Hockmann, Sebastian
Schrödinger Str. 10
07745 Jena
Tel.: 03641 / 601608
Tel.: 0177 / 589 37 15
e-mail: sebastian.hockmann@stud.tu-ilmenau.de
Matrikelnummer : 26144

Vergleich Microsoft .NET mit Sun ONE

Projektarbeit an der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau

Fachgebiet Telematik
Betreuer : Thorsten Strufe
Abgabetermin: 1.September 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Die Integration von Geschäftspartnern in die eigenen Geschäftprozesse und somit die Unternehmensübergreifende Zusammenarbeit ermöglicht eine effizientere Gestaltung des Unternehmens und Kostensenkung durch die Optimierung der eigenen Geschäftsprozesse. Dies ist notwendig um im globalen Konkurrenzdruck durch das E-Business im B2B Bereich und der Erschließung neuer Kunden und Geschäftsfelder zu bestehen. Problematisch sind hierbei vielfältige, disparate (evtl. besser: heterogene/ ungleiche) Systeme und die Verteilung der Geschäftsdaten über mehrere Systeme hinweg. Anwendungen sind meist auf ihre bestimmte Abteilung begrenzt und es werden auch längst nicht alle Endgeräte unterstützt.

Für einen reibungslosen Ablauf müssen Daten über diese Grenzen hinaus ausgetauscht werden. Für dieses Problem bestehen heute bereits Lösungen. Der Nachteil dieser ist aber deren Komplexität und Spezialisierung Wichtig ist eine einfache und unkomplizierte Technologie, die die Zusammenarbeit der Systeme ermöglicht. Zum einen wird hierfür eine geeignete Infrastruktur benötigt auf der das Framework betrieben werden kann und zum anderen ein einheitliches Programmiermodell, was die Programmierung verteilter Anwendungen erleichtert. Es sollten im Rahmen der gegenwärtigen Entwicklung des M-Commerce auch mobile Endgeräte unterstützt werden. Im Bemühen um die Anforderungen der neuen Generation des Internet sind zwei große Frameworks entstanden: Microsoft .NET und SunONE.

1.2 Zielsetzung

Ziel dieser Arbeit ist es der Vergleich der beiden Frameworks Microsoft .NET und SunONE unter Betrachtung ihrer Vor- und Nachteile. Dabei wird insbesondere die Thematik der Webservices im Zusammenhang mit den Frameworks näher betrachtet.

1.3 Vorgehensweise

Im ersten Teil der Arbeit wird ein kurzer Überblick über Ziel und Architektur der Webservices gegeben. Im zweiten Teil werden die Frameworks Microsoft .NET und SunONE kurz einführend betrachtet.

Im dritten Teil der Arbeit werden beide Frameworks anhand eines Kriterienkatalogs verglichen.

Im vierten Teil wird eine Zusammenfassung beider Architekturen mit Vorteil- und Nachteilen dargestellt und abschließend versucht einen Ausblick auf die zukünftige Entwicklung in diesem Umfeld zu geben.

2 Webservices

2.1 Was sind Webservices ?

Grob können Webservices als maschinenlesbare Webseiten beschrieben werden. Ein Programm kann an diese Webseite herantreten und diejenigen Daten bekommen die es weiterverarbeiten möchte.[1]

Diese Idee ist nicht neu. Bisher serviceorientierte Implementierungen wie RPC, CORBA, DCE, COM und RMI sind aber an ein bestimmtes Protokoll oder eine Programmiersprache gebunden. So kann beispielsweise COM nur mit COM interagieren. Zudem sind im Internet die notwendigen Ports für die Kommunikation oft durch Firewalls gesperrt. Eine echte und breite Eignung für das Web ergibt sich aufgrund des hohen COM/RMI Kommunikationsoverheads gegenüber SOAP und der Firewallproblematik nicht. Webservices sind webbasierte, plattformunabhängige modulare Applikationen, die in einem Verzeichnis veröffentlicht, lokalisiert und über ein Netzwerk, im allgemeinen das Internet, aufgerufen werden können. Langfristig bilden Webservices eine Integrationsplattform. Die Idee hierbei ist, die verschiedenen Webservices als Komponenten zu sehen und sie in neuen Applikationen zu verwenden. Die Applikationen können hierbei aus einer beliebigen Kombination von Webservices zusammengestellt werden.

2.2 Architektur

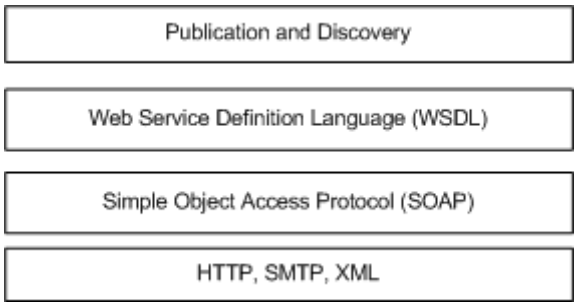


Abbildung 1 - Webservice Architektur[2]

[1] Siering /Microsoft Internet/

Die Komponenten der Webservicearchitektur können in verschiedene Schichten unterteilt werden. Diese sind in Abbildung 1 dargestellt und werden nachfolgend beschrieben. Die Basis der Webservice Architektur ist die Netzwerkschicht. Sie bildet die Grundvorrausetzung für die Erreichbarkeit eines Webservice. Das Standardprotokoll hierfür ist HTTP. Es können aber auch weitere Protokolle wie SMTP und FTP verwendet werden. SOAP ist das meist verwendete Kommunikationsprotokoll der Webservices und basiert auf HTTP, SMTP und XML. Über SOAP werden Nachrichten zwischen Webservices gesendet und empfangen. Die Beschreibung der Fähigkeiten eines Webservice erfolgt mit WSDL. WSDL ist der de facto Standard für die XML basierte Servicebeschreibung. Für die Publikation der Webservices gibt es verschiedene Mechanismen. Dazu gehören Direct Publish, DISCO (Discovery of Webservices), ADS (Active Directory Service) und Verzeichnisdienste wie UDDI (Universal Description, Discovery and Integration).

2.2.1 SOAP

Ein RPC (Remote Procedure Call) ist der Aufruf einer Funktion über Programm-,Prozess- oder Rechnergrenzen hinweg. SOAP (Simple Object Access Protocol) ist im Grunde die Umsetzung dieses RPC-Mechanismus im Internet. Hierbei werden die notwendigen Informationen für den Aufruf einer Funktion (Methode, Input-, Outputparameter,...) als XML Struktur von Sender A nach Empfänger B transportiert. Dort wird die richtige Methode gefunden und die entsprechende Implementierung mit den Inputparametern belegt. Nach der Verarbeitung werden die Ergebnisse zum Aufrufenden zurückgeschickt. Als Transportprotokoll wird HTTP verwendet.

2.2.2 WSDL

WSDL (Web Service Description Language) dient der Beschreibung von SOAP basierten Webservices. Mehrere Abschnitte in einer WSDL Datei beschreiben unterschiedliche Ebenen möglicher Webservice-Aufrufe. Der Aufbau ist sehr modular, beginnend mit der Definition wie Parameter und ganze Signaturen in XML zu codieren sind und endet mit der Zuordnung von XML Strukturen zu Kanälen, über die SOAP Aufrufe an den Webservice abgesetzt werden dürfen (SOAP, HTTP POST, SMTP).

[2] Technet /.NET für IT Experten/ S.4

2.2.3 UDDI[3]

Universal Description, Discovery and Integration (UDDI) ist eine Spezifikation für verteilte webbasierte Verzeichnisse von Webservices. Webservices werden dort von ihren Betreibern registriert und von den Nutzern gesucht. Die Webservicebeschreibung ist in einem XML Format implementiert (WSDL) und wird in der Registrierung

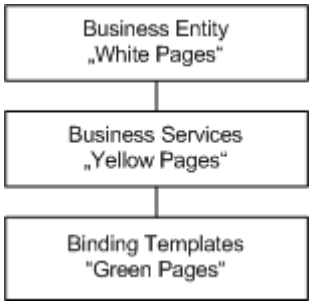


Abbildung 2 – UDDI Struktur

gespeichert. Das Verzeichnis besteht aus drei Bereichen wie Abbildung 2 dargestellt. Einen weiteren Überblick gibt Tabelle 1. Die jeweiligen Bereiche stehen für Beschreibungen der Dienste auf verschiedenen Ebenen. Allgemeine Informationen über den Anbieter (White Pages), Einordnung der Dienste in einen größeren Zusammenhang mithilfe von Taxonomien (Yellow Pages) und technische Informationen (Green Pages).

White Pages	Yellow Pages	Green Pages
Firmen- und Kontaktinfo Allgemeine Dienstbeschreibung	Dienst- & Produktindex Industriezuordnung Geographische Einordnung	E-Business Regeln Dienstbeschreibungen Dienstaufruf Datenbindung

Tabelle 1 - UDDI Pages

2.2.4 Zugriff auf das UDDI

Ein Nutzer interagiert via SOAP mit einem Verzeichnisdienst für Webservices (UDDI). Ein Anbieter eines Webservices veröffentlicht die Informationen über den Webservice in diesem Verzeichnis. Das Verzeichnis enthält alle Informationen über das Webserviceangebot der eingetragenen Webserviceanbieter.

[3] Vgl. Knuth /Webservices/ S.95

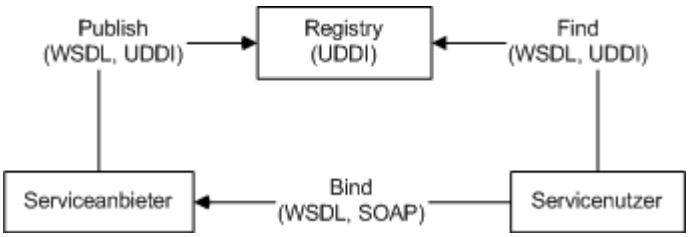


Abbildung 3 - Webservice Interaktion

Der einfachste Fall der Webservice Interaktion wird in Abbildung 3 beschrieben. Ein potentieller Nutzer sendet eine Anfrage zum Verzeichnisdienst (UDDI). Der Verzeichnisdienst liefert eine kategorisierte Sammlung über die gemäss der Anfrage verfügbaren Webservices. Nachdem der passende Webservice gefunden wurde (Find), kann der Nutzer Details wie Nachrichtenformat und benutzte Protokolle des betreffenden Service über den Verzeichnisdienst abfragen. Basierend auf der Servicebeschreibung (WSDL) wird eine Protokollbindung generiert und die Webservices können kommunizieren.

3 Einführung zu Microsoft .NET

3.1 Überblick[4]

Microsoft .NET ist eine völlig neue Entwicklung und stellt den Nachfolger von Windows DNA dar. Microsoft verwendet sehr starke Marketingaktivitäten auf .NET womit der Begriff teilweise nebulös erscheint. Im Kern ist .NET eine Entwicklungsplattform für Internet- und verteilte Applikationen sowie Webservices und soll die Entwicklung und Verteilung von Software vereinfachen. Die Interoperabilität zwischen Systemen und Applikationen soll verbessert und der interaktive und universelle Zugriff auf Applikationen von allen Geräten möglich sein. .NET besteht aus den Hauptbestandteilen .NET Framework, .NET Enterprise Server, .NET Building Block Services (HailStorm, myServices) und Visual Studio.NET die nachfolgend kurz beschrieben werden.

[4] Vgl. Hoffman, Gabriel, Gosnell, Hasan, Holm, Musters, Narkiewickz, Schenken,Thangarathinam, Wylie, Ortiz /.NET Framework/ S.10 ff.

3.2 .NET Framework

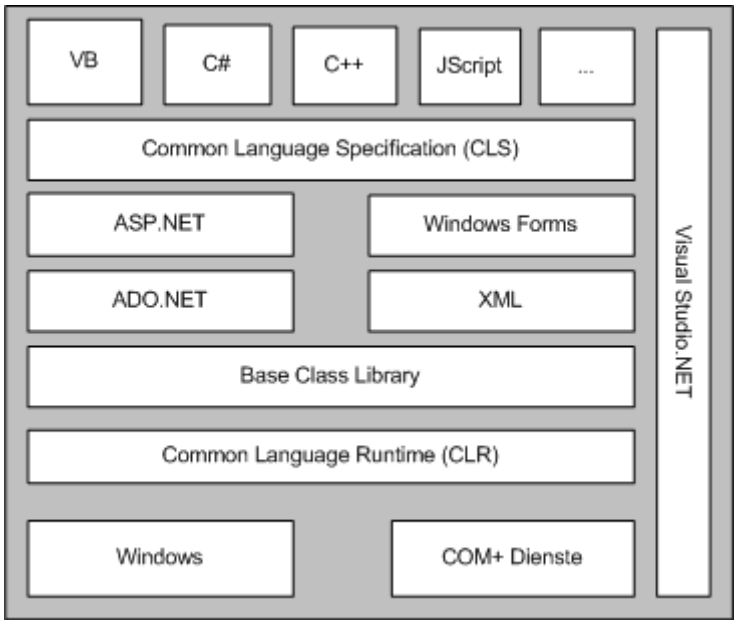


Abbildung 4 - .NET Framework[5]

Das .NET Framework stellt die Infrastruktur, Funktionalität und Dienste zur Entwicklung von Desktop-Anwendungen und Weblösungen bereit. Die Architektur kann in verschiedene Schichten aufgeteilt werden, wobei die oberen Schichten auf den unteren aufbauen. Nachfolgend werden die einzelnen Bestandteile des .NET Frameworks vorgestellt.

3.2.1 Windows und COM+ Dienste

Diese Komponenten sind nicht Bestandteil des .NET Framework, aber sie sind notwendig für die Arbeit mit dem .NET Framework SDK.

3.2.2 Common Language Runtime (CLR)

Die CLR ist eine umfangreiche Laufzeitumgebung und behandelt wichtige Laufzeitaufgaben für den Entwickler. Die Hauptaufgaben der CLR sind die Übersetzung von Zwischensprachencode (IL) in Maschinencode, die Speicherverwaltung, Verwaltung von Prozessen und Threads, die Durchsetzung der Sicherheitsmechanismen und das Laden von Komponenten. Die CLR ist um die Common Language Specification (CLS) aufbaut, welche eine allgemeine Menge von Datentypen für alle .NET konformen Sprachen definiert.

[5] Hoffman, Gabriel, Gosnell, Hasan, Holm, Musters, Narkiewickz, Schenken,Thangarathinam, Wylie, Ortiz /.NET Framework/ S.15

3.2.3 Base Class Library (BCL)

Die BCL stellt umfangreiche funktionale Basisklassen bereit und bietet eine objektorientierte Sicht über nahezu alle Betriebssystemleistungen. Der Zugriff auf das Dateisystem, die Anzeige von Fenstern, Druckfunktionen, Reflection, Serialisierung, Remoting, Grafikfunktionen, Datenzugriff usw. sind in hierarchischen Namensräumen organisiert. Die Klassen können in Benutzerklassen erweitert oder aber direkt benutzt werden.

3.2.4 Extended Class Library (ECL)

Die ECL umfasst abstrakte Klassen, die auf einen Aspekt der Entwicklung fokussiert sind. So wird ASP.NET für die Entwicklung Webservices, ADO.NET für den Datenzugriff, XML um Dokumente zu parsen und zu manipulieren und Windows Forms für die Entwicklung von Windows-basierten Anwendungen verwendet.

3.2.5 Common Language Specification (CLS)

Die CLS definiert Anforderungen an .NET konforme Sprachen.

3.2.6 Programmiersprachen

VB.NET, C++ und C# sind nur einige von den ca. 25 verfügbaren (zur Zeit 25?? Oder geplant?) Sprachen. .NET bietet eine Plattform für viele Programmiersprachen. Dabei werden auch nicht-Microsoft Sprachen wie COBOL und Perl unterstützt.[6] Auch Borland hat .NET Unterstützung für Delphi angekündigt.[7]

3.2.7 Visual Studio .NET

Visual Studio .NET ist die integrierte Entwicklungsumgebung (IDE) für das Programmieren mit dem .NET Framework und dient der Erhöhung der Entwicklerproduktivität.

3.3 .NET Enterprise Servers

Microsoft richtet im Rahmen der .NET Initiative viele seiner Serverprodukte auf .NET aus. Sie dienen dem Geschäftsprozessmanagement und der Integration verschiedenster IT-Infrastrukturen.

- **Windows 2000 Server** – Betriebssystemplattform für die .NET Server und Applikationen insofern das .NET Framework installiert ist. Windows 2000

[6] Microsoft /.NET Resources/

[7] Heise /Borland .NET/

Advanced Server ist die von Microsoft empfohlene Plattform für die Entwicklung von Unternehmensanwendungen. Aber auch Windows 2000 Professional und Windows XP sind nutzbar.

- **Application Center 2000** – bietet Management-Unterstützung für Applikationen um ihre Skalierbarkeit und Verfügbarkeit zu erhöhen
- **SQL Server 2000** – relationales Datenbanksystem (RDBMS) mit XML Unterstützung zur Datenspeicherung, Analyse und Indizierung.
- **Exchange Server 2000** – Unterstützung für HTTP und XML bei der Unternehmenskommunikation.
- **Host Integration Server 2000** – Integration mit Legacy-Hostsystemen
- **Internet Security and Acceleration Server (ISA)** – Internetverbindungs- und Firewallmanagement
- **Commerce Server 2000** – Entwicklung von e-Commerce Websites
- **Biztalk Server 2000** – Plattform für die Integration unterschiedlicher Plattformen, Applikationen und Diensten. Basiert auf XML Standards wie X12 und EDIFACT und unterstützt die Protokolle HTTP, SMTP, FTP und SOAP.
- **Mobile Information Server 2001**
  Plattform für den mobilen Zugriff auf bereits verfügbare Applikationen in der .NET Umgebung
- **Sharepoint Portal Server 2001**
  Aufbau von Portallösungen. Suchmechanismen und Dokumentenmanagement-Funktionen werden unterstützt.

3.4 .NET Building Block Services

Microsoft und dessen Partner erstellen einen Kern von Building Block Services die routinemäßige Aufgaben ausführen und die eine Basis darstellen, auf den Entwickler und IT-Geschäfte aufbauen können. Die Building Block Services setzen direkt auf der Plattform auf. Es werden Funktionalitäten für die Authentifikation, Notifikation, Messaging, Verzeichnis, Suche und Kalender bereitgestellt. Die Dienste nutzen XML und stehen sowohl im Intranet als auch Offline oder über das Internet zur Verfügung. Bei den ersten dieser Dienste handelt es sich um .NET myServices (früher HailStorm), die auf Building-blöcken wie Passport und .NET Alerts aufbauen. Die Daten sollen

dabei auf Microsoft eigenen Servern gespeichert werden. Dies hat heftigen Protest der Datenschützer ausgelöst, woraufhin Microsoft sich bereit erklärt hat die Datenspeicherung an Partner abzugeben. Microsoft hat nach eigener Aussage die Entwicklung dieser Dienst aber zunächst gestoppt.

3.5 Visual Studio .NET

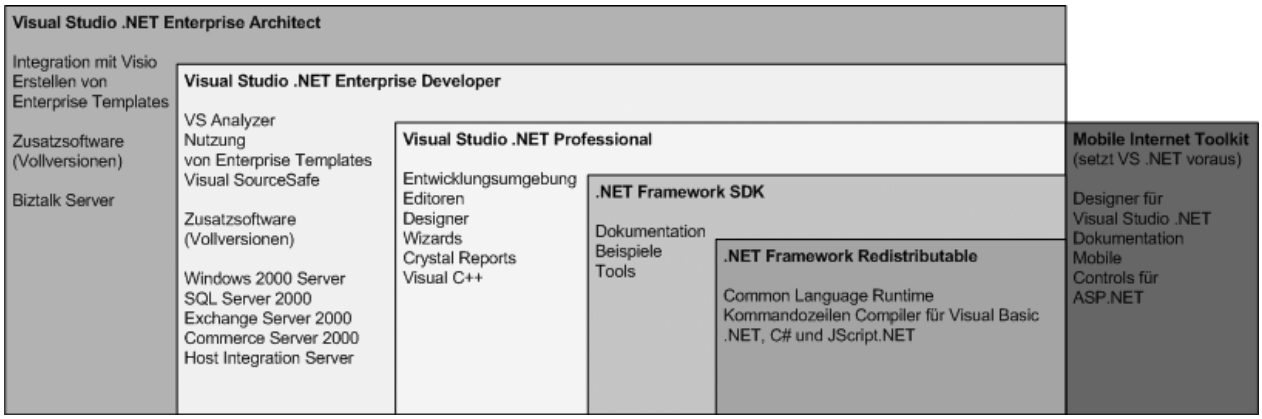


Abbildung 5 - Überblick über die .NET Entwicklerprodukte[8]

Die Integrierte Entwicklungsumgebung (IDE) Visual Studio NET ist vollständig in das .NET Framework integriert. Es bietet eine gemischte Sprachenumgebung und Cross-Language Debugging sowie RAD (Rapid Application Prototyping) Features für die Anwendungsentwicklung. Es gibt visuelle Designer für XML, HTML und Daten und erweitertes Debugging zwischen Projekten, inklusive Stored Procedures. Abbildung 5 gibt einen Überblick über die verschiedenen Versionen von Visual Studio .NET.

[8] Lelic, Schwichtenberg /Visual Studio .NET/ S.80

3.6 Architektur

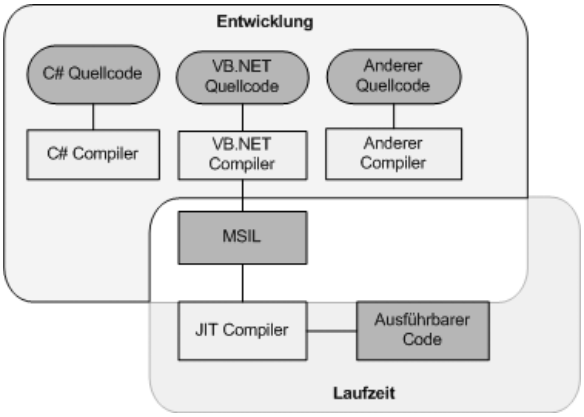


Abbildung 6 – Arbeitsweise der .NET-Compiler[9]

Die Entwicklung eines Projektes erfolgt in den jeweiligen Programmiersprachen. Der Quellcode wird anschließend durch den jeweiligen .NET Compiler in einen Zwischencode, den MSIL Code kompiliert. Der MSIL Code wird zusammen mit den Metadaten in einem Assembly gespeichert.

Metadaten beschreiben die wichtigen Daten über ein kompiliertes Programm. Dazu zählen beispielsweise eine Liste der Dateien in der Assembly, Versionsinformationen und Konfigurationsinformationen. Die Ausführung des Programms findet im Rahmen der CLR statt. Dazu wird die das? Assembly zusammen mit den Bibliotheksklassen vom Klassenlader geladen. Die Ausführung übernimmt der JIT Compiler. Ein gravierender Unterschied zu Java Bytecode besteht darin, das MSIL Code nicht für die Interpreterverarbeitung ausgelegt wurde. MSIL wird immer in prozessorspezifischen Maschinencode übersetzt und dann erst ausgeführt.

3.7 Webservices und .NET

In .NET werden Webservices mit ASP.NET entwickelt. Webservices basieren auf offenen Standards. Somit können auch Webservices die auf unterschiedlichen Plattformen entwickelt wurden mit. NET Webservices kombiniert werden.

Webapplikationen werden über das 3-Schichten Modell realisiert. In der Präsentationsschicht werden Web Forms eingesetzt. Die Geschäftslogik wird über .NET managed Componentes und dem Internet realisiert. Die dritte Schicht basiert auf einem RDBMS. Über einen Webbrowser kann auf eine installierte Webapplikationen

[9] Lovisach, Schulz, Violka /Sunspiration/

auf dem Webserver zugegriffen werden. Die Applikation selbst kann die Betriebssystemdienste oder Webservices auf diesem Webserver nutzen. Alternativ kann die Applikation auf unterschiedliche Webservices über das Internet zugreifen.

4 Einführung zu SunONE

4.1 Überblick[10]

Im Februar 2001 hat Sun seine auf J2EE basierende Architektur für Webservices, das Sun Open Network Environment (SunONE) vorgestellt. Nach der Aussage von Sun besteht SunONE aus den vier Hauptelementen Vision, Architektur, Plattform und

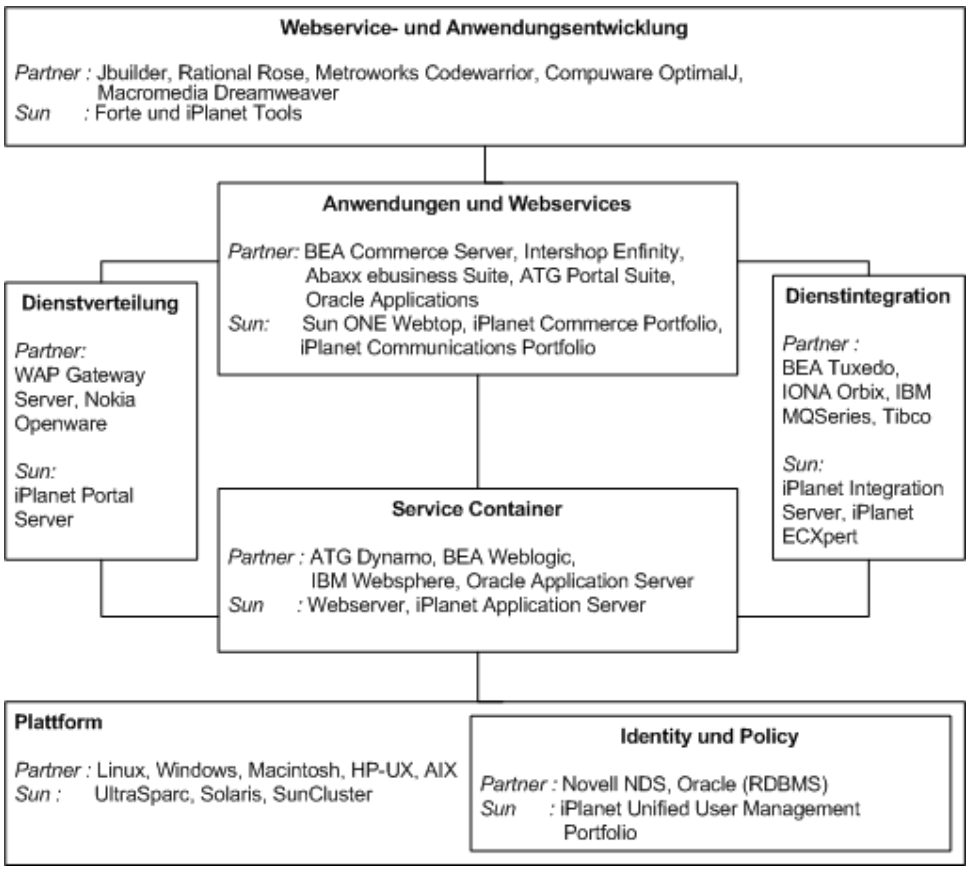


Abbildung 7 - SunONE Überblick[11]

Expertise. Vision ist ein Modell demzufolge die Software-Infrastruktur eines Unternehmens in der Lage ist Informationen, Daten und Anwendungen jeder Person, zu jeder Zeit an jedem Ort zur Verfügung zu stellen. Architektur meint ein auf offenen Standards basierendes, Softwaredesign, das es ermöglicht sowohl Legacy-Systeme als

[10] Sun /Sun ONE/

[11] Vgl. Schmidt /Sun ONE/

auch Produkte und Technologien anderer Anbieter zu integrieren. Plattform steht für ein, laut Sun, offenes, integrationsfähiges Produktportfolio, das es nicht nur ermöglicht heutige Geschäftsaufgaben zu erfüllen, sondern auch zukünftige Aufgabenstellungen zu meistern. Expertise meint die langjährige Erfahrung von Sun in Netzwerklösungen und im Support der Produkte.

Im oberen Teil der Abbildung 7 ist die Webservice und Anwendungsentwicklung dargestellt. Ein Webservice besteht aus mehreren diskreten Servicekomponenten oder Teilservices. Diese repräsentieren unterschiedlichen Content, Geschäftslogik und Applikationen. Der Entwicklungsprozess ist in zwei Phasen unterteilt. Zunächst werden die diskreten Services, auch Microservices genannt, entwickelt. Hierfür können integrierte Entwicklungsumgebungen, Codegeneratoren, XML Editoren und Authoring Tools eingesetzt werden. Service Assembler verwenden im zweiten Schritt Prozess-Modellierungs- und Workflow-Tools, Scripting Engines und andere Werkzeuge, um die Microservices zu Macroservices zusammenfügen. Sobald ein Webservice vollständig getestet wurde, kann er an eine Deployment Plattform übergeben werden. Benutzerdaten, Identität, Rollen, Präferenzen, Sicherheit und Policies werden in einem öffentlichen Verzeichnis verwaltet. Dieses kann auch nicht-personenbezogene Informationen wie Daten zu internen Ressourcen, Betriebsmitteln, Produktinformationen und Konfigurationsdaten enthalten. Die Plattform ist im unteren Teil der Abbildung dargestellt. Diese stellt die Umgebung für den Ablauf von Komponenten und Services bereit. Sie umfasst Betriebssystem, virtuelle Maschinen und andere Basiskomponenten für den Zugriff auf Hardware, Speicher und Netzwerke. Anwendungen und Webservices sind in der Mitte der Abbildung repräsentiert. Was für den Anwender wie ein kompletter Webservice aussieht, kann eine Zusammenstellung aus mehreren Microservices sein. Diese Microservices können jederzeit, auch zur Laufzeit, zusammengesetzt, konfiguriert und rekonfiguriert werden. Microservices können auf unterschiedliche Plattformen verteilt und in unterschiedlichen Sprachen geschrieben sein. Der Rest der Grafik illustriert die SunONE Deployment Architektur für Webservices. Auf der linken Seite dient die Dienstverteilung als Schnittstelle zum Nutzer. Basisverbindung, Lokation, Discovery und Kommunikation werden von dieser Schicht angeboten. Ziel ist es dem Nutzer die von ihm angeforderten Webservices zur Verfügung zu stellen. Ein Nutzer kann eine Person, Applikation oder ein Webservice sein. Webservices werden in einem Service Container ausgeführt. Dies kann ein Webserver oder ein Applikationsserver sein. Der Servicecontainer bietet eine

Laufzeitumgebung für den Webservice sowie Prozessmanagement und weitere Dienste. Abschliessend ist die Dienstintegration auf der rechten Seite verantwortlich für den Zugriff auf andere Webservices und Resourcen wie Datenbanken, Dateien, Verzeichnisse und Legacy Applikationen.

4.2 J2EE

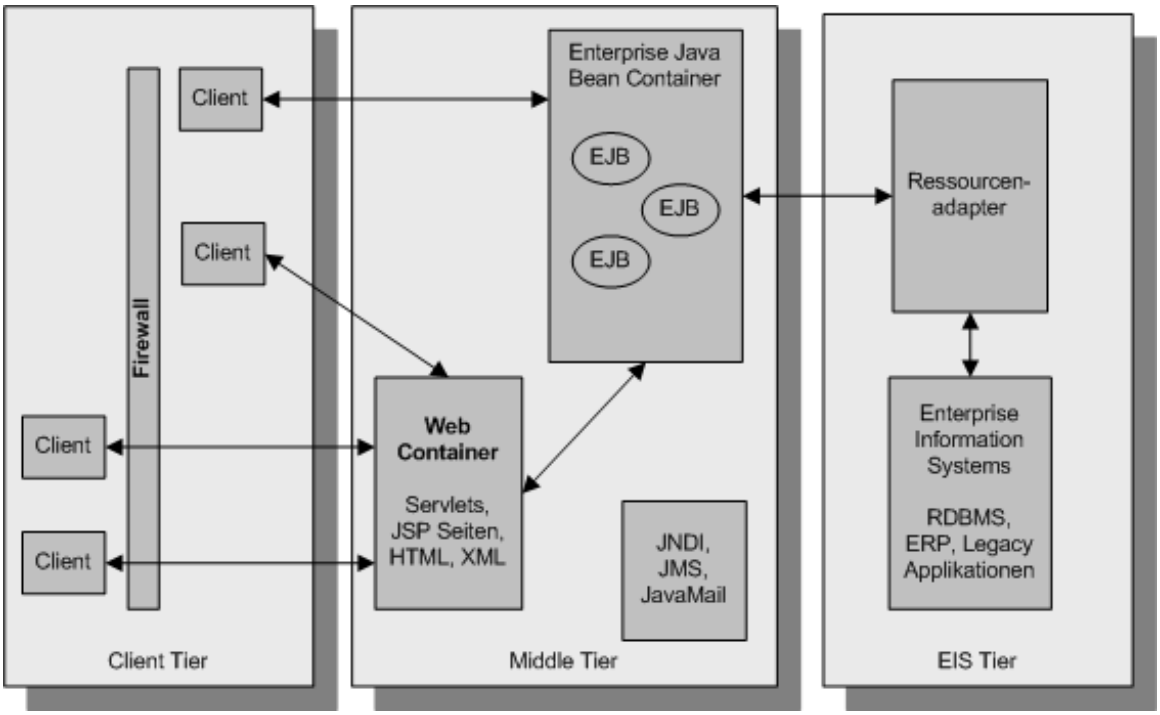


Abbildung 8 - J2EE Architektur[12]

Die Java 2 Plattform Enterprise Edition (J2EE) definiert eine Architektur für die Entwicklung von Multi-Tier-Unternehmensanwendungen und ist im Gegensatz zu .NET kein Produkt sondern ein auf Java basierender Industriestandard. [13]

J2EE baut auf der Java 2 Standardedition auf und bietet zusätzliche APIs die die Softwareentwicklung im Unternehmensumfeld erleichtern. J2EE umfasst die folgenden Elemente: Die J2EE Plattform, die J2EE Referenzimplementierung (J2SDKEE), die J2EE Compatibility Test Suite und das J2EE Application Programming Model (J2EE Blueprints). Auf die J2EE Plattform und die J2EE Referenzimplementierung wird im

[12] Turau, Saleck, Schmidt S.4

[13] Vgl. Turau, Saleck, Scmidt /J2EE/ S.1 ff

folgenden näher eingegangen. Die J2EE Referenzimplementierung ist frei erhältlich und soll in frühen Phasen von J2EE Projekten zur Prototyperstellung eingesetzt werden. Eine kommerzielle Verwendung ist nicht erlaubt. Die J2EE Compatibility Testsuite soll sicherstellen das ein J2EE Produkt mit den J2EE Spezifikationen kompatibel ist. Sie soll die Portabilität von J2EE Anwendungen zwischen den verschiedenen J2EE Produkten erhöhen. Das J2EE Application Programming Model definiert ein Standardprogrammiermodell für die Entwicklung mehrschichtiger, verteilter Anwendungen auf J2EE Basis. Es wird die sinnvolle Kombination von verschiedenen J2EE Technologien zu einer Anwendung beschrieben.

Die J2EE Plattform basiert auf einer Drei-Schichten Architektur die in Abbildung 8 dargestellt ist. Hierbei gibt es drei Arten von Anwendungskomponenten. Diese unterteilen sich in Client-, Web- und Enterprise Java Beans (EJB) Komponenten. Client Komponenten stellen eine grafische Benutzeroberfläche bereit. Sie können sowohl als Applets oder als eigenständige Java Anwendungen realisiert sein. Webkomponenten können Java Server Pages (JSP) oder Java Servlets sein. Sie werden in dem entsprechenden Container eines J2EE Servers ausgeführt. Generell erzeugen die Webkomponenten HTML oder XML Dokumente die dann von den Webbrowsern angezeigt werden können. In EJBs wird die Geschäftslogik einer Anwendung programmiert. EJBs können sowohl von Thin- als auch Rich-Clients angesprochen werden. Der Entwickler eines EJBs definiert ein Remote-Interface, dass die öffentlichen Methoden der Komponente enthält. Zur Laufzeit des Clients wird über ein JNDI-Lookup die Referenz zum EJB-Home Interface ermittelt, um über den Client-Stub auf die eigentliche Komponente in der die Geschäftslogik programmiert ist zuzugreifen. Das EJB-Home Interface definiert die Methoden, die ein Client aufruft, um ein EJB zu finden oder zu erzeugen. EJB unterteilen sich in Session und Entity Beans. Session Beans können einen Status haben (stateful Beans) oder statuslos sein (stateless Beans). Session Beans beschreiben einen Programmablauf. Ein typischer Ablauf, der oft als Beispiel genommen wird, ist das "Kaffee kochen". Entity Beans sind Entitäten wie zum Beispiel ein "Mitarbeiter", der Kaffee kochen möchte, eine Kaffeemaschine, etc. also reelle Objekte einer Umgebung. Entity Beans werden persistent in einer Datenbank abgelegt. Das Speichern kann automatisch durch den Application Server erfolgen (container managed) bzw. durch das Bean (bean managed).

4.3 J2EE Infrastruktur

In der J2EE Architektur gibt allgemeine Dienste und Funktionen für die oben genannten Komponenten. Diese umfassen Transaktions- und Zustandsverwaltung, Multithreading, Ressourcepooling und weitere Systemaufgaben. In J2EE wird die Bereitstellung dieser Funktionalitäten von Containern übernommen. Container bilden die Schnittstelle zwischen Komponenten und den darunterliegenden plattformspezifischen Funktionen. Bevor eine Komponente ausgeführt werden kann, muss sie zu in einer J2EE Anwendung kompiliert und an den entsprechenden Container verteilt werden. Während der Verteilungsprozedur müssen bestimmte Parameter für den Container definiert werden. Diese erlauben die Anpassung des Containers an die Komponente. Dies umfasst Sicherheit, Transaktionsmanagement, Java Naming and Directory Interface (JNDI) Lookups und Remote Connectivity. Wie erwähnt müssen J2EE Komponenten zu J2EE Anwendungen kompiliert werden bevor sie benutzt werden können. Jede Komponente, dazugehörende Grafiken, HTML Dateien und Utility Klassen werden zusammen mit dem Deployment Descriptor in einem Modul zusammengefasst. Eine J2EE Anwendung kann aus einem oder mehreren Modulen bestehen. Der Deployment Deskriptor ist textbasiert und in XML kodiert. Er enthält Informationen über Sicherheitsrichtlinien und Transaktionsattribute. Im Resultat wird eine JAR Datei mit einer .ear Erweiterung erzeugt, das Enterprise Archive.

4.4 J2EE Referenzimplementation

Die J2EE Referenzimplementierung (J2EE SDK) enthält einen J2EE Applikationsserver, einen Webserver, eine relationale Datenbank, J2EE APIs und eine Sammlung von Entwicklungs- und Deployment-Tools. Der Webserver bietet Dienste für einen oder mehrere Web Container. Als Beispiel soll das HTTP Message Handling genannt werden, das vom Webserver angeboten und von einem Webcontainer benutzt wird, der eine Webkomponente hostet. Es wird kein bestimmter Webserver benötigt, somit können im Gegensatz zu .NET, welches nur den IIS unterstützt, unterschiedliche J2EE Produkte benutzt werden. Eine relationale Datenbank bietet persistente Datenspeicherung. Es wird kein bestimmtes Produkt benötigt und es können unterschiedliche J2EE Produkte verwendet werden. Für das Ablaufen des J2EE SDKs wird die Java 2 Standard Edition (J2SE) benötigt. J2SE enthält Core APIs für das Schreiben von J2EE Anwendungen, Entwicklungswerkzeuge und die Java Virtual Machine (JVM). Die wichtigsten APIs sind nachfolgend aufgeführt.

API	Beschreibung
EJB 2.0 – Enterprise Java Beans	Kapseln die Geschäftslogik und implementieren die vorgegebenen Schnittstellen. Werden in einem J2EE-konformen Application Server installiert und greifen auf dessen Dienste zurück, um beispielsweise Persistenz, Transaktionssicherheit, Zugriffsschutz und Lastverteilung zu realisieren.
JDBC – Java Database Connection	Funktionieren mit jedem Datenbankserver, für den ein entsprechender Java-Treiber verfügbar ist. Vorraussetzung ist, dass keine SQL-Statements, die spezifisch für einen bestimmten Hersteller sind, benutzt werden.
JCA – J2EE Connector Architecture	Definiert eine Standardarchitektur um J2EE Anwendungen mit heterogenen Legacy Systemen zu verbinden. Von den Herstellern werden Ressource Adapter angeboten. Wenn ein solcher existiert kann auf das System von jedem J2EE Produkt zugegriffen werden.
JMS – Java Message Service 1.0	API um Messaging Funktionalität zu implementieren. Erlaubt Anwendungen Nachrichten zu erzeugen, zu Senden, zu Empfangen und zu Lesen. Zudem wird die asynchrone Kommunikation ermöglicht.
JavaIDL	Bietet Interoperatibiltät mit CORBA. Includes ein ORB Implementation und ein IDL to Java Compiler
RMI – Remote Method Invocation	Mechanismus für entfernte Funktionsaufrufe. Java-Applikationen können mittels RMI über das Netzwerk auf Objekte einer anderen Virtual Machine zugreifen.
JNDI – Java Naming Directory Interface	Erlaubt den Zugriff auf Verzeichnisdienste wie LDAP.
XML APIs	Für den Zugriff auf XML-Dateien enthält J2EE die Schnittstellen für SAX, DOM und XSLT. Als separaten Download bietet Sun das XML-Pack an, das neben XML-Parsern die APIs für XML-basierte Web Services mit SOAP, WSDL und UDDI enthält. Mit der nächsten Version von J2EE werden diese Bibliotheken Bestandteil der Plattform. Weitere Klassenbibliotheken bietet Sun auf seinen Entwicklerseiten als optionale Packages.

Tabelle 2 - J2EE APIs[14]

4.5 Webservices und J2EE

J2EE wurde bisher hauptsächlich für die Entwicklung unternehmensweiter Anwendungen eingesetzt. Es kann aber auch als Plattform für die Entwicklung von

[14] Vgl. Loviscach, Schulz, Violka /Sunspiration/

Webservices genutzt werden. Dabei kommen im wesentlichen zwei Technologien zum Einsatz: XML und Java. XML ist ein Kommunikationsstandard. J2EE bietet XML Unterstützung über verschiedene APIs. Geschäftslogik- und Präsentationsschicht werden mit den J2EE APIs entwickelt. Unterstützung für bestehende Applikationen wird ebenso über APIs geboten. Bestehender J2EE Code kann nahtlos integriert und benutzt werden. Die Webserviceunterstützung ist in SunONE zum jetzigen Entwicklungsstand nicht so ausgereift wie in Microsoft .NET.

5 Vergleich

SunONE und .NET sind sehr ähnliche Technologien. Aus diesem Grund werden zunächst die Gemeinsamkeiten beider Ansätze verglichen. Nachfolgend werden die Vor- und Nachteile der jeweiligen Frameworks anhand eines Kriterienkatalogs diskutiert.

5.1 Kriterien

Beide Architekturen werden im Hinblick auf die Webservice Unterstützung verglichen.

Technologie	**.NET**	**SunONE**
Unterstützung		
Verteilungsprotokoll	DCOM, SOAP	RMI/IIOP
Firewall	ISA*	nicht spezifiziert
HTML Seitencaching	ISA*, ASP.NET	nicht spezifiziert
Präsentationsschicht		
Client GUI Applikationen	Windows Forms	Swing
Browserintegration	ActiveX Controls	Java Applets
Programmiermodell	ASP.NET	Servlets, JSP
Webserver	IIS	iPlanet Webserver, J2EE Referenzimplementierung, weitere verfügbar
Middleware		
Komponenten	COM+, .NET managed Components	EJB
Hochverfügbarkeit, Lastverteilung, Clustering, Management	NLBS, ACS, andere	Kombination von Applikations-Server, Webserver und Load Balancer
Sicherheits API	COM+ Security Call Context	JAAS
Message Queue API	MSMQ	JMS 1.0
Asynchronous components	Queued (COM+)	Message driven beans (EJB 2.0)
Verzeichnisdienst	ADSI	JNDI
Transaktionsmanagement	MS-DTC	JTA, JTS, Tuxedo, CICS
Datenschicht		
Relationale DB API	ADO.NET	JDBC 2.0
Hierarchische DB API	ADO.NET	-
RDBMS	SQL Server, viele unterstützt	nicht spezifiziert, JDBC Treiber für bekannte Produkte verfügbar
Legacy Integration	HIS (?), COM Transaction	JMS, JCA, JNI, CORBA,

	Integrator, MSMQ, Biztalk Server, Webservices	Webservices
Framework Technologien		
Framework	.NET Framework	Java Bibliotheken
eCommerce Framework	Commerce Server*	-
B2B orchestration	BizTalk Server*	-
Weitere Merkmale		
Laufzeitumgebung	CLR	JVM
Unterstützte Programmiersprachen	Microsoft unterstützt VB, C#, und managed C++, es sind ca. 25 weitere Sprachen verfügbar	Java
Code Repräsentation	MSIL	Java Bytecode
Unterstützte Plattformen	Windows	jede etablierte Plattform
Programmierwerkzeuge	Visual Studio.NET	Forte for Java 3.0

Tabelle 3 -.NET und SunONE[15]

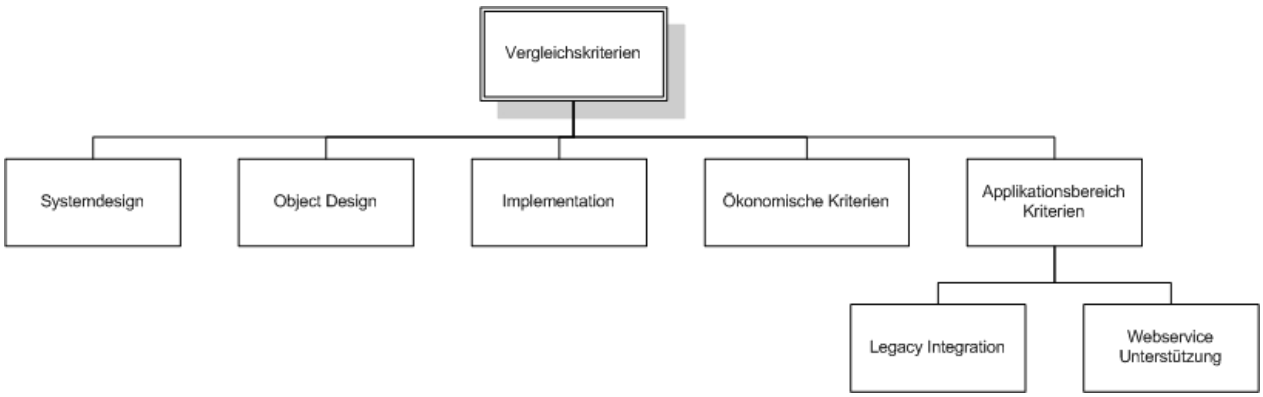


Abbildung 9 - Vergleichskriterien

5.2 Systemdesign

5.2.1 Unterstützte Hardware

Microsoft .NET benötigt ein Windows basiertes System. SunONE hingegen läuft auf fast jeder Plattform. Beide Architekturen unterstützen Embedded Systeme. Microsoft bietet mit Windows XP embedded und Windows CE.NET zwei Alternativen[16]. Windows XP embedded ist für High-End Embedded Systeme konzipiert, Windows CE.NET hingegen kann an die Anforderungen der jeweiligen Plattform angepasst

[15] Vgl. Farley /.NET vs. SunONE/ und Sessions /J2EE vs. NET/

[16] Lüders /.NET embedded/

werden. Die SunONE Architektur beinhaltet die Java 2 Micro Edition (J2ME) welche einen Service Container für Consumer und Embedded Systems bietet.

5.2.2 Sicherheit

Die Sicherheitsfunktionen von Microsoft .NET sind in den Basisklassen des Frameworks integriert. Es werden dabei Authentifikation, Autorisation, Datenschutz und Auditing abgedeckt. SunONE bietet verschiedene Sicherheits-APIs. Java Authentication and Authorization (JAAS) implementiert Authentifikationsmechanismen und Zugriffsrechte für die Benutzer. Java Cryptography Extension (JCE) bietet ein Framework für Verschlüsselung. Schlüsselgeneration und Schlüsselmanagement. Java Secure Socket Extension (JSSE) ermöglicht die sichere Internetkommunikation.

5.2.3 Datenzugriff

ADO.NET ist eine neue Datenzugriffs-API in .NET, die XML für Datenübertragung von der Datenquelle zu den ADO.NET Komponenten benutzt. Somit kann alles was XML Daten erzeugt als Datenquelle verwendet werden. Microsoft .NET bietet Zugriff auf den MS SQLServer.NET und implementiert ebenso den Zugriff über OLE DB. Jede Datenbank die den OLE DB Zugriff bietet kann über ADO.NET erreicht werden.
Der Datenzugriff in SunONE ist über JDBC möglich. Datenbanken werden über Treiber angebunden, welche für die bekannten Datenbanken verfügbar sind.

5.2.4 Performance

Load Balancing Beide Architekturen bieten Methoden für das Load Balancing und Clustering auf Webserver Farmen. Microsoft .NET verwendet hierfür das MS Application Center 2000 und SunONE den iPlanet Application Server.

Vorkompilierte Webseiten Beide Architekturen beinhalten die Technologie für dynamische Webseiten. Microsoft .NET benutzt hierfür ASP.NET und SunONE JSP. ASP.NET und JSP halten eine Seite nach der ersten Kompilation im Speicher und erlauben die Wiederverwendung dieser Seite ohne Neukompilation. ASP.NET erlaubt ebenso die Batch-Kompilierung was die Performance noch weiter steigern kann.

5.2.5 Abhängigkeit

Fehlerbehandlung In beiden Architekturen wird ein ähnlicher Ansatz verwendet. Die Fehlerbehandlung wird durch `Exception` Objekte erreicht. Die Objekte bieten Informationen über den aufgetretenen Fehler und ermöglichen es dem Programmierer den Fehler zu behandeln anstatt das Programm zu beenden. Beide benutzen ein `try, catch, finally` Konstrukt zur Fehlerbehandlung.

5.2.6 Kosten

Produkte Visual Studio .NET ist für $1600 Dollar verfügbar. Forte for Java 3.0 ist in der Community Version frei verfügbar und kostet in der Enterprise Version $1995. In der Enterprise-Version ist die Webservice Unterstützung enthalten. Der Vergleich der Serverprodukte führt zu keinem sinnvollen Ergebnis, da unterschiedliche Lizenzmodelle verwendet werden.

5.2.7 Wartung

Skalierbarkeit Eine Plattform ist skalierbar wenn eine Erhöhung der Hardwareressorcen zu einer linearen Erhöhung der maximalen User-Load bei gleichbleibender Antwortzeit führt. Da .NET nur Windows-basierte Systeme unterstützt, werden eventuell aufgrund der Prozessorbeschränkung mehr Maschinen benötigt als eine vergleichbare SunONE Installation. Diese Vielfalt kann für eine Organisation schwierig zu verwalten sein.

Portierbarkeit Microsoft .NET ist nur für Windows Plattformen verfügbar und Microsoft plant keine Unterstützung weiterer Plattformen. Die CLI wird auf FreeBSD Unix implementiert. Weiterhin gibt es OpenSource Initiativen wie das MONO Projekt[17] um die ECMA Spezifikationen auf Linux umzusetzen. Die Entwickler um den Gnome-Gründer Miguel de Icaza planen, bis Mitte 2002 eine benutzbare Linux-Version fertigzustellen[18]. Zentrale Bausteine des Projekts "Mono" wie der Compiler für die .NET-Programmiersprache C# sind zu etwa 30 Prozent fertig. Microsoft will keine Linux-Version von .NET entwickeln, plant aber zumindest eine Beispielimplementierung als "Shared Source".[19] Diese Initiativen werden teilweise von Microsoft unterstützt[20]. SunONE hingegen ist ebenso wie J2EE plattformunabhängig. Einige Produkte, insbesondere die iPlanet Server sind nur für Windows und drei Unix Derivate verfügbar. Somit ist die Portierbarkeit basierend auf den iPlanet Produkten ebenso eingeschränkt.

[17] MONO Project – http://www.ximian.com/

[18] Heise /.NET als OpenSource/

[19] Heise /Shared Source/

[20] Heise /.NET für Linux/

Lesbarkeit Teile von SunONE sind als Quellcode verfügbar. Dies schließt speziell die JVM und J2EE Referenzimplementierung mit ein. Für Microsoft .NET ist momentan kein Quellcode verfügbar. Microsoft hat aber eine Referenzimplementierung des ECMA CLI Standards angekündigt.

5.3 Object Design

5.3.1 Klassenbibliothek

Die Basisklassen welche die in SunONE und Microsoft .NET angeboten werden, basieren auf einem identischen Satz von Kernklassen. Diese unterstützen Eingabe- und Ausgabefunktionalität, Threadverwaltung, Netzwerk, Sicherheitsmanagement, Reflection Funktionalität und Collections.

5.3.2 Komponentenarchitektur

Beide Architekturen unterstützen die Entwicklung von Unternehmensanwendungen durch einen komponentenbasierten Ansatz. In Microsoft .NET repräsentiert jede Assembly eine „Managed Component“. „Managed“ meint hier dass die Komponente durch CLR verwaltet wird. Zudem ermöglicht .NET Serviced Components. Basierend auf COM+ unterstützen Serviced Components, die von `System.ServicedComponent` abgeleitet werden müssen, Object Pooling oder Transaktionen. Serviced Components werden innerhalb der CLR ausgeführt. SunONE bietet eine strikte Spezifikation für Komponenten mit EJB 2.0. Eine EJB ist eine Komponente einer verteilten, transaktionsorientierten Applikation. EJBs werden von einem Container verwaltet und unterstützen die Benutzung von Middleware-Diensten wie Transaktionen, Sicherheit und Datenbankverbindungen.

5.3.3 GUI Unterstützung

Lightweight-Clients Für Lightweight-Clients benutzt Microsoft .NET ASP.NET und Web-Forms. Web-Forms sind ähnlich zu Windows Forms, aber sie rendern sich selbst in der entsprechenden Auszeichnungssprache, wie HTML. .NET unterscheidet HTML Controls und Web-Form Controls. HTML Controls ahmen typische HTML Elemente nach. Web-Controls bietet erweiterte Funktionalität, zum Beispiel Data Binding auf Serverseite. Zudem können sie in jeder Auszeichnungssprache zu gerendert werden. ASP.NET und Web-Forms laufen nur mit dem IIS als Webserver. Ähnliche Funktionalität wird in SunONE von JSP und Java Servlets geboten. Aber eine Technologie ähnlich den Web-Forms existiert bis jetzt nicht. Sun kündigte das Java

Server Faces UI Kit an. Das Kit soll dieselbe Funktionalität besitzen wie die Webforms in .NET.

Heavyweight-Clients Heavyweight Clients können als alleinstehende Applikationen auf einem Client laufen. .NET bietet hierfür die Windows-Forms. Windows-Forms kombinieren die Elemente bereits existierender Frameworks wie MFC und die Visual Basic Komponenten. Windows-Forms sind nur für Windows verfügbar. SunONE bietet das Swing-Paket zur Entwicklung grafischer Benutzeroberflächen. Swing bietet ein komplettes Set von User-Interface Elementen welche vollständig in Java geschrieben sind.

5.4 Implementation

5.4.1 Entwicklungswerkzeuge

Beide Architekturen bieten vollständige IDEs. .NET beinhaltet VisualStudio .NET. Dieses bietet in der Grundausstattung Entwicklungsunterstützung für C#, VB.NET und Managed C++. Es ist mit dem IIS integriert, was die Verteilung von Applikationen erleichtert. Der IIS ist der einzige unterstützte Webserver. SunONE bietet Forte for Java. Forte ist eine modulare und erweiterbare in Java geschriebene Plattform. Sie beinhaltet typische Eigenschaften wie GUI Builder, Quelltexteditor, Debugger und Skriptwerkzeuge.

5.4.2 Laufzeitumgebung

Beide benutzen eine virtuelle Maschine zur Ausführung ihrer Programme. .NET basiert auf der CLR und SunONE benutzt die JVM.

Code Repräsention In SunONE wird der Java Quellcode zu Java Bytecode kompiliert und in einer `.class` Datei gespeichert. Viele Applikationen bestehen aus mehreren .class Dateien die in einem `.jar` Archiv kombiniert wurden. Die JVM wird mit der `main` - Methode der spezifizierten Startklasse aufgerufen. In .NET wird der Quellcode der verschiedenen Quelldateien in MSIL kompiliert. Zusätzlich werden Metadaten generiert. MSIL Code und Metadaten bilden das .NET Portable Executable (PE) Format. Wird eine .NET PE Datei ausgeführt, wird die CLR geladen. Diese lädt die Applikation Startklassendatei und ruft die Main-Funktion auf.

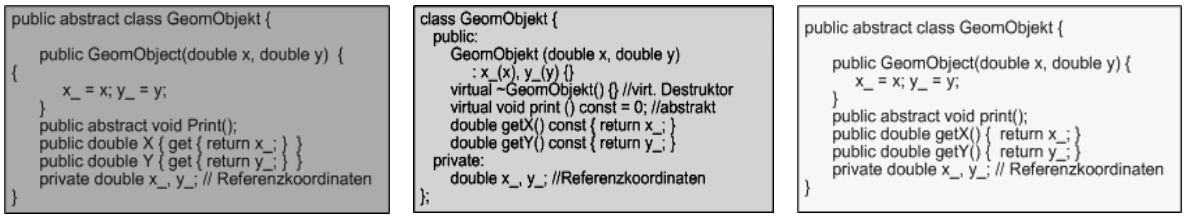
Klassenlader Ein Klassenlader ist verantwortlich für das Laden von Java Bytecode oder MSIL. Ein Fehler wird in beiden Architekturen geworfen, wenn die Klassenlade-Prozedur fehlschlägt

Codeüberprüfung Der geladene Code wird dann auf verschiedene Eigenschaften hin überprüft. Für Java Bytecode bedeutet dies Prüfungen auf gültige JVM Anweisungen, gültige Anweisungsparameter und gültiges Code-Branching. Die .NET PE Überprüfung bezieht sich auf wohlgeformte Metadaten und gültiges MSIL. Zusätzlich kann MSIL unter Benutzung einer PKI erstellt werden um sicherzustellen, dass der Code nicht verändert wurde. Die CLR wird in diesem Fall öffentliche und private Schlüssel dieser Dateien prüfen.

JIT Compiler Nachdem der Code geladen und verifiziert wurde, muss er in nativen Code kompiliert werden um ausgeführt zu werden. Die Kompilation in nativen Code kann in zwei Phasen auftreten. Vor oder nach der Ausführung. Beide Architekturen benutzen die Ausführung zur Laufzeit als Standardeinstellung. Im Fall von Java nimmt die JVM, sobald eine `.class` Datei geladen und verifiziert wurde, die gesamte Klasse konvertiert sie in nativen Code und cachet sie im Speicher. In .NET wird der JIT aufgerufen wenn eine Methode einer Klasse aufgerufen wird und diese Methode nicht schon im nativen Code vorliegt. In beiden Fällen, .NET und SunONE kann der native Code auch vor der Ausführung erzeugt werden. Dieser native Code wird dann im Allgemeinen schneller ausgeführt als zur Laufzeit kompilierter Code.

5.4.3 Programmiersprachen

SunONE ist Java zentriert. In der Theorie, ist der JVM Bytecode sprachenneutral, aber in der Praxis wird er nur mit Java benutzt. Komponenten wie EJB müssen in Java geschrieben werden. Die Lücke zu Komponenten die in anderen Sprachen geschrieben wurden können nur mit Technologien wie CORBA, JCA oder Webservices geschlossen werden. In .NET wird theoretisch jede Programmiersprache unterstützt die MSIL Code erzeugen kann. Zur Zeit werden ca. 15 Programmiersprachen unterstützt. Die für .NET bevorzugte Sprache ist aber C#. C# weist große Ähnlichkeit zu Java auf. Mit dem Unterschied dass es nur eingeschränkt portabel ist. Dass Microsoft nicht auf Java als Grundlage für .NET zurückgegriffen hat, liegt an der gerichtlichen Auseinandersetzung mit Sun über Java. Die C# Sprachspezifikation verweist oft die C++ und kein einziges Mal auf Java. Trotzdem erinnert C# mehr an Java als an C++. Um die Ähnlichkeiten zwischen C++ und Java aufzuzeigen ist in Abbildung 10 ein Quellcodevergleich zwischen C#, Java und C++ dargestellt. Diese Sprachen werden in nachfolgenden Tabelle verglichen und bewertet.



```
public abstract class GeomObjekt {

    public GeomObject(double x, double y)  {
{
        x_ = x; y_ = y;
    }
    public abstract void Print();
    public double X { get { return x_; } }
    public double Y { get { return y_; } }
    private double x_, y_; // Referenzkoordinaten
}
```

```
class GeomObjekt {
  public:
    GeomObjekt (double x, double y)
        : x_(x), y_(y) {}
    virtual ~GeomObjekt() {} //virt. Destruktor
    virtual void print () const = 0; //abstrakt
    double getX() const { return x_; }
    double getY() const { return y_; }
  private:
    double x_, y_; //Referenzkoordinaten
};
```

```
public abstract class GeomObjekt {

    public GeomObject(double x, double y) {
        x_ = x; y_ = y;
    }
    public abstract void print();
    public double getX() {  return x_; }
    public double getY() {  return y_; }
    private double x_, y_; // Referenzkoordinaten
}
```

Abbildung 10 – C#, Java und C++ Quellcodebeispiel[21]

	C#, .NET	Java	C++
Version	ECMA-334/335	J2SDK 1.4	ISO/IEC 14882
Quellcode und Text			
Präprozessor	-	-	✓
Eine Datei pro öffentliche Klasse	wahlweise	✓	Wahlweise
Bedingte Compilierung	✓	(eingeschränkt)	✓
Dokumentation in Quellcode	-[22]	✓	-[23]
‚Attribute'	✓	-	-
Assert	✓[22]	✓	✓
Overflow-Test	Für ganzzahlige Typen	-	-
Exceptions spezifizieren	-	Teilweise verpflichtend	Wahlweise
Typen und Operatoren			
Erzwungene Initialisierung	✓	✓	-
Vorzeichenlose Zahlentypen	✓	-	✓
Aufzählungstypen (enum)	✓	-	✓[23]
Dezimal-Gleitkomma	✓	-	-
Gleitkommaarithmetik maschinenunabhängig	-	✓ (strictfp)	-
Mehrdimensionale Arrays zur Laufzeit anlegen	✓	✓	Nur erste Dimension variabel oder eigene Klasse
Grenzenprüfung bei Arrays	✓	✓	(durch Überladen)
Zeiger (arithmetik)	✓ (in unsafe Bereichen)	-	✓
Operatoren überladen	teilweise	-	✓
Get und set als Zuweisung	✓ (Properties)	-	Eingeschränkt (a.x = 1;)
Implizite Konvertierungsoperatoren	✓	-	✓
Objektorientierung			
Globale Funktionen	-	-	✓
Mehrfachvererbung	(Interfaces)	(Interfaces)	✓
Nicht-virtuelle Methoden	✓	-	✓
Virtuelle Methoden explizit überschreiben	✓	-	wahlweise
Kovarianz von Arrays	✓	-	-
Kovarianz bei Rückgabetypen	-	-	✓

[21] Breymann, Loviscach /C-Klasse/

[22] nicht ECMA-Standard, aber Teil des .NET Framework SDK

[23] nicht ISO-Standard, aber Lösung verfügbar

Ressourcenverwaltung			
Garbage Collector	✓	✓	-[23]
Objektartige Werttypen	✓	-	✓
Automatisch aufgerufener Destruktor	eingeschränkt	-	✓
Parameterübergabe			
Parameterübergabe per Referenz	✓ (ref)	-	✓ (&)
Variable Parameteranzahl	✓ (params)	-	✓ (...)
Unveränderliche Parameter und Objekte	-	-	✓
Reine Eingabe-Parameter per Referenz	-	-	✓ (const &)
Reine Ausgabe-Parameter per Referenz	✓ (out)	-	-
Sonstiges			
Generische Programmierung	eingeschränkt	eingeschränkt	✓ (Templates)
Automatisches ‚Boxing'	teilweise	-	Selten nötig
Container / Collections	✓	✓	✓
Foreach-Schleife	✓	-	Eingeschränkt
Finally-Klausel für Excpetions	✓	✓	Nicht nötig wegen Destruktoraufruf
Events	✓	✓	-[23]
GUI, Grafikroutinen	-[22]	✓	-[23]
Internet (Sockets usw.)	✓	✓	-[23]
‚Delegates'	✓	-	Eingeschränkt
Threads	✓	✓	-[23]
Objektserialisierung	-[22]	✓	-[23]
Remote Procedure Call	✓	✓	-[23]
Reflection	✓	✓	Sehr eingeschränkt (RTTI)
Reguläre Ausdrücke	-[22]	✓	-[23]
Codesicherheit	✓	✓	-[23]
Bewertung			
Geschwindigkeit	o	o	++
Schutz gegen Programmierfehler	-	-	--
Erlernbarkeit	+	++	O

Tabelle 4 - C#, Java und C++ Vergleich[24]

5.4.4 Gemeinsame Bibliotheken

In .NET wird eine gemeinsame Bibliothek Shared Assembly (SA) genannt. Um von mehreren Clients benutzt werden zu können, muss eine Shared Assembly im Global Assembly Cache (GAC) registriert sein. Sobald es registriert ist verhält sich die SA wie eine Systemkomponente. Um registriert zu werden, werden Informationen über den Originalhersteller und Version benötigt. Es ist möglich verschiedene Versionen derselben Assembly auf einem Host zu registrieren. In SunONE sind gemeinsame

[24] Breymann, Loviscach /C-Klasse/

Bibliotheken als JAR Dateien realisiert. JAR Dateien können eine hohe Bandbreite an Funktionalität unterstützen, einschließlich digitaler Unterschriften, Versionskontrolle und Package Sealing. Die notwendige Information für diese Funktionalität wird von einer Manifestdatei in der JAR Datei gespeichert. Klassen in JAR Dateien werden automatisch geladen sobald sie angefordert werden. JAR Dateien müssen im Klassenpfad spezifiziert werden. Das ist der Pfad in dem die JVM nach Klassen sucht.

5.4.5 Dokumentation

Microsoft und Sun bieten eine große Menge an Dokumentation für ihre Frameworks. Die Dokumentation auf beiden Seiten enthält Walk-Throughs, Tutorials für Anfänger und ausführliche Dokumentation zu den verschiedenen Teilen der jeweiligen Frameworks.

Code Documentation Sun bietet Javaprogrammieren eine Dokumentationsgenerator, den javadoc an. Der Generator durchsucht den Javacode nach relevanten Informationen wie Kommentaren und produziert Text-Output. Der Benutzer kann den Inhalt und das Format der Ausgabe des javadoc Tools festlegen. Der Standard ist HTML. Alle verfügbaren Java APIs sind mit javadoc dokumentiert. .NET bietet ein ähnliches, aber weniger leistungsfähiges Tool für Programme in C# an. Der C# Compiler kann eine XML Datei, ausgehend von speziellen Dokumenten-Tags, erzeugen. Weiteres Abarbeiten der XML Datei ist nicht möglich. Die Dokumentation der .NET Framework Klassen wird innerhalb der IDE mittels des proprietären Microsoft Dokumenten Explorers angeboten.

5.5 Ökonomische Kriterien

Für den Vergleich können nicht nur technische Kriterien berücksichtigt werden. Die Vergangenheit zeigt, dass nicht nur die Technologie allein über die Marktakzeptanz entscheidet. Aus diesem Grund werden auch ökonomische Kriterien in den Vergleich miteinbezogen.

5.5.1 Marktstellung

Microsoft ist bekannt für seine guten Marketingmöglichkeiten und für die Durchsetzung ihrer Produkte. Seit der Einführung von .NET vermarktet Microsoft alle seine Produkte im .NET Kontext. Daraus ergibt sich ein Hype um .NET, wobei die Inhalte von .NET oft nebulös erscheinen. Sun hat seine SunONE Initiative kurz nach der Einführung von .NET gestartet. Die Webservice Einbindung kam aber erst nach der Microsoft .NET

Initiative. J2EE ist der Kern von SunONE und wird nicht nur von Sun vermarktet sondern auch von mehr als 50 Lösungsanbietern, welche Produkte auf der J2EE Basis anbieten.

5.5.2 Reife

Beide Architekturen resultieren aus der Evolution bereits existierender Produkte. Webservice Unterstützung ist aber eine völlig neue Entwicklung. .NET ist auf bereits existierenden Komponentenarchitekturen wie COM+ und Windows DNA aufgebaut. Die CLR als der Kern von .NET ist eine völlig neue Entwicklung ebenso wie C#. Somit sind die Hauptbestandteile von .NET noch unausgereift. Die Basisarchitektur von SunONE, J2EE, hat seine Fähigkeiten in verschiedenen Produkten wie Internet Applikationsservern bewiesen. Der Standard ist Version 1.3 verfügbar. Der Webservice Unterstützung ist neu und bisher nur in Teilen angekündigt. Als Beispiel wurde der SOAP Unterstützung erst im Dezember 2001 in Java Pack XML Winter01 Bundle veröffentlicht. Die Unterstützung für Standards wie SOAP und WSDL ist sehr unausgereift. Die Unterstützung für SOAP in der Forte IDE ist nur als Technologie Preview verfügbar.

5.5.3 Verfügbarkeit

Das SunONE StarterKit beinhaltet Dokumentation, Code Beispiele, Entwicklungstools und Evaluierungsversionen für die iPlanet Server Produkte. Alle Teile sind zum Download verfügbar. Die CD Version beinhaltet 4 CDs und ist für $19,95 verfügbar. Visual Studio ist im Handel erhältlich.

5.5.4 Hersteller

.NET ist eine Ein-Anbieter Lösung. Alle Teile sind nur von Microsoft verfügbar. Immerhin ermutigt Microsoft seine Partner Add-On Produkte wie Komponentenbibliotheken anzubieten. SunONE ist im allgemeinen ebenso eine Ein-Anbieter Lösung. Aber viele von bereits verfügbaren Drittanbieter-Produkten können integriert werden, da SunONE auf dem J2EE Standard basiert. Eine Vielzahl von Legacy Anwendungen wie Internet Applikationsserver basieren auf J2EE, zum Beispiel die Produkte von BEA und IBM. Somit ist die Integration dieser Legacy Komponenten einfacher als in .NET.

5.6 Integration

5.6.1 Legacy Integration

In SunONE basiert die Legacy Integration auf JCA. Die JCA definiert eine Standardarchitektur um heterogene Legacy Systeme mit der J2EE Plattform zu verbinden. Die Spezifikation ist in Version 1.0 verfügbar und befindet sich im Public-Review. Daneben sind viele Adapter gemäss der Spezifikation verfügbar. Darunter Adapter für SAP R/3, Peoplesoft, Siebel, Oracle und IBM CISC. Microsoft .NET bietet die Integration vorangegangener proprietärer eigener Technologien wie MS DNA direkt im .NET Framework an. Die Integration hostbasierter Systeme übernimmt der MS Integration Server 2000. Des weiteren bietet Microsoft den Biztalk-Server für den Aufbau von Geschäftsprozessen und die Applikationsintegration. Biztalk bietet Unterstützung für XML und EDI. Microsoft selbst bietet einen Adapter zu Biztalk für SAP und MQseries an. Andere Adapter werden von Partnern entwickelt.

5.6.2 CORBA Unterstützung

SunONE bietet starke CORBA Integration. EJB, RMI over IIOP, JavaIDL und Java Transaction Service (JTA) benutzen die CORBA Technologie. Das macht CORBA zu einem integralen Bestandteil von J2EE. Die EJB Kommunikation wird über RMI over IIOP implementiert. RMI over IIOP beinhaltet die volle Funktionalität eines CORBA Object Request Brokers (ORB). Java IDL beinhaltet einen CORBA Object Request Broker, der in jeder Version der Java 2 Plattform enthalten ist.

Microsoft bietet keine direkte CORBA Unterstützung. Es gibt aber eine indirekte Unterstützung über COM / CORBA Bridging.

5.7 Webservice

5.7.1 Webservice Werkzeuge

IDE Die Webservices-Unterstützung ist der Visual Studio .NET IDE nahtlos integriert. Es bietet beides, Unterstützung für Entwicklung von Webservices und für den Zugriff auf Webservices. Die Entwicklung eines Webservice in .NET resultiert in einer DLL, welche nur auf dem Internet Information Server installiert werden kann. Um auf einen Webservice zugreifen zu können sind mehrere Schritte notwendig. Zuerst muss eine Referenz zu der Webservice-Beschreibung angeboten werden. Dann wird ein Proxy ausgehend von der Webservice-Beschreibung erzeugt. Auf diesen Proxy kann schließlich zugegriffen und als lokale Komponente benutzt werden. Der Proxy ist verantwortlich um den Remote Webservice via SOAP aufzurufen.

Forte for Java, die Entwicklungsumgebung von SunONE, integriert ebenfalls die Webserviceunterstützung. In Version 3.0 wird keine direkte Unterstützung für SOAP angeboten. Stattdessen werden proprietäre XML Nachrichten für die Kommunikation mit einem Webservice genutzt. Vor kurzem veröffentlichte Sun ein SOAP RPC Technologie Preview, welches erstmals SOAP RPC Unterstützung über die Apache SOAP Implementation in Forte for Java integriert. Assistenten unterstützen das Deployment von EJB Geschäftsfunktionen unter Benutzung von SOAP für den Zugriff. Eine Beschreibung des Webservice kann ebenso angeboten und ein Client Proxy generiert werden.

Befehlszeilen-Werkzeuge Unterschiedliche Befehlszeilenwerkzeuge sind für die Arbeit mit Webservices in .NET verfügbar. Das Werkzeug `wsdl.exe` kann benutzt werden um Client Proxies für einen Webservice zu generieren. `soapuds.exe` bietet Unterstützung für die Webservice Kommunikation innerhalb von .NET Remoting. Der Webservice Discovery Prozess wird von dem Werkzeug `disco.exe` unterstützt, welches den .disco Web Service Discovery Ansatz umsetzt. In diesem Ansatz bietet ein Webserver die URLs der WSDL Beschreibungen von Webservices, die in einer zentralen .disco Datei gespeichert sind, an.

Werkzeuge von Drittanbietern Da SunONE java-basiert ist, können verschiedene Tools für die Webserviceentwicklung verwendet werden. Für .NET existieren keine Tools von Drittanbietern.

5.7.2 Standardunterstützung

SOAP Die Microsoft SOAP Implementierung wurde vollständig in .NET integriert. Klassen für Behandlung von SOAP Nachrichten und Kommunikation sind im Framework enthalten. Die SOAP Unterstützung in SunONE ist nicht so integriert wie die in .NET. Aber es ist schon durch das JAX Pack verfügbar und wird in Zukunft durch das Webservice-Pack erweitert werden. Die Unterstützung innerhalb der IDE wurde kürzlich wie oben bereits beschrieben in Form ein Technologie Preview vorgestellt.

WSDL WSDL ist ebenso wie SOAP stark in .NET integriert und wird unterstützt. Proxies für Webservices können von WSDL Beschreibungen mit dem `wsdl` Tool entweder statisch oder dynamisch zur Laufzeit generiert werden. In SunONE gibt es eine Java API for WSDL (JWSDL) Spezifikation, die sich im Moment im Java

Community Prozess befindet. Wenn es freigegeben wird, wird es eine API für die Manipulation von WSDL Dokumenten bieten ohne direkt mit den XML Dokumenten arbeiten zu müssen. Die volle Funktionalität kann derzeit durch JAXP erreicht werden, dann muss aber WSDL per Hand behandelt werden. Für Forte for Java gibt es ein Technologie Preview für SOAP RPC.

UDDI Microsoft bietet das UDDI Software Development Kit an, um UDDI Verzeichnisinformationen zu behandeln oder ein lokales UDDI Verzeichnis zu Testzwecken zu betreiben. Das SDK ist als Managed-Code Klassenbibliothek implementiert. welches ein einfaches Programmiermodell für die Manipulation der Request und Response Daten bietet. Die Daten sind notwendig für die Interaktion mit dem UDDI Verzeichnis von Visual Studio .NET Applikationen. Das SDK ist in der Version 1.5 verfügbar.
In der SunONE IDE gibt es keine Unterstützung für UDDI. Nur ein Web-Frontend von verschiedenen UDDI Verzeichnisanbietern kann aufgerufen werden. Für den programmatischen Zugriff auf UDDI gibt es das JAXR API. Neben anderen Verzeichnissen unterstützt JAXR auch UDDI. Ebenso werden Registering und die Suche nach Geschäfts-Webservices unterstützt.

ebXML Microsoft .NET unterstützt die ebXML Spezifikationen nicht direkt. Aber der Zugriff kann über die Klassen für die XML Unterstützung erreicht werden. SunONE bietet mehr ebXML Unterstützung als .NET. Mit JAXR kann einfach auf die ebXML Verzeichnisdienste zugegriffen werden. Der ebXML Messaging Dienst kann über JAXM benutzt werden.

5.7.3 Webservice Entwicklung

Beide Architekturen unterstützen die Entwicklung von Webservices durch Assistenten in ihren IDEs. Sowohl die Entwicklung von Webservices als auch Webservice Clients werden unterstützt. In Microsoft .NET ist diese Unterstützung nahtlos integriert. SunONE hat vor kurzem Unterstützung für die Webserviceentwicklung hinzugefügt. Zuvor war nur ein propiertärer Weg für die Generierung XML basierter Webservices verfügbar.

5.7.4 Webservice Deployment

Webservices in .NET werden auf dem Internet Information Server verteilt. Werkzeuge für das Deployment sind in Visual Studio .NET enthalten. SunONE bietet Deployment

zu verschiedenen J2EE basierten Web- und Applikationsservern, insbesondere zu dem J2EE RI Webserver und dem iPlanet Application Server. Das Webservice Deployment im proprietären ESP wird unterstützt ebenso wie SOAP basierte Webservices unter Benutzung der ApacheSOAP Implementation.

5.7.5 Gemeinsamer Kontext

In Zukunft wird die Behandlung von gemeinsamen Kontext wichtig werden. Beide Firmen haben unterschiedliche erste Ansätze für das Kontexthandling innerhalb der Webserviceinfrastruktur.

Passport Microsoft hat Passport als zuerst verfügbaren Webservice eingeführt. Passport in ein Authentifikationsdienst und baut auf der Basis von Microsoft .NET myServices auf. Grundlegende Nutzerinformationen wie Name, Adresse und email, aber auch Kreditkartendetails werden auf dem von Microsoft betriebenen Server gespeichert. Ein Nutzer meldet sich auf dem Passport Server an und seine Informationen können nun von allen am Passportsystem ? Webserviceanbietern genutzt werden. Microsoft hat auf Sicherheits- und Datenschutzbedenken reagiert und angekündigt Passport für die Interaktion mit ähnlichen Authentifikationsdiensten zu öffnen. Weiterhin denkt Microsoft darüber nach, die Operation des Passportdienstes in eine Föderation, bestehend aus Rivalen und Partnern zu übergeben. Passport arbeitet bereits und hat eine Nutzeranzahl von ca. 150 Millionen. Der Grund hierfür ist in dem Anmeldezwang bei Passport für jeden neuen MSN Teilnehmer zu suchen.

Liberty Alliance Zusammen mit seinen Partnern hat Sun mit Liberty Alliance einen ähnlichen Ansatz im September 2001 angekündigt. Liberty Alliance[25] ist eine neue Organisation und hat sich zum Ziel gesetzt ein offenes, föderatives, Single Sign On Identifizierungssystem für die digitale Wirtschaft zu erstellen.

6 Zusammenfassung

SunONE und Microsoft .NET bieten eine sehr ähnliche Menge von Produkten und Features. Der Hauptunterschied ist, dass .NET viele unterschiedliche Programmiersprachen unterstützt aber nur eine Plattform, die Windowsfamilie. SunONE ist für viele Plattformen verfügbar aber javazentriert. Die folgende Tabelle

[25] http://www.projectliberty.org

fasst die Kriterien zusammen und versucht aufzuzeigen wo SunONE und wo .NET Vorteile bzw. Nachteile haben.

Kriterium	Interpretation	ONE	.NET
System Design			
Unterstützte Hardware	Microsoft ist auf Hardware beschränkt auf der das Windows Betriebssystem läuft	+	-
Sicherheit	Eigenschaften für Implementierung sicherer Applikation ist in beiden Architekturen enthalten.	+	+
Datenzugriff	Beide Architekturen bieten einfachen Zugriff auf Datenbanken. Sun bietet aber keine kommerzielle Datenbank an.	-	+
Performance und Abhängigkeit	Beide Architekturen bieten ein ähnliches Instrumentarium für die Entwicklung robuster, zuverlässiger, hochverfügbarer, fehlertoleranter, sicherer und hochperformanter Applikationen	+	+
Wartung	Obwohl Teile von .NET als offener Standard verfügbar sind, ist SunONE offener.	+	-
Object Design			
Klassenbibliothek	Die angebotenen Klassenbibliotheken haben ähnliche Eigenschaften	+	+
Komponentenarchitektur	Beide Architekturen bieten ähnliche Unterstützung für die Komponentenentwicklung	+	+
GUI Unterstützung	Swing und Windows Forms sind beide reich ausgestattet. Aber SunONE hat kein Äquivalent zu WebForms.	o	+
Implementation			
Entwicklungswerkzeuge	Visual Studio ist einfacher zu benutzen als Forte. Beide bieten Assistenten zur schnellen Entwicklung	o	+
Laufzeitumgebung	JVM ist seit 5 Jahren verfügbar und sehr ausgereift. Die CLR ist eine neue Entwicklung und nicht im Quellcode verfügbar.	+	o
Programmiersprachen	.NET bietet Unterstützung für mehrere Programmiersprachen	-	+
Gemeinsame Bibliotheken	In beiden Umgebungen ist es einfach gemeinsame Bibliotheken zu erzeugen und zu deployen	+	+
Dokumentation	Obwohl es für beide Plattformen sehr viel Dokumentation gibt, ist die von SunONE besser, da die J2EE Referenzimplementierung im Quellcode vorliegt	+	o

Ökonomische Kriterien			
Marktstellung	Microsoft verwendet viele Ressourcen in Werbung. Sun ist etabliert und wird von vielen Herstellern unterstützt	+	+
Reife	Das gesamte .NET Framework ist eine neue Entwicklung. Die JVM ist seit 5 Jahren am Markt, aber die Webserviceunterstützung ist neu	+	-
Verfügbarkeit	Produkte im Handel	o	o
Hersteller	J2EE wird von vielen Herstellern angeboten	+	-
Integrationsunterstützung			
Legacy Integration	Der offene Ansatz des JCA übertrifft .NET	+	o
CORBA Unterstützung	Microsoft unterstützt CORBA nicht direkt. SunONE integriert die CORBA Unterstützung	+	-
Webservice			
Werkzeuge	Die Unterstützung für Webservices ist nahtlos in Visual Studio .NET integriert	-	+
Unterstützung von Standards	Die Unterstützung von Standards, mit Ausnahme von ebXML ist in .NET weiter fortgeschritten als in SunONE	-	+
Entwicklung	Die Entwicklung von Webservices ist einfacher mit Microsoft Tools	-	+
Verteilung	.NET Webservices können nur auf dem IIS deployet werden, aber das sehr einfach. SunONE gestaltet das ähnlich einfach, ist aber für unterschiedliche Web und Applikationsserver möglich.	+	o
Gemeinsamer Kontext	Obwohl Passport schon verfügbar und kostenlos ist, ist ein föderativer Ansatz im Sinne der Liberty Alliance vorzuziehen.	+	-

Tabelle 5 - Gegenüberstellung .NET und SunONE

Legende:

+ = Vorteil

- = Nachteil

o = keine Entscheidung

7 Ausblick

Microsoft hat eine starke Stellung im PC Betriebssystem- und Anwendungsmarkt und ist Marktführer im Webbrowser Markt. .NET scheint der logische ? für die Entwicklung eines Betriebssystems für das Internet zu sein. Microsoft sieht Software nicht mehr als

Produkt sondern als Dienstleistung. .NET bietet ähnliche Eigenschaften und Technologien wie das längere Zeit verfügbare J2EE. J2EE unterstützt Webservices im Moment durch XML Bibliotheken. In Zukunft werden aber noch spezielle Werkzeuge für die Webservice-Entwicklung hinzukommen.

In gewisser Weise ist die Situation vergleichbar mit der Situation im Webbrowser Markt vor 5 Jahren. Heute hat Microsoft Internet Explorer einen Marktanteil von über 75 Prozent. Hauptsächlich deshalb weil er kostenlos mit Microsoft Betriebssystemen gekoppelt wurde. Die Basisdienste in .NET myServices wie Passport sind kostenlos verfügbar. Aber in Zukunft sicher nicht.

Letzten Endes wird die Zeit den erfolgreichen Ansatz aufzeigen. Der beste Ansatz ist die nahtlose Zusammenarbeit der verschiedenen Ansätze. Dies sollte solange möglich sein, insofern die Webservices, wie von beiden Seiten versprochen, die Lösung für die Interoperabilität über alle Plattformen und Netzwerke bleiben.

8 Literaturverzeichnis

Breymann, Loviscach /C-Klasse/

Ulrich Breymann, Jörn Loviscach : Die neue C-Klasse – C# im Vergleich mit C++ und Java

In: c't – Magazin für Computertechnik 04/2002, S. 98

Farley /.NET vs. SunONE/

Jim Farley : Microsoft .NET vs. J2EE – How do they stack up ?

http://java.oreilly.com/news/farley_0800.html, Abruf am 2002-04-25

Heise /.NET als OpenSource/

Heise Newsticker: Linux-Entwickler planen .NET als Open-Source

http://www.heise.de/newsticker/data/ju-08.07.01-002/, Abruf am 2002-08-30

Heise /.NET für Linux/

Heise Newsticker: Microsoft will .NET für Linux unterstützen

http://www.heise.de/newsticker/data/odi-17.07.01-000/ Abruf am 2002-08-30

Heise /Borland .NET/

Heise Newsticker: Borland schwenkt auf .NET ein

http://www.heise.de/newsticker/data/hos-23.05.02-000/, Abruf am 2002-05-23

Heise /Shared Source/

Heise Newsticker: Microsoft .NET als "Shared Source"

http://www.heise.de/newsticker/data/hos-28.06.01-000/ Abruf am 2002-08-30

Hoffman, Gabriel, Gosnell, Hasan, Holm, Musters, Narkiewickz, Schenken, Thangarathinam, Wylie, Ortiz /.NET Framework/

Kevin Hoffmann, Jeff Gabriel, Denise Gosnell, Jeff Hasan, Christian Holm, Ed Musters, Jan Narkiewickz, John Schenken, Thiru Thangarathinam, Scott Wylie, Jonothon Ortiz : Professional :NET Framework

Birmingham 2001

Knuth /Webservices/

Michael Knuth : Webservices – Einführung und Übersicht

Frankfurt 2002

Lelic, Schwichtenberg /Visual Studio .NET/

Senaj Lelic, Holger Schwichtenberg -

Visual Studio .NET Enterprise Architect – Unternehmensweit

In: iX - Magazin für professionelle Informationstechnik 6/2002

Loviscach, Schulz, Violka /Sunspiration/

Jörg Loviscach, Hajo Schulz, Karsten Violka : Sunspiration - .NET und SunONE im Plattformvergleich

In: C't – Magazin für Computertechnik 04/2002, S.92

Lüders /.NET embedded/

Daniel Lüders: ... und jetzt alle zusammen. Microsoft stülpt PDAs, Smartphones und Desktops sein .NET über

In: C't – Magazin für Computertechnik 10/2002, S.19

Microsoft /.NET Resources/

Microsoft, List of Third-Party .NET Resources.

http://msdn.microsoft.com/net/thirdparty/default.asp#lang, Abruf am 25-07-2002.

Schmidt /Sun ONE/

Donatus Schmidt : Von Applikationen zu Services SunONE

www.sun.de, Abruf am 2002-31-05

Sessions /J2EE vs. NET/

Roger Sessions : Java 2 Enterprise Edition (J2EE) versus the .NET Plattform – Two visions for e-Business

http://www.objectwatch.com, Abruf am 2002-06-01

Siering /Microsoft Internet/

Peter Siering: Das Microsoft Internet - .NET und was dranhängt

In: C't – Magazin für Computertechnik 4/2002, S.86

Sun /Sun ONE/

Sun Microsystems : Sun Open Net Environment (Sun ONE)

www.sun.de/Produkte/Software/SunONE/Material/pdf/SunONE_Deutsch_extern_2.2.pdf, Abruf am 2002-04-25

Technet /.NET für IT Experten/

Microsoft Technet: Microsoft .NET für IT-Experten

http://www.microsoft.com/germany/ms/technetdatenbank/overview.asp?siteid=511736, Abruf am 2002-04-17

Turau, Saleck, Schmidt /J2EE/

Volker Turau, Krister Saleck, Marc Schmidt : Java Server Pages und J2EE – Unternehmensweite Web-basierte Anwendungen

Wiesbaden, 2001

9 Anhang A

Glossar

.asmx	Dateierweiterung für Webservice Quellcode
.aspx	Dateierweiterung für ASP.NET Quellcode
.disco	Dicovery of Webservices. Ein Webservice hat eine oder mehr .disco Dateien welche Informationen beinhalten wie auf die WDSL Beschreibung zuzugreifen ist.
.ear	Enterprise Archive
.NET	Microsoft Webservice Initiative
ADO.NET	ActiveX Data Objects.NET
ADS	Active Directory Services
API	Application Programming Interface
ASP.NET	Active Server Pages.NET
CIL	Common Intermediate Language
CLI	Common Language Infrastructure
CLR	Common Language Runtime
COM	Component Object Model
CORBA	Common Object Request Broker
DCE	
DCOM	Distributed COM
DNA	Distributed Network Architecture
EJB	Enterprise Java Beans
HIS	Host Integration Server
IDL	Interface Definition Language
IIOP	Internet Inter ORB
IIS	Internet Information Server
J2EE	Java 2 Platform, Enterprise Edition
J2ME	Java 2 Platform, Micro Edition
J2SE	Java 2 Platform, Standard Edition
JAR	Java Archive
JCA	J2EE Connector Architecture
JDBC	Java Database Connection
JIT	Just in Time
JMS	Java Message Service
JNDI	Java Naming and Directory Service
JSP	Java Server Pages
JTA	Java Transaction Service
JVM	Java Virtual Machine
MSIL	Microsoft Intermediate Language
NLBS	Network Load Balancing Server
RI	Referenzimplementierung
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SunONE	Sun Open Network Environment
UDDI	Universal Description, Discovery and Integration
VB.NET	Visual Basic.NET
VS.NET	Visual Studio.NET
WML	Wireless Markup Language
WWW	World Wide Web
XML	eXtendend Markup Language