JAVA DAYS ´99



Java Entwurfsmuster
Einsatz in einer Connector Architektur

Thomas Bühner
IT Architekt
MERANT Solutions GmbH, München
thomas.buehner@merant.com

Sun microsystems

Java Patterns
Patterns - Übersicht



Patterns / Entwurfsmuster:

- Standardlösungen für Software Probleme
- „Best Practice“ Lösungen
- Verhindern das Rad neu zu erfinden

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Patterns - Kategorien



Design Patterns

- Architektur Patterns
- Design Patterns
- Programmiersprachen Idiome

Analyse Patterns - Analysemodelle

Organisations Patterns - Projektorganisation

Prozess Patterns - Entwicklungsprozess

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Patterns - Kategorien



Architektur Patterns

- Layer
- Broker

Struktur der Statischen Connector Runtime Komponente

Design Patterns

- Singleton
- Abstract Factory
- Remote Proxy
- Command
- Publish-Subscribe

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Middleware - Business Überlegungen



- Time-To-Market
- Geschäftsideen schnell umsetzen
- Wenige Anwendungsteile neu schreiben
- Wiederverwendung existierender Komponenten
- Applikationslandschaft = Service-Komponenten

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Middleware Vielfalt



Heterogene Middleware Technologien im Unternehmenseinsatz

- Sockets - TCP/IP
- CORBA - Gremien-Standard
- COM - De-Facto-Standard
- MQSeries - Message-Oriented Middleware

© 1999 Sun Microsystems GmbH

Java Patterns
Möglichkeiten zur Optimierung



- Wenig Middleware Ausbildungszeit für Anwendungsentwickler
- Konzentration der Anwendungsentwickler auf Business Logik
- Automatisierte Kontrolle der Programmschnittstellen
- Unabhängigkeit von Programmiersprachen
- Anbindung von externen Systemen (SAP)
- Zukünftige Lösung für Java: Connector Architecture

© 1999 Sun Microsystems GmbH

Java Patterns
Broker - eine Lösung



- Unterstützung von synchronen und asynchronen Middleware-Protokollen
- Sockets, CORBA und MQSeries können miteinander kommunizieren
- Unterstützung von objektorientierten und prozeduralen Programmiersprachen
- Integration mit aktuellen Technologien (Enterprise Java Beans)
- Entlastung der Anwendungsentwickler

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Broker - Systemkomponenten



- Definition der Komponentenschnittstellen
- Generierung von Adapter-Code
- Business Applikation sieht nur Adapter-Code (Blackbox Prinzip)
- Runtime Komponenten übernehmen Middleware Transport
- Automatisierte Behandlung von Laufzeitfehlern
- Automatisierte Kontrolle der Programmschnittstellen

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Broker - Typischer Ablauf



- Business Applikation - Client
- Generierter Adapter-Code
- Broker Runtime Komponente - Client
- *Middleware Protokoll (z.B. Sockets)*
- Zentraler Broker - Middleware-Vermittler
- *Middleware Protokoll (z.B. MQSeries)*
- Broker Runtime Komponente - Server
- Generierter Adapter-Code
- Business Applikation - Service

© 1999 Sun Microsystems GmbH

Java Patterns
Architektur Pattern - Layer



Einsatzort Gesamtstruktur Client

Einsatzziel Klare Aufteilung der Aufgaben

Vorgehensweise Abstraktionsebenen definieren - von „Daten anfordern“ bis zum Middleware API Call

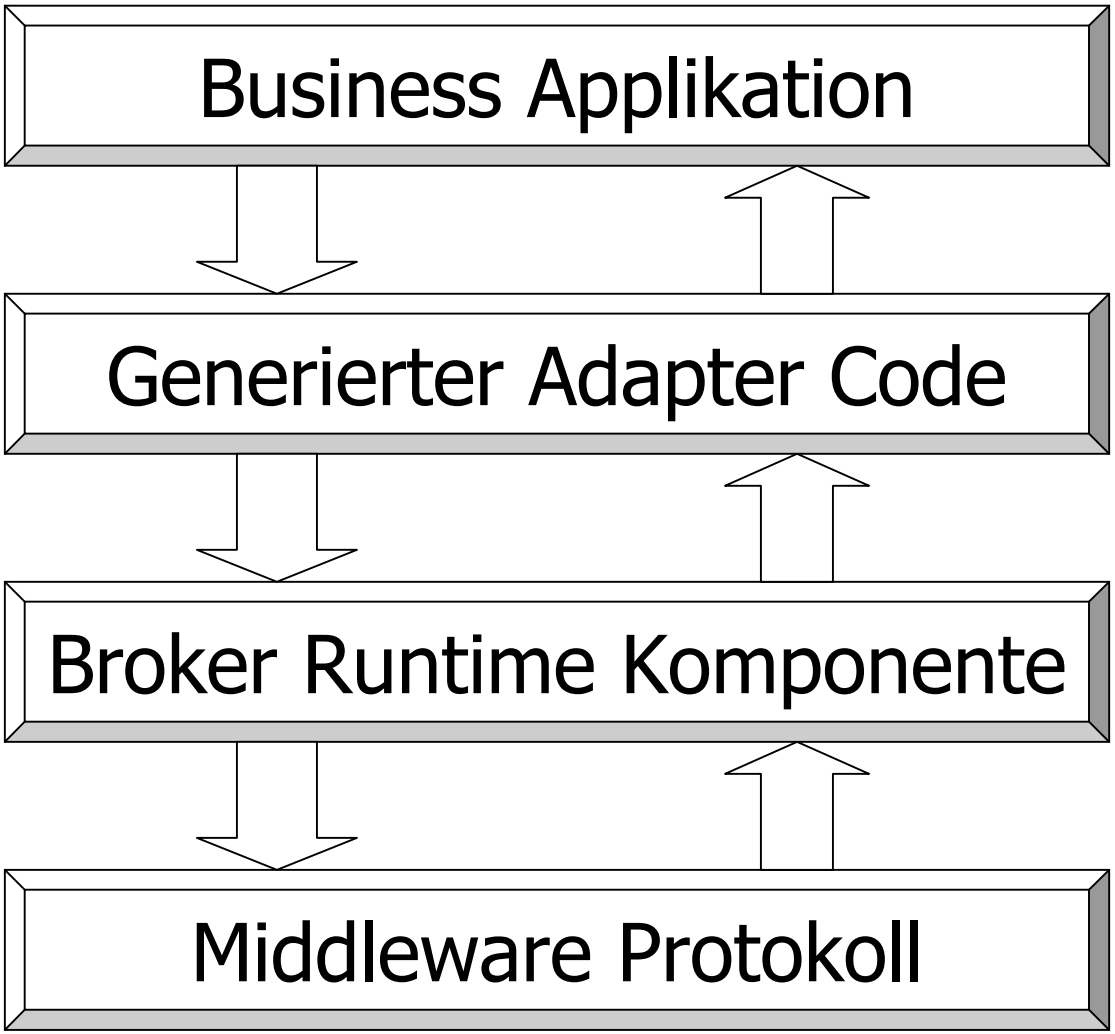
Benefit

- Einfache Steuerung auf abstrakter Ebene
- Middleware Spezifika werden nur auf unteren Ebenen behandelt

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Architektur Pattern - Layer



© 1999 Sun Microsystems GmbH

Java Patterns
Architektur Pattern - Broker



Einsatzort Zentraler Broker

Einsatzziel Vermitteln zwischen verschiedenen Middleware Protokollen

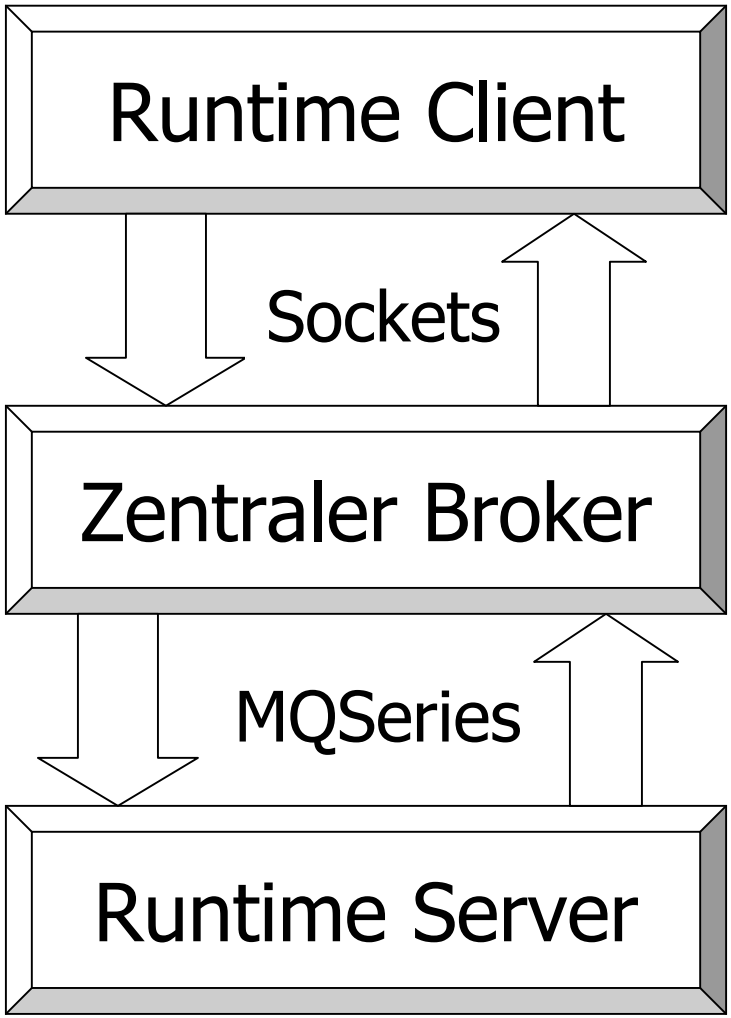
Vorgehensweise Empfang von Protokoll A und Senden von Protokoll B, je nach Vorhandensein der Protokolle

Benefit

- Zusammenarbeit von Komponenten mit verschiedenen Middleware Protokollen

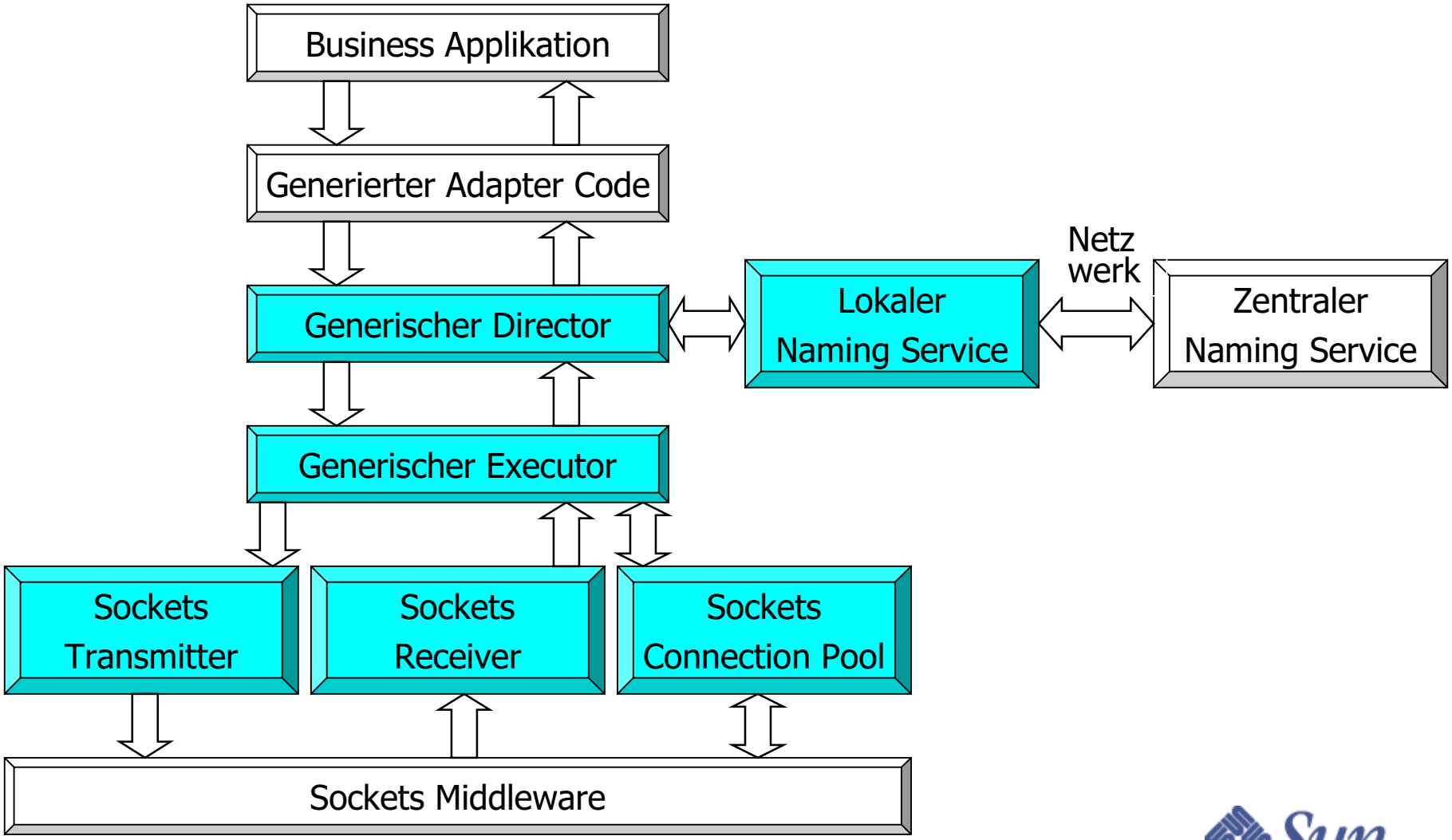
© 1999 Sun Microsystems GmbH

Java Patterns
Architektur Pattern - Broker



© 1999 Sun Microsystems GmbH

Java Patterns
Runtime Komponente - Struktur



© 1999 Sun Microsystems GmbH

Java Patterns
Runtime Komponente - Struktur



(Code Beispiel - Director)
(Code Beispiel - Disconnector)

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Design Pattern - Singleton



Einsatzort Lokaler Naming Service

Einsatzziel Sicherstellen, daß alle Komponenten auf dasselbe Objekt zugreifen

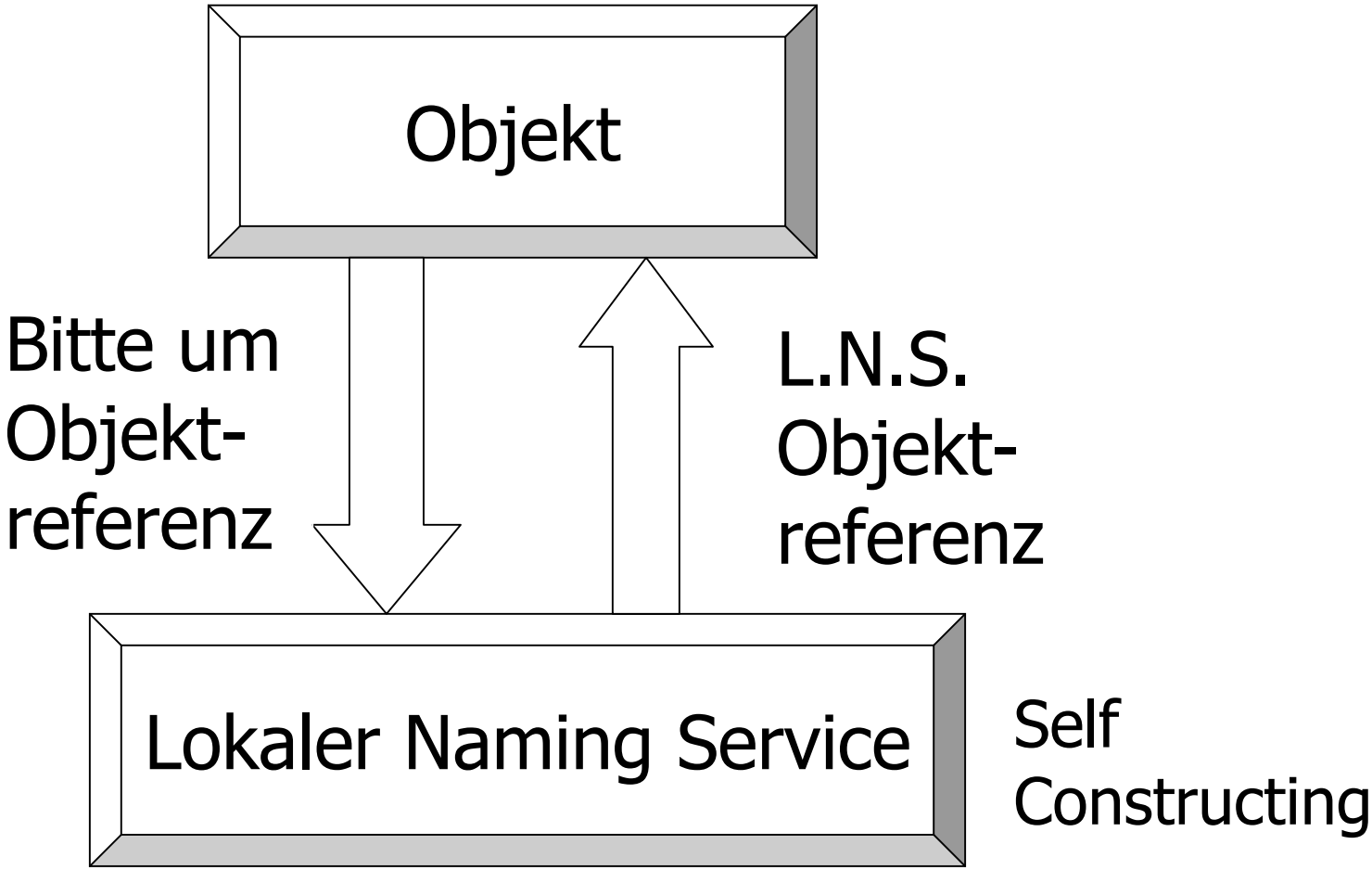
Vorgehensweise Nur der Lokale Naming Service kann eine Instanz von sich selbst anlegen

Benefit

- Datenkonsistenz sichergestellt
- Optimale Resourcennutzung

© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Singleton



© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Singleton



(Code Beispiel - Singleton)

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Design Pattern - Abstract Factory



Einsatzort Transmitter, Receiver

Einsatzziel Generisches Steuerprogramm

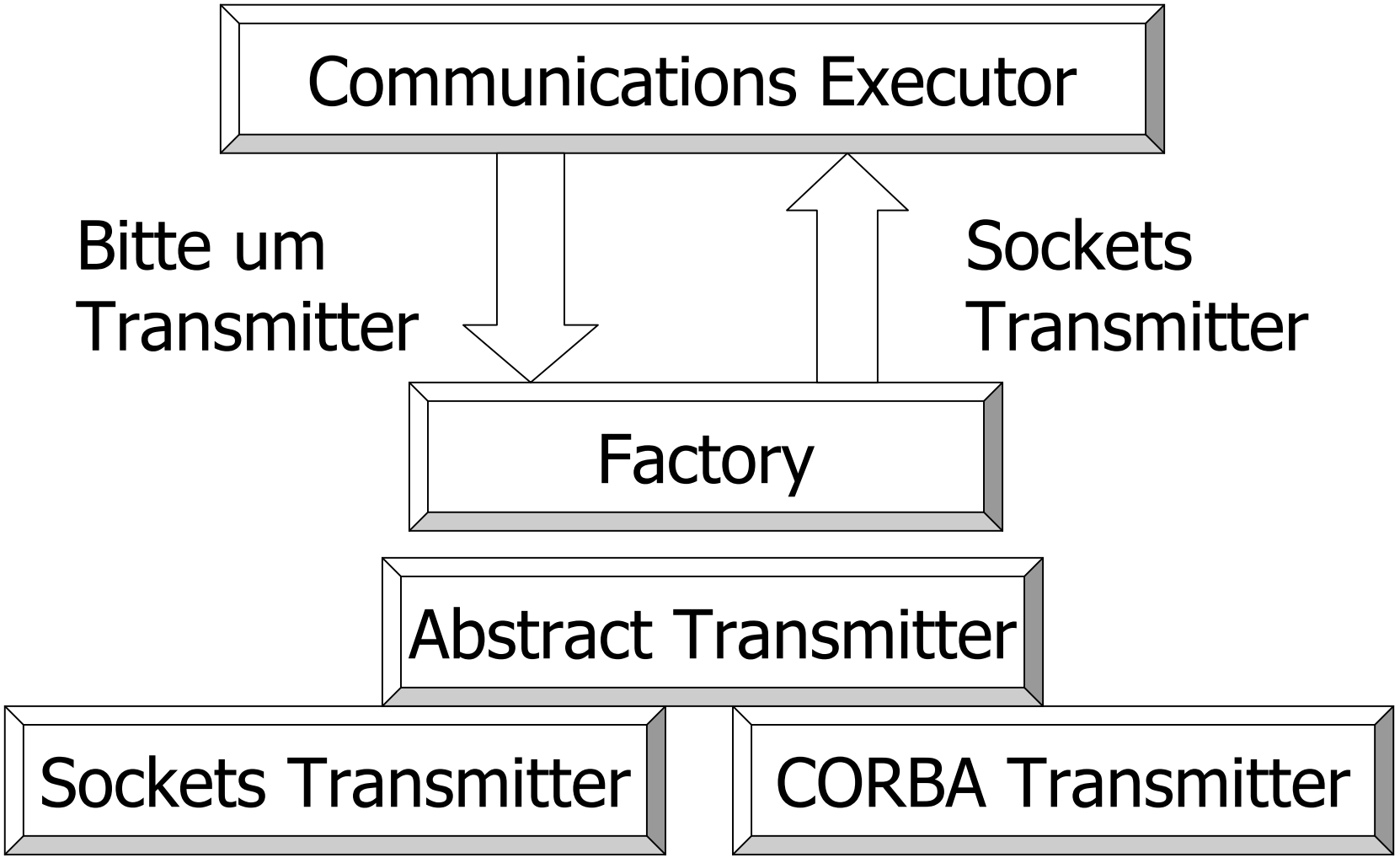
Vorgehensweise Abstrahierung der gemeinsamen Eigenschaften, z.B. Senden, Empfangen, Timeout

Benefit

- Viele Programmteile sind unabhängig von spezifischer Middleware
- Flexibler Code, schnell zu erweitern

© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Abstract Factory



© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Remote Proxy



Einsatzort Lokaler Naming Service

Einsatzziel Puffern von Remote Daten

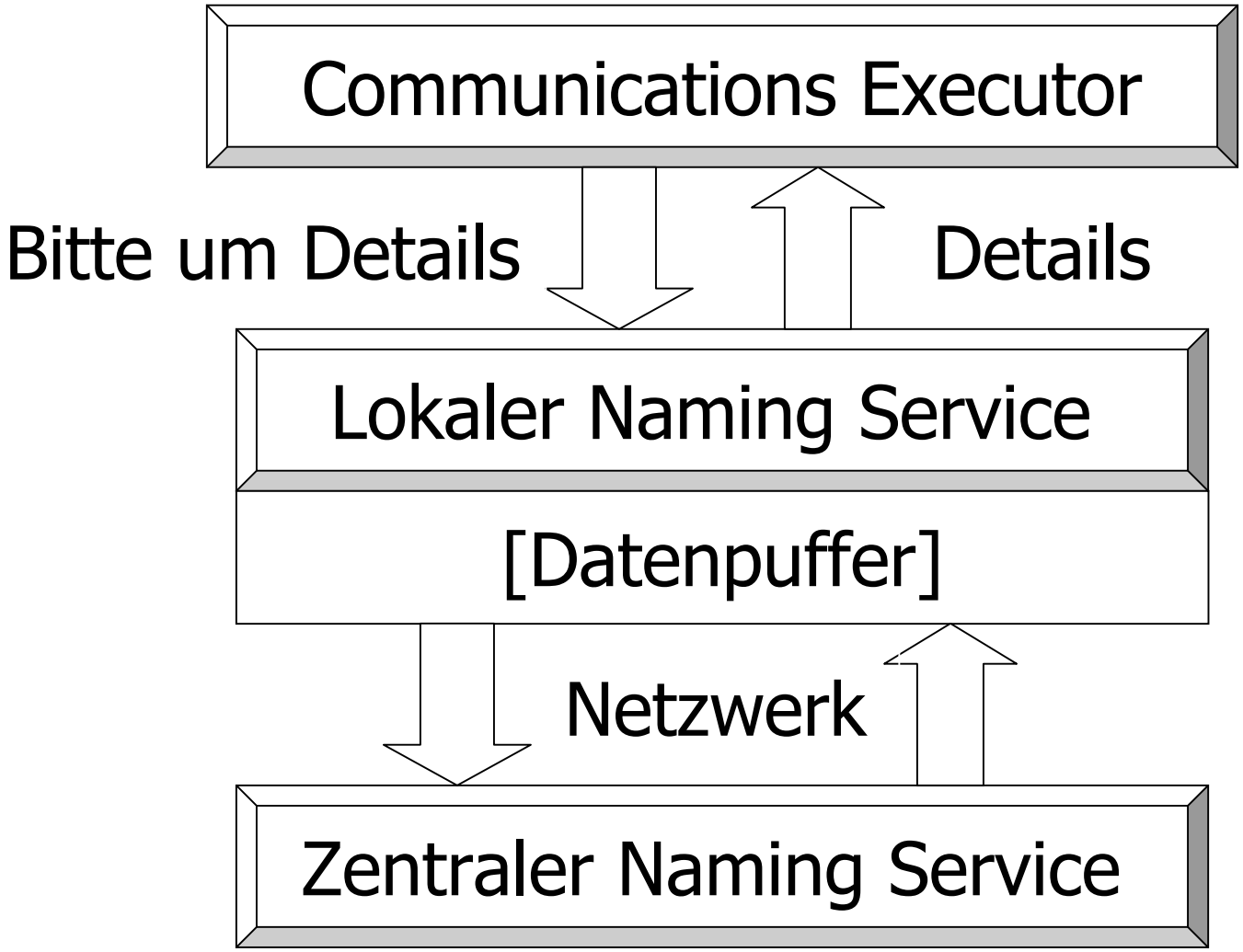
Vorgehensweise Lokale Weiterverwendung von bereits bekannten Daten

Benefit

- Erhöhte Performance
- Verringerung der Netzwerk Belastung

© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Remote Proxy



© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Command



Einsatzort Alle Runtime Komponenten

Einsatzziel Befehl an andere Komponente

Vorgehensweise Command Objekt kann als „normale“ Daten verschickt werden

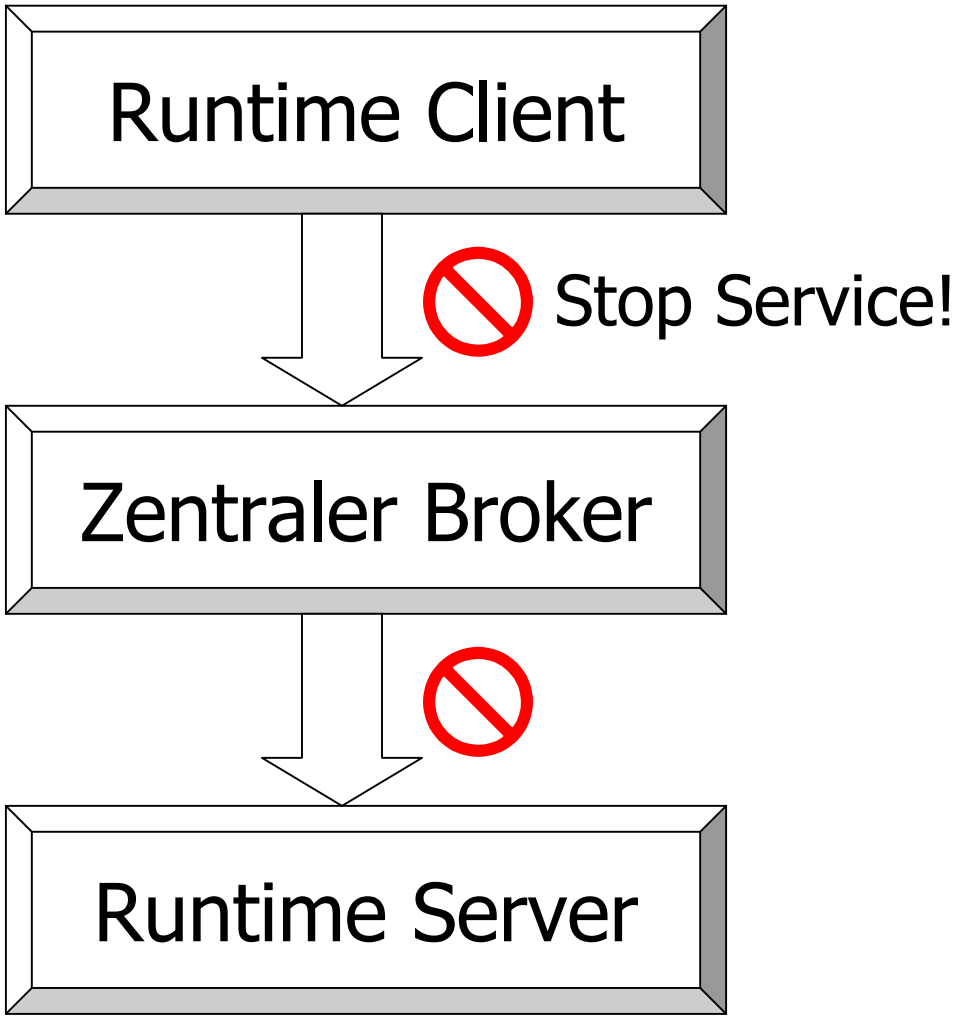
Benefit

- Fernsteuerung von Prozessen
- Minimierung des Resourcenverbrauchs

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Design Pattern - Command



© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Publish-Subscribe



Einsatzort Zentraler Naming Service

Einsatzziel Informieren der Lokalen Naming Services über Änderungen

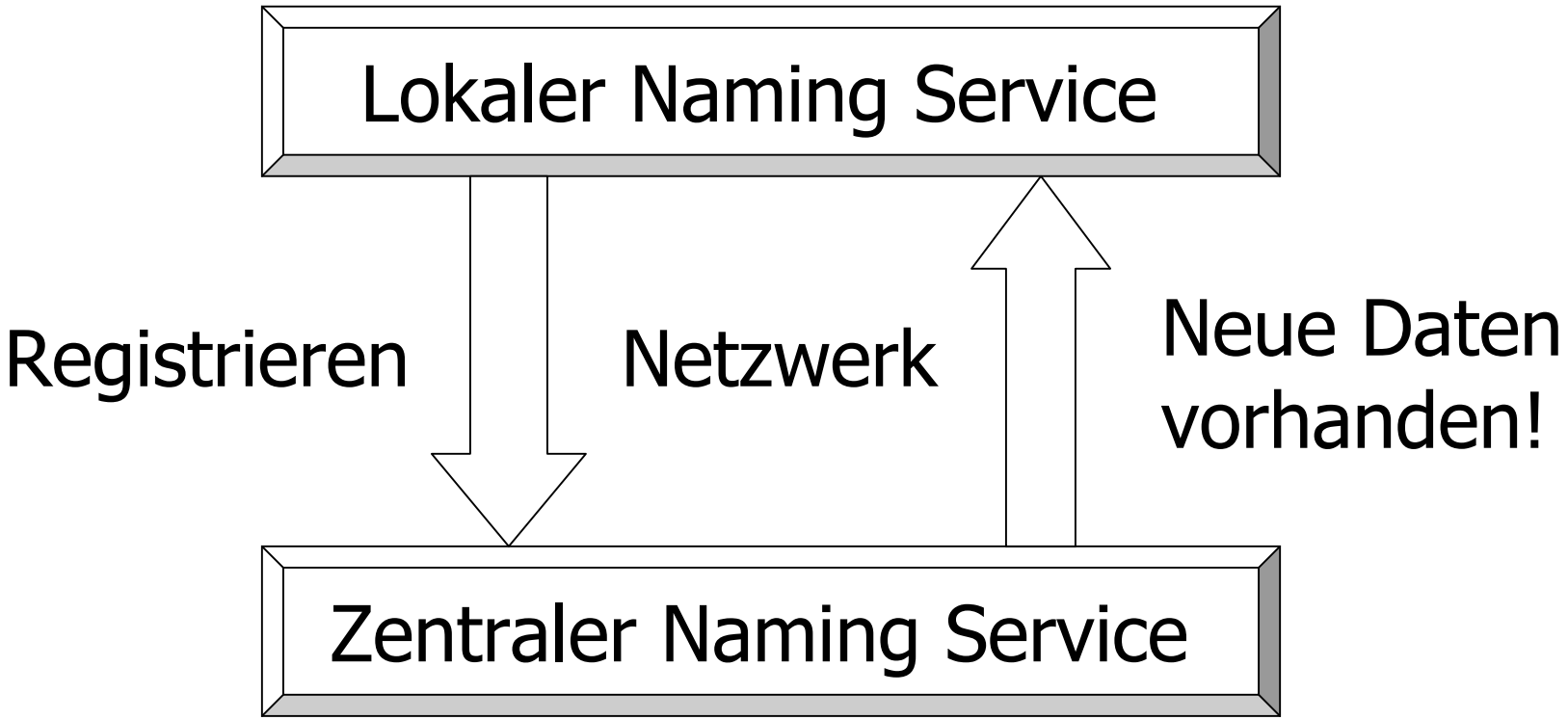
Vorgehensweise Lokale Naming Services registrieren sich beim Zentralen, Zentraler informiert alle Lokalen bei Änderungen

Benefit

- Korrekte Verwendung von Daten trotz Pufferung

© 1999 Sun Microsystems GmbH

Java Patterns
Design Pattern - Publish-Subscribe



© 1999 Sun Microsystems GmbH

Java Patterns
Broker - Benefits



- Schnellerer Time-To-Market für Applikationen
- Zeitersparnis bei Ausbildung der Anwendungsentwickler
- Konzentration des Middleware Wissens auf eine kleine Spezialgruppe
- Build or Buy? Aufwand etwa gleich
- Wartung liegt im Unternehmen

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Patterns - Benefits



- Zeitersparnis bei Design und Programmierung
- Einfachere Kommunikation mit Fachkollegen durch gemeinsame Sprache
- Ergebnis: Flexible Architekturen
- Änderungen und Erweiterungen einfacher
- Schnellerer Time-To-Market

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Patterns - Schlußfolgerung



Einsatz von Patterns ist essentiell für den Erfolg von Komponenten-Anwendungen

Investitionen in Design zahlen sich aus

Investitionen in Patterns Design Ausbildung zahlen sich aus

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Zukunft Heterogener Systeme



Java Connector Architecture

- Verbindung mit heterogenen Enterprise Systemen
- Standardisierte, austauschbare Schnittstellen
- Momentan in der ersten Standardisierungsphase

© 1999 Sun Microsystems GmbH

Sun microsystems

Java Patterns
Literatur



- Frank Buschmann et.al.: "A System of Patterns", Wiley, 1996
- Erich Gamma et.al.: "Design Patterns", Addison-Wesley, 1995
- John M. Vlissides (ed.): "Pattern Languages of Program Design", 1995
- Patterns Home Page http://hillside.net/patterns/
- Peter Coad et.al.: „Java Design, 2nd Ed.“, Yourdon 1999
- J2EE Connector Architecture: http://java.sun.com/aboutJava/communityprocess/jsr/jsr_016_connect.html

© 1999 Sun Microsystems GmbH