

7 Assemblies

- Anwendungen (.exe) bzw. Anwendungskomponenten (.dll) für .NET
 - Portable Execution (PE) Files
- Metadaten (Manifest) zur Selbstbeschreibung
 - Name, Version, Authentisierungsinformation
 - enthaltene Dateien, Typinformationen
 - Liste referenzierter Assemblies
- statische Assemblies
- dynamische Assemblies
 - ◆ zur Laufzeit von anderen .NET-Anwendungen erzeugte Assemblies
 - z. B. zur Template-basierten Codegenerierung in Entwicklungsumgebungen

8 Virtual Execution System — VES

- Teil der CLR
- Class Loader
- Verifier
- Domain-Konzept
 - ▶ Kapselung der Ausführung von Assemblies
- Thread-Unterstützung
- Sicherheits-Management
- Garbage Collection
- Metaprogrammierung / Reflexion

9 Unmanaged / Unsafe Code

- Aufruf von Komponenten / Bibliotheksfunktionen / COM-Objekten, die nicht in MSIL vorliegen
 - ◆ Werden nicht von der CLR verwaltet: *Unmanaged Code*
 - ◆ Freigabe der Ressourcen bleibt dem Aufrufer überlassen (kein Garbage Collector)
- Verwendung von Zeigeroperationen: *Unsafe Code*
 - ◆ Direkter Zugriff auf den Speicher, keine Garbage Collection
 - ◆ Code muss als "unsafe" gekennzeichnet werden

10 Application Domains

- eine .NET-CLR wird in einem virtuellen Adressraum ausgeführt
 - keine Hardware-Kapselung innerhalb der CLR
- Software-Kapselung der Ausführung von Assemblies
 - kein direkter Zugriff auf Assemblies anderer Domains möglich
 - CLR verhindert Übergabe von Objekt-Referenzen zwischen Domains
 - Interaktion nur über speziell gesicherten Inter-Domain-Kommunikationsmechanismus möglich
- eigene Domains werden erzeugt wenn
 - Code mit anderen Sicherheits-Einstellungen/Zugriffsrechten geladen wird
 - Isolation zwischen Code-Teilen explizit gewünscht wird
 - Code unabhängig voneinander terminieren können soll
- Terminierung von Domains
 - kontrollierte Freigabe aller belegten Ressourcen

10 Application Domains (2)

- CLR startet mit einer *Default Application Domain*
- Default Application Domain lädt *Hosting Code*
 - baut Umgebung für eine Anwendung auf
 - erzeugt weitere Application Domains
- Weitere Application Domains führen *User Code* aus
 - *User Code* enthält die eigentliche Anwendungslogik
 - Ist in Form von Assemblies geladen
 - In der Regel mehrere Assemblies pro Application Domain
- Code Access Security
 - Legt Rechte basierend auf Code-Charakteristiken (Evidence) fest
 - Evidence = Quelle des Codes (lokal, remote) oder Signaturen
- Role-based Security
 - Rechte basierend auf aktuellem Benutzer (z. B. login-Information)

11 Programmierung verteilter Systeme

- Verteilungskonfiguration
 - ◆ Fernaufruf: Fernverweis auf Objekt wird übergeben (call by reference)
 - ◆ Serialisierung: Objektkopie wird übergeben (call by value)
- Objekterzeugung
 - ◆ Bekanntmachung eines öffentlichen Objekts (lokale Erzeugung)
 - ◆ Fernerzeugung eines öffentlichen Objekts (Erzeugung wird lokal vorbereitet, aber erst durchgeführt, wenn ein Client darauf zugreift)
 - ◆ Fernerzeugung eines privaten Objekts
- Proxy und Skeleton werden implizit zur Laufzeit aus Metadaten generiert
- XML-Konfigurationsdatei legt Verteilung fest

11 Programmierung verteilter Systeme

- Unterstützung verschiedener Transportprotokolle
 - ◆ z.B. TCP oder HTTP, Binärformat oder XML
- Eingriffsmöglichkeiten in das Nachrichtenformat (*Formatters*) und den Transport von Nachrichten (*Sinks, Channels*)
 - ◆ Verschlüsselung, Kompression, Mitführen von Zusatzinformationen
- "Leasing Distributed Garbage Collector"
 - ◆ Objektreferenz ist nur für eine bestimmte Zeit gültig
 - ◆ Lease wird durch Fernaufruf verlängert
 - ◆ kein Referenzzähler oder periodisches Pingen notwendig (DCOM)

H.6 OSGi — Open Services Gateway Initiative

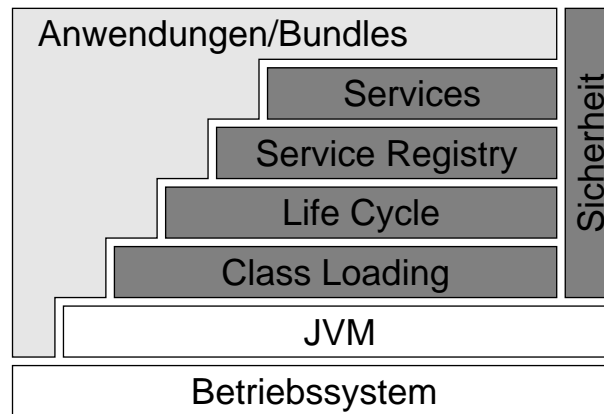
1 Überblick

- OSGi Spezifikationen:
standardisierte, komponenten-orientierte Ausführungsumgebung für vernetzte Dienste
- Basis: Java
- Ziel-Anwendungsbereiche
 - Mobiltelefone
 - Automobil (Infotainment-Bereich)
 - Telematik
 - Eingebettete Systeme
 - Haustechnik
 - PC- und Mainframe-Anwendungen
- Literatur: <http://www.osgi.org/>

2 Einsatzbeispiele

- NG Smart Phones
 - einheitliche Plattform für Mobiltelefon-Anwendungen
 - Anwendungen zentral administrierbar
- Shell HomeGenie
 - Softwareplattform zur Integration von Kameras, Sensoren, Thermostaten, Stromschaltern, etc. im Hausautomatisierungsbereich
- Eclipse
 - Java-Entwicklungsumgebung
 - ab Version 3.0: Installation und Update von Plug-ins zur Laufzeit
- BMW iDrive (5er und 7er, neuer 3er)
 - Basis: Betriebssystem VxWorks und JVM Jeode von Windriver
 - OSGi-Implementierung von Siemens VDO
 - Integration von Navigation, Mobiltelefon (Bluetooth), Internet-Anbindung und anderen Infotainment-Anwendungen

3 Architektur



- mehrere unabhängige Anwendungen in einer JVM
- Anwendung = Bundle
 - JAR-Datei mit zusätzlichen Header-Informationen (Manifest)
 - Schnittstellenkonventionen -> Komponentenmodell
 - Installation, Update, Start, Stop und Deinstallation über Netzwerk

4 Services

- Class Sharing
 - Bundles können Pakete exportieren und von anderen Bundles importieren
 - vermeidet Redundanz von Code
 - ermöglicht Bereitstellung von Diensten in einer OSGi-Umgebung
 - Regeln zum Umgang mit unterschiedlichen Versionen
- Life Cycle Management
 - Dienste zur Installation von Bundles
- Service Registry
 - Nameservice
 - registrierte Objekte werden *Services* genannt:
Schnittstelle und Menge von Eigenschaften (*Properties*)
 - automatische Abmeldung bei Bundle-Deinstallation + Information anderer Bundles