procesora język pośredni Microsoft Intermediate Language (MSIL) i system wykonawczy (wspólny dla wielu języków programowania) Common Language Runtime (CLR)
- technologia budowy dynamicznych stron WWW (ASP.NET)
- technologia komponentów rozproszonych (.NET Managed Components)
- zestaw protokołów komunikacyjnych (COM+, SOAP)
- zestaw produktów implementujących podstawowe usługi:

  - Microsoft Message Queue (MSMQ) — do komunikacji asynchronicznej i przesyłania komunikatów
  - Microsoft Transaction Server (MTA) — do realizacji rozproszonych transakcji
  - Active Data Objects.NET (ADO.NET) — do komunikacji z bazami danych

- .NET Servers — zestaw produktów implementujących inne usługi (stanowią przepakowanie zestawu najpopularniejszych produktów Microsoftu pod nową wspólną nazwą)
  - Internet Information Server 2000 — serwer WWW wraz ze środowiskiem wykonawczym dla ASP.NET
  - SQL Server 2000 — baza danych
  - Exchange 2000 Server — oprogramowanie do pracy grupowej
  - Commerce Server 2000 — oprogramowanie do tworzenia elektronicznych systemów handlowych
  - Application Center Server 2000 — do zarządzania klastrami
  - Host Integration Server 2000 — do komunikacji z różnego rodzaju istniejącymi systemami (np. systemami ERP)
  - Internet Security and Acceleration Server 2000 — zapora ogniowa i serwer pośredniczący WWW
  - BizTalk Server 2000 — do integracji procesów biznesowych przez Internet w oparciu o XML

- zestaw narzędzi programistycznych (Visual Studio.NET)

2.3 Porównanie dostępnych mechanizmów

Java 2 Enterprise Edition jest zbiorem specyfikacji, dla których istnieje wiele implementacji w postaci produktów różnych firm. Microsoft.NET jest zestawem technologii i produktów pochodzących od jednego producenta i jako całość nie opiera się o standardy — formalne ani przemysłowe.

Mimo tej różnicy, obie technologie udostępniają porównywalny zestaw mechanizmów, potrzebnych do budowy oprogramowania rozproszonego. Dlatego powody czysto techniczne nie wystarczają do podjęcia decyzji, która z technologii nadaje się bardziej do realizacji tego czy innego projektu.

3 Analiza porównawcza czynników o znaczeniu rynkowym

W dalszej części dokumentu przedstawiamy porównanie obu technologii z punktu widzenia ich wartości rynkowej. To porównanie powinno okazać się bardziej przydatne przy podejmowaniu decyzji o wyborze technologii do realizacji konkretnego projektu lub przy ustalaniu strategii firmowej.

3.1 Czas potrzebny do wprowadzenia produktu na rynek

Czas potrzebny do wprowadzenia produktu na rynek jest jednym z decydujących czynników wpływających na sukces lub klęskę projektu komercyjnego. Zarówno J2EE jak i Microsoft.NET mają szereg cech, wpływających na ten czas.

Java 2 Enterprise Edition ma kilka mechanizmów, których nie znajdziemy w Microsoft.NET. Automatyczne zarządzanie trwałym stanem (*persistent state management*) oferowane przez komponenty entity beans pozwala na uniknięcie jawnego programowania dostępu do danych przechowywanych w bazach relacyjnych, co przyspiesza tworzenie oprogramowania i uniezależnia je od konkretnych baz danych. Mechanizm ten pozwala jednak na pełną programową kontrolę nad transakcjami. Znaczniki JSP definiowalne przez programistę pozwalają na rozdzielenie projektowania wyglądu interfejsów WWW od oprogramowania logiki programu. Interfejs JCA pozwala na rozszerzanie systemu o możliwość komunikacji z innymi istniejącymi systemami (na przykład SAP R/3, Siebel, CICS/COBOL i wiele innych) bez modyfikacji istniejącego oprogramowania.

Microsoft.NET pozwala na budowę interfejsu użytkownika, który jest niezależny od urządzenia klienckiego — komponenty ASP.NET same generują odpowiednie dane w języku HTML, WML, XML itp. Mechanizm komunikacji asynchronicznej z komponentami Queued Components jest bogatszy i bardziej elastyczny niż występujący w J2EE mechanizm Message Driven Beans (MDB). Z drugiej strony, Microsoft starał się jak najbardziej uprościć mechanizmy programistyczne, rezygnując z niektórych istotnych w typowych zastosowaniach (na przykład serwery ze stanem (*stateful*), proste transakcje). Wyjście poza narzucone ograniczenia jest możliwe, ale jednocześnie uniemożliwia korzystanie z większości mechanizmów, które czynią technologię Microsoft.NET atrakcyjną.

Obie technologie dostarczają ważnych mechanizmów przyspieszających tworzenie oprogramowania, ale mechanizmy te nie są bezpośrednio porównywalne.

3.2 Niezależność

Istnieje wiele czynników, od których technologia budowy oprogramowania może być zależna lub nie. Omawiamy najważniejsze z punktu widzenia porównywanych technologii.

3.2.1 Od dostawcy

J2EE jest zbiorem specyfikacji, który ma wiele implementacji różnych producentów, takich jak IBM, HP, Oracle, IPlanet, Borland, Sybase i wielu innych. Użytkownik nie jest zatem zdany na jednego dostawcę oprogramowania, ani nawet na konieczność wyboru produktu komercyjnego (projekty JBoss, Jonas, Enhydra).

Co prawda decydujący głos przy tworzeniu specyfikacji J2EE ma firma Sun Microsystems, jednak skuteczność rynkowa wymaga współpracy z innymi dostawcami oprogramowania. Praca nad specyfikacjami wchodzącymi w skład J2EE prowadzona jest w ramach Java Community Process (JCP), a aktywny udział w pracach biorą w zasadzie wszystkie liczące się na rynku firmy poza Microsoftem. Dotychczasowy przebieg prac wskazuje, że nie należy obawiać się dominacji Suna.

Przenośność między implementacjami J2EE różnych dostawców nie jest doskonała. Specyfikacja J2EE nie obejmuje wszystkich mechanizmów, które są potrzebne przy tworzeniu skomplikowanych systemów (chociażby wysoka dostępność czy wsparcie dla klastrów i równoważenia obciążenia). Jednak ocenia się, że przy przenoszeniu oprogramowania opartego o J2EE między systemami różnych dostawców zmiany wymaga jedynie kilka (do 10) procent kodu.

Dla odmiany Microsoft.NET jest rozwiązaniem jednego dostawcy. Obejmuje system operacyjny, zestaw interfejsów programistycznych, zestaw produktów. Firma decydująca się na zastosowanie technologii Microsoft.NET jest zatem skazana na decyzje marketingowe i technologiczne podejmowane przez Microsoft.

Część mechanizmów związanych z technologią Microsoft.NET została jawnie udokumentowana i objęta standaryzacją (język pośredni MSIL i system wykonawczy CLR, protokół komunikacyjny SOAP). Mimo to, dominująca większość mechanizmów potrzebnych przy budowie rzeczywistych systemów jest zależna od produktów dostępnych tylko przy korzystaniu z produktów Microsoftu. Wykorzystanie mechanizmów objętych standaryzacją pozwala jedynie na współpracę innych systemów z systemami opartymi o technologie i produkty Microsoftu, a nie na uzależnienie się od Microsoftu jako dostawcy systemu i narzędzi.

3.2.2 Od języka programowania

Java 2 Enterprise Edition jest oparta o język programowania Java i maszynę wirtualną dla tego języka. Komunikacja z oprogramowaniem napisanym w innych językach wymaga wykorzystywania pomostów opartych o protokół komunikacyjny IIOP i technologię CORBA, interfejsy JNI, adaptery JCA, protokół SOAP (lub inne protokoły komunikacji sieciowej). W zasadzie nie ma możliwości bezpośredniej intergracji z kodem napisanym w innych niż Java językach programowania.

Technologia Microsoft.NET jest niezależna od konkretnego języka programowania. Maszyna wirtualna i system wykonawczy CLR pozwalają na kompilację większości istniejących języków programowania (także tych bardziej nietypowych) i na bezpośrednie łączenie kodu napisanego w różnych językach.

Jednakże MSIL i CLR są oparte o model programowania obiektowego. łączenie kodu napisanego w różnych językach programowania oparte jest o wywołania metod między obiektami i dziedziczenie. Wiele języków programowania nie ma mechanizmów obiektowych, albo ma mechanizmy różniące się od konkretnego modelu narzuconego przez CLR. Ponadto CLR narzuca dodatkowe ograniczenia, które nie każdy język spełnia (dotyczy to nawet języków tak popularnych, jak C czy C++).

Zatem implementacje wielu języków programowania dla systemu wykonawczego CLR wymagają niestandardowych zmian w tych językach. Użycie istniejącego kodu może zatem wymagać wielu zmian i w większości przypadków nie jest czynnością aż tak prostą, jak można by sobie życzyć.

3.2.3 Od systemu operacyjnego

Java 2 Enterprise Edition jako technologia oparta o język Java i jego maszynę wirtualną jest niezależna od systemu operacyjnego. Implementacje J2EE są dostępne dla wszystkich istotnych architektur serwerowych i systemów operacyjnych (w tym wszystkich ważnych odmian Uniksa i Linuksa, systemów mainframe, Microsoft Windows i innych). Możliwość stosunkowo łatwego uzyskania dostępu do źródeł maszyny wirtualnej Javy pozwala na jej przeniesienie nawet na bardziej nietypowe systemy.

Maszyna wirtualna i system wykonawczy CLR są również niezależne od systemu operacyjnego. Implementacja oferowana przez Microsoft jest dostępna dla wybranych wersji systemu Microsoft Windows, jednak istnieją (i powstaną) implementacje dla innych systemów.

Mimo to nadzieja na przenośność do innych systemów operacyjnych (i architektur procesora) jest cokolwiek złudna. Microsoft.NET to nie tylko maszyna wirtualna, to także zestaw stan-

dardowych usług, protokołów komunikacji i produktów serwerowych. Wszystkie rzeczywiste systemy będą korzystać z wielu z nich, gdyż to na nich właśnie opiera się cała atrakcyjność technologii Microsoftu. Nie sposób wyobrazić sobie rzeczywistego dużego systemu, który nie korzysta chociażby z:

- ASP.NET i jego bibliotek komponentów interfejsu użytkownika
- dostępu do baz danych przez ADO.NET
- asynchronicznej komunikacji przez MSMQ
- obsługi transakcji rozproszonych.

Usługi i produkty te są dostępne wyłącznie dla systemów operacyjnych Microsoftu. Standardowym mechanizmem komunikacji komponentów jest nowa wersja COM+, niedostępna dla innych niż Microsoftu systemów operacyjnych [1]. Narzędzia programistyczne Microsoftu (Visual Studio.NET) i popularne wśród programistów języki programowania (Visual Basic) także nie są dostępne dla innych platform.

Nie można zatem budować rzeczywistych systemów opartych o technologię Microsoft.NET bez wykorzystania systemów operacyjnych i technologii firmy Microsoft. Przenośność kodu oferowana przez MSIL i CLR nie stanowi żadnego rozwiązania [2].

3.3 Integracja z istniejącymi systemami

Integracja technologii Java 2 Enterprise Edition z istniejącymi systemami (*legacy systems*, *legacy code*) jest możliwa na kilka sposobów:

- przez systemy wymiany komunikatów (JMS)
- standardowe protokoły (na przykład CORBA/IIOP, SOAP)
- interfejsy JNI do korzystania z istniejących bibliotek kodu maszynowego

[1] SOAP jako drugi standardowy mechanizm jest przydatny raczej przy komunikacji przez Internet gdyż wprowadza duże narzuty.

[2] Poza tym technologia Microsoft.NET nie wymaga bynajmniej, by oprogramowanie było dostarczane w postaci kodu MSIL. Kompilatory Visual Studio.NET mogą bezpośrednio generować kod maszynowy. Kod MSIL korzystający z technologii Microsoftu nie da się i tak wykonać bez środowiska systemowego Microsoftu, więc zapewne wielu producentów oprogramowania będzie rozpowszechniać swoje produkty w postaci kodu binarnego specyficznego dla procesora i systemu, a nie kodu MSIL.

- J2EE Connector Architecture (JCA).

W przypadku integracji z istniejącymi złożonymi systemami, szczególnie atrakcyjnym mechanizmem jest JCA. Jest to specyfikacja pozwalająca na tworzenie modułów interfejsu (adapterów) dla poszczególnych systemów (takich jak SAP R/3, BAAN, Siebel itp.), podobnie jak JDBC pozwala na tworzenie adapterów do różnych baz danych. Wykorzystanie JCA pozwala na zachowanie modularności przy integracji z różnymi systemami (adaptery JCA dla różnych systemów można tworzyć i ładować niezależnie, tak samo jak sterowniki JDBC).

Integracja technologii Microsoft.NET z istniejącymi systemami jest także możliwa na wiele sposobów:

- przez systemy wymiany komunikatów (MSMQ)
- standardowe protokoły (na przykład CORBA/IIOP, SOAP)
- Host Integration Server 2000.

Host Integration Server 2000 jest produktem o analogicznym przeznaczeniu jak adaptery JCA. Oferuje on podobny rodzaj funkcjonalności. Jednak jest to konkretny, zamknięty produkt, a zatem nie da się go rozszerzyć o możliwość integracji z systemami, których nie przewidział Microsoft.

3.4 Odbiór rynkowy

Java 2 Enterprise Edition jest promowana na rynku przez wszystkich dostawców tej technologii, co obejmuje kilkadziesiąt znanych i szanowanych firm.

Z drugiej strony, Microsoft.NET jest promowane przez Microsoft, którego możliwości marketingowe są prawdopodobnie największe na rynku.

Należy zatem przyznać, że jak dotąd na polu marketingu wygrywa technologia Microsoftu.

3.5 Dojrzałość

Java 2 Enterprise Edition jest technologią dostępna na rynku od kilku lat. Część mechanizmów dostępnych w najnowszej wersji specyfikacji (SOAP i Web Services, JCA, MDB) jest nowa, jednak technologia J2EE jest sprawdzoną i stabilna, podlegając obecnie zmianom o charakterze ewolucyjnym, nie rewolucyjnym. To samo dotyczy większości najważniejszych implementacji J2EE.

Część technologii Microsoft.NET jest oparta o niektóre mechanizmy dostępne wcześniej w ramach Microsoft DNA — technologii obecnej na rynku przez rozsądnie długi okres. Jednak właściwie wszystkie technologie, które nie są całkiem nowe, uległy zauważalnym zmianom zarówno w zakresie specyfikacji, jak i implementacji. Do tego stopnia, że Microsoft.NET jako platforma technologiczna jest cały czas w fazie beta testów i nie została jeszcze udostępniona jako pełny produkt komercyjny.

3.6 Przeróbka istniejących systemów

Java 2 Enterprise Edition nie stwarza żadnych problemów. Oprogramowanie tworzone w tej technologii nie wymaga zauważalnie dużych zmian wraz z nowszymi wersjami specyfikacji J2EE.

Technologia Microsoft.NET opiera się o znane mechanizmy (takie jak COM+ i MTS), jednak mechanizmy te zostały zaimplementowane na nowo i w oparciu o nową specyfikację.

Użycie nowego systemu wykonawczego CLR wymusza zmianę typów danych we wszystkich językach programowania. Kod oparty o COM+ CLR (czyli .NET Managed Components) wymaga dość znacznych zmian w stosunku do starego modelu COM+ (związanych z odśmiecaniem, unikaniem stosowania wskaźników, mechanizmem rejestru Windows itp.). Kod nie korzystający z CLR i (niemal) nie podlegający ograniczeniom nie może korzystać z wielu istotnych mechanizmów technologii .NET. Istotnym zmianom podlega technologia budowy interfejsów WWW.

Widać zatem, że przeróbki istniejącego kodu będą dość znaczne i nie jest to problem, który można zignorować.

3.7 Narzędzia

Oprogramowanie dla Java 2 Enterprise Edition można tworzyć za pomocą narzędzi dostępnych od wielu producentów, takich jak Sun, IBM, Oracle, Borland, Sybase, WebGain i wielu innych. Istnieje wiele produktów dostępnych za darmo lub bardzo tanio.

Microsoft jest znany ze swojego środowiska programistycznego Visual Studio. Dla technologii Microsoft.NET dostępna jest wersja Visual Studio.NET. Tradycyjnie już jest to narzędzie pozwalające na bardzo sprawne tworzenie oprogramowania i obejmujące wszystkie mechanizmy dostępne w ramach technologii Microsoftu.

W sumie narzędzia dla technologii Java 2 Enterprise Edition liczone łącznie dają szerszą funkcjonalność niż Visual Studio.NET. Jednak ten produkt wyprzedza prawdopodobnie niemal każde pojedyncze narzędzie dla technologii Java 2 Enterprise Edition.

3.8 Koszt wdrożenia

Cena implementacji technologii Java 2 Enterprise Edition waha się od zera (implementacje darmowe) do dziesiątek tysięcy dolarów za licencję na jeden procesor. Implementacje działają na platformach od bardzo tanich (komputer PC z systemem Linux) do bardzo drogich (systemy Mainframe, wieloprocesorowe serwery uniksowe).

Technologia Microsoft.NET wymaga serwerów z procesorem x86 i systemów operacyjnych Microsoftu. Jest więc droższa od najtańszych implementacji J2EE i tańsza od najdroższych.

Należy jednak zwrócić uwagę, że przy większości projektów komercyjnych (za wyjątkiem bardzo małych) koszt związany z licencją na oprogramowanie jednej lub drugiej technologii jest zaniedbywalny w porównaniu z kosztem sprzętu, szkolenia programistów, produkcji oprogramowania, wdrożenia i utrzymania systemu po wdrożeniu.

3.9 Wydajność

W zasadzie nie da się porównać wydajności dwóch różnych technologii budowy systemów oprogramowania. Wydajność stworzonego systemu zależy od użytego sprzętu, natury rozwiązywanego problemu, architektury oprogramowania (która może być bardzo różna w zależności od celów projektu) i wielu innych czynników.

Sytuacja jest tym trudniejsza, że Java 2 Enterprise Edition nie jest produktem, ale zbiorem specyfikacji. Zatem porównywać można wydajność systemów stworzonych w technologii Microsoft.NET z wydajnością systemów stworzonych za pomocą poszczególnych implementacji Java 2 Enterprise Edition.

Nie ma jednak powodów, by spodziewać się dramatycznych różnic pomiędzy oprogramowaniem zbudowanym w obu konkurencyjnych technologiach i działających na podobnym (i odpowiednim do zadania) sprzęcie. Być może oprogramowanie zrealizowane za pomocą technologii Microsoft.NET okaże się nieco szybsze od zrealizowanego w podobnej architekturze w technologii Java 2 Enterprise Edition, ze względu na daleko posuniętą integrację z systemem operacyjnym oraz pochodzenie wszystkich elementów rozwiązania od tego samego dostawcy. Należy się jednak spodziewać, że różnice będą zaniedbywalne w większości przypadków [3].

Jako ciekawostkę należy potraktować opublikowane niedawno porównanie wydajności (oraz rozmiaru kodu źródłowego) programu Java Pet Store [4] w implementacji firmy Oracle oraz pro-

[3] W tego typu porównaniach nawet 30% jest różnicą zaniedbywalną z praktycznego punktu widzenia, choć nie spodziewamy się różnic aż tak znacznych.

[4] Jest to przykładowy program firmy Sun microsystems pokazujący różne mechanizmy programowe technologii Java 2 Enterprise Edition.

gramu o podobnej funkcjonalności stworzonego w technologii Microsoft.NET. Porównanie zostało przygotowane przez firmę trzecią sponsorowaną przez Microsoft.

Wyniki wskazują, że program stworzony w oparciu o technologię Microsoft.NET działa w skrajnym przypadku niemal 30 razy szybciej, niż oryginalny program Java Pet Store w implementacji firmy Oracle. Jednak różnice między obydwoma implementacjami są na tyle duże, że wyniki porównania (zarówno liczbowe, jak analityczne) są całkowicie nieprzydatne.

3.10 Skalowalność

Specyfikacja Java 2 Enterprise Edition nie obejmuje skalowalności (jako możliwości uruchomienia systemu na klastrze serwerów). Jednak wszystkie liczące się implementacje komercyjne tej technologii dostarczają niezbędnych mechanizmów. W przypadku implementacji niekomercyjnych jest to oczywiście również możliwe, jednak wymaga oprogramowania w ramach tworzonego oprogramowania.

Również technologia Microsoft.NET pozwala na budowę systemów działających w architekturze klastrowej.

Jedyna istotna różnica w skalowalności polega na tym, że technologia Microsoft.NET jest przywiązana do komputerów w architekturze PC z procesorami x86, podczas gdy implementacje Java 2 Enterprise Edition mogą działać na dowolnych, także znacznie mocniejszych, maszynach (w tym na komputerach klasy mainframe i serwerach uniksowych wyposażonych w wieleset procesorów).

3.11 Wnioski

Nie mamy zamiaru przekonywać czytelników do wyboru żadnej z omówionych technologii [5]. Przedstawione różnice będą mieć różne znaczenie dla różnych osób. Celem tekstu jest tylko przedstawienie najistotniejszych argumentów, które pomogą czytelnikom dokonać własnego wyboru.

4 Materiały źródłowe

Tekst powstał o wiele różnych materiałów dostępnych w Internecie, przede wszystkim o artykuły porównawcze, opublikowanych na stronach WWW. Trudno mi wskazać wszystkie, które przyczyniły się do powstania tego tekstu, ale chciałbym szczególnie wyróżnić jeden z artykułów:

[5] Co nie znaczy, że nie mamy własnego zdania.

J2EE vs. Microsoft.NET, A comparison of building XML-based web services autorstwa Chada Vawtera i Eda Romana.

