

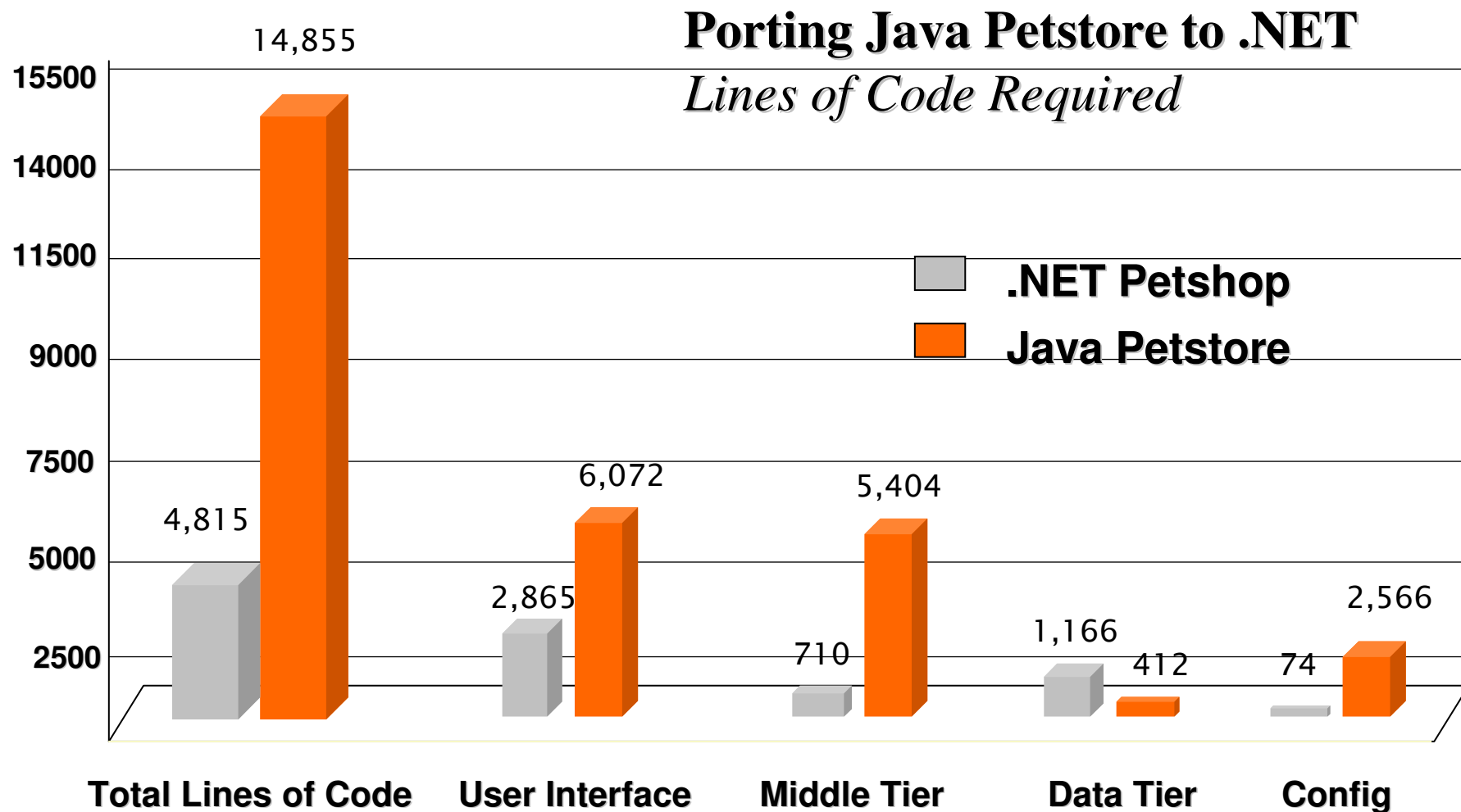


.net in der Praxis

Von der Idee bis zum Einsatz

Harald Haller und Alexander Ramisch
OOP 20. Januar 2003

Microsoft verspricht einiges mit .NET ...



und noch mehr ...

- .NET Petshop
 - 68% weniger Code
 - 28 mal schneller
als J2EE Petstore
 - Web Services lassen sich mit
 - 60% weniger Code und
 - 52% kürzerem Entwicklungsaufwand
zum Petshop hinzufügen als bei IBM Websphere 4.0
- ➔ *Die zukunftsichere Technologie*

... ist kinderleicht!



Agenda

■ Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

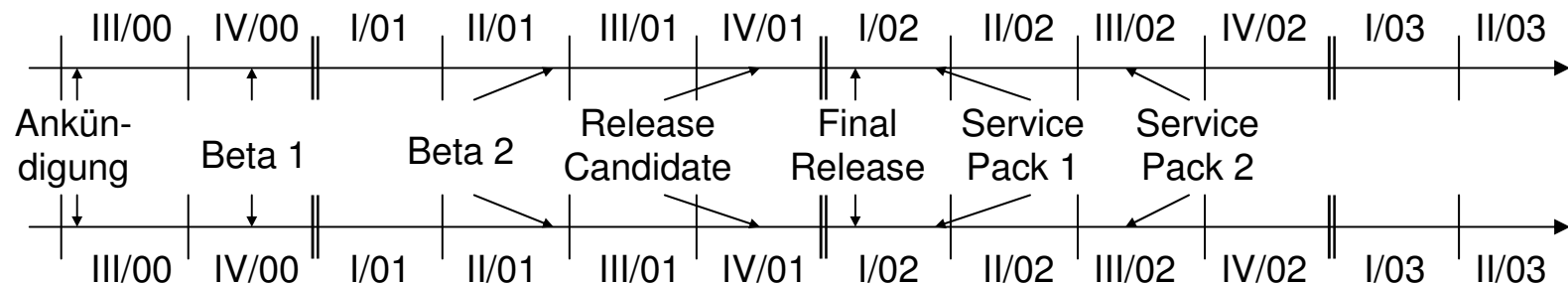
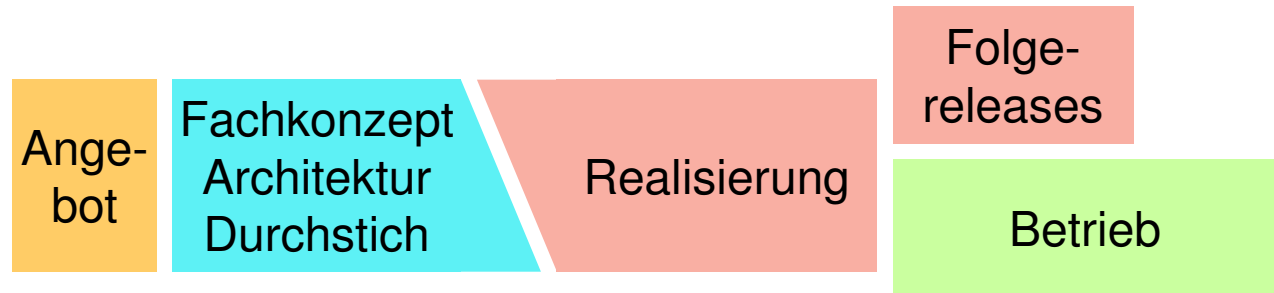
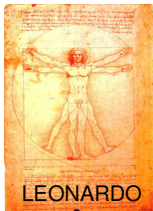
Entwicklungsumgebung

Betrieb

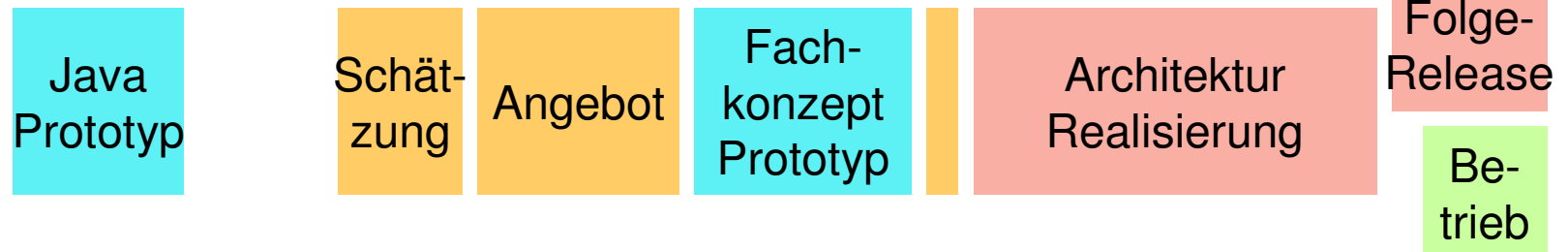
Aufwand

Zusammenfassung

Bereits auf der Basis von Betaversionen haben sich unsere Kunden für .NET entschieden



Münchener Rück
Munich Re



Die Real I.S. setzt auf .NET für ein System, das ihre Kernprozesse komplett abbildet

- Kunde REAL I.S. AG, ca. 100 Mitarbeiter,
100%-Tochter der Bayerischen Landesbank,
Fokus auf geschlossenen Immobilienfonds,
„Alles rund um die gewerbliche Immobilie“
- Projekt Kernprozesse des Kunden maßgeschneidert integriert
Abwicklung aller Geschäftsvorfälle während der
Fondlaufzeit
- Zeitraum Mai 2001 bis Februar 2003
- Team bis zu 14 Mitarbeitern
- Mengen-
gerüst ca. 100 Nutzer
ca. 140 Dialoge
ca. 160 DB-Tabellen mit ca. 2000 Attributen
ca. 1.100 Programmklassen (C#)

Mit dem .NET-Projekt MIRA nehmen wir die Vorreiterrolle bei der Münchener Rück ein

- Kunde Münchener Rück:
Weltweit führende Rückversicherung
5.000 Kunden (Erstversicherer) in 150 Ländern
- MIRA Munich Re Internet Risk Assessor
Risikobewertungssystem für personenbezogene Versicherungen
Weltweit in verschiedenen Markt- und Sprachversionen
- Projekt **Neues internationales Pflegesystem für MIRA** mit
Dokumentenmanagement
Workflow-Unterstützung
MS-Office-Integration
- Zeitraum September 2001 bis März 2003 (Stufe 1)
- Team bis zu 10 Mitarbeiter im .Net-Teilprojekt
- Mengen-
gerüst ca. 100 Nutzer
generische und konfigurierbare Dialoge
ca. 90 DB-Tabellen mit ca. 600 Attributen
ca. 1.200 Programmklassen (C#)

MIRA Maintenance - [International - Occupations - Edit Submenu]
Subject Edit Window Help

In work
Market Versions
International
General items
Underwriting
What's New
Medical
Residential
Occupations
Items
Submenus
Additional docs
Allowed Rates
Shortcut texts
Pastimes
Financial
Calculators
References
About MIRA
User Guide
Claims
UK

Go to
Menu or Item:

Occupations Menu
Introduction
Industry
Driving and Transportation
Health and Welfare
Health and Welfare General Background
Dentists
Dentists General Background
Dentist
Orthodontist
Doctors
Nurses
Other Industries

Type
☒ Submenu
☐ Top item
☐ Standard item

Properties
Area:
Main ID:
Display:

Synonyms

Synonym (	Synonym (lang)
Dental Pro	Dental Professions
Dental Sur	Dental Surgeons

Last updated: 12/23/01 09:11 by Wowra
Last published: 12/09/01 16:28 by Mückenhausen

Operation successfully terminated.

Agenda

Projektvorstellung

■ Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

Betrieb

Aufwand

Zusammenfassung

J2EE und .NET haben viele Entsprechungen

Es gibt Unterschiede im Detail

Zweck	.NET	J2EE
Ausführungsumgebung	CLR (MSIL)	Java VM (Bytecode)
Basisbibliotheken	.NET Framework Class Library	Java Core API
Programmiersprache	C#, managed C++, VB.NET	Java
Web-Clients	ASP.NET	JSP
Native Clients	Windows Forms	Swing
Datenzugriff	ADO.NET	JDBC
Webservices	ASP.NET, .NET Services	SUN's JXTA

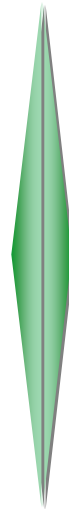
Trotz Gegenargumenten fiel in 2001 die Entscheidung für .NET

Risiken bei .NET

Neue Technologie

- Entwicklungsumgebung?
- Laufzeitumgebung?
- Performanz und Stabilität?
- Kein Know-how (Vorreiter)
- Keine fertigen Komponenten

Andere bewährte Technologien:
JAVA, Visual Basic



Gründe für .NET

- **Firmenstrategie: Microsoft**
- **Strategie von Microsoft: .NET**
- Anbindung von MS-Produkten
- Sicherheitskonzepte
- Softwareverteilung
- In Teilen innovativer als J2EE, Visual Basic

Entscheidung für .NET
(trotz Preis)

Agenda

Projektvorstellung

Entscheidung für .NET

■ **Vorgehen**

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

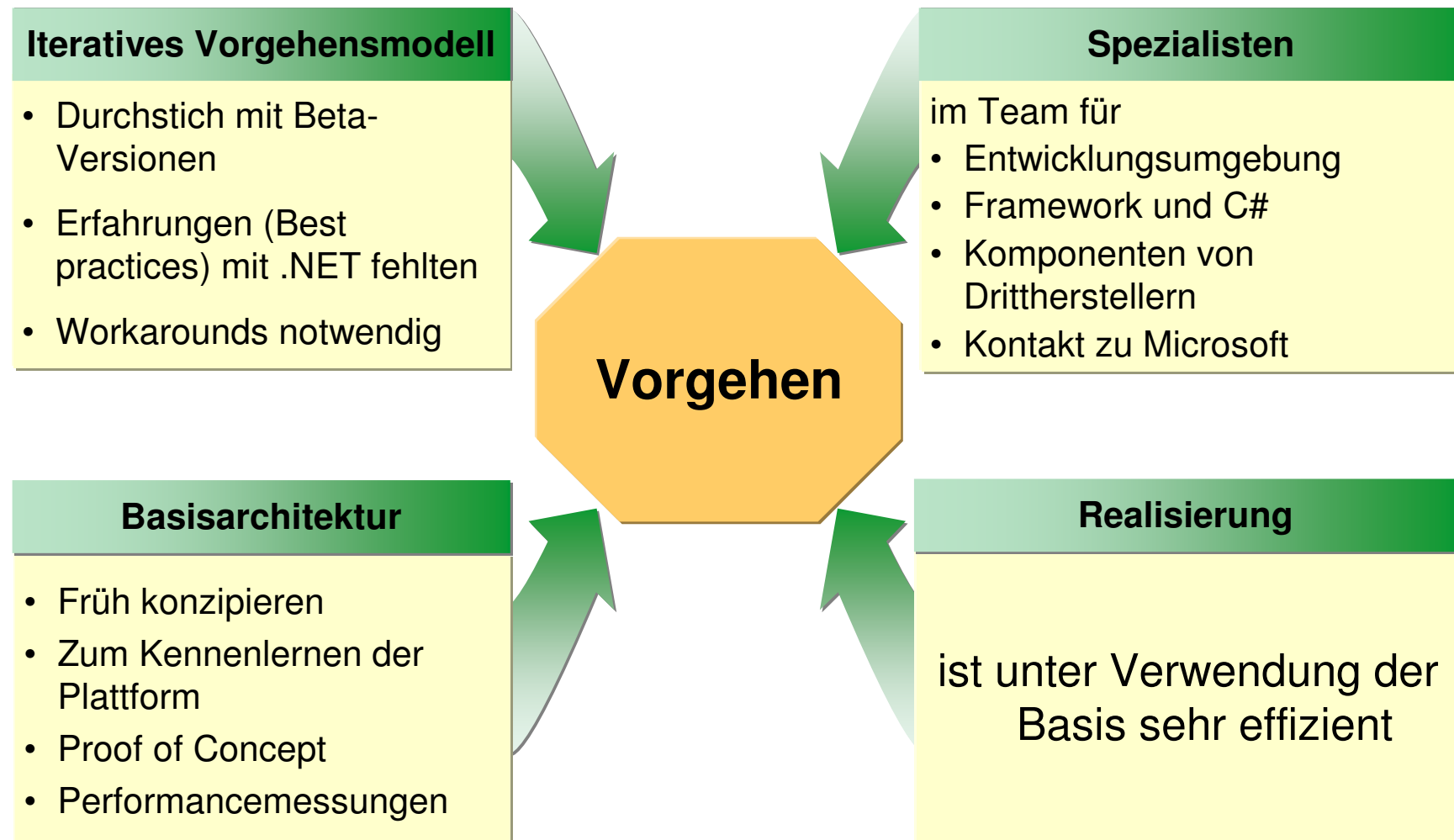
Entwicklungsumgebung

Betrieb

Aufwand

Zusammenfassung

Es wurde ein Vorgehen gewählt, das Risiken eindämmen soll



Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

■ **Architektur**

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

Betrieb

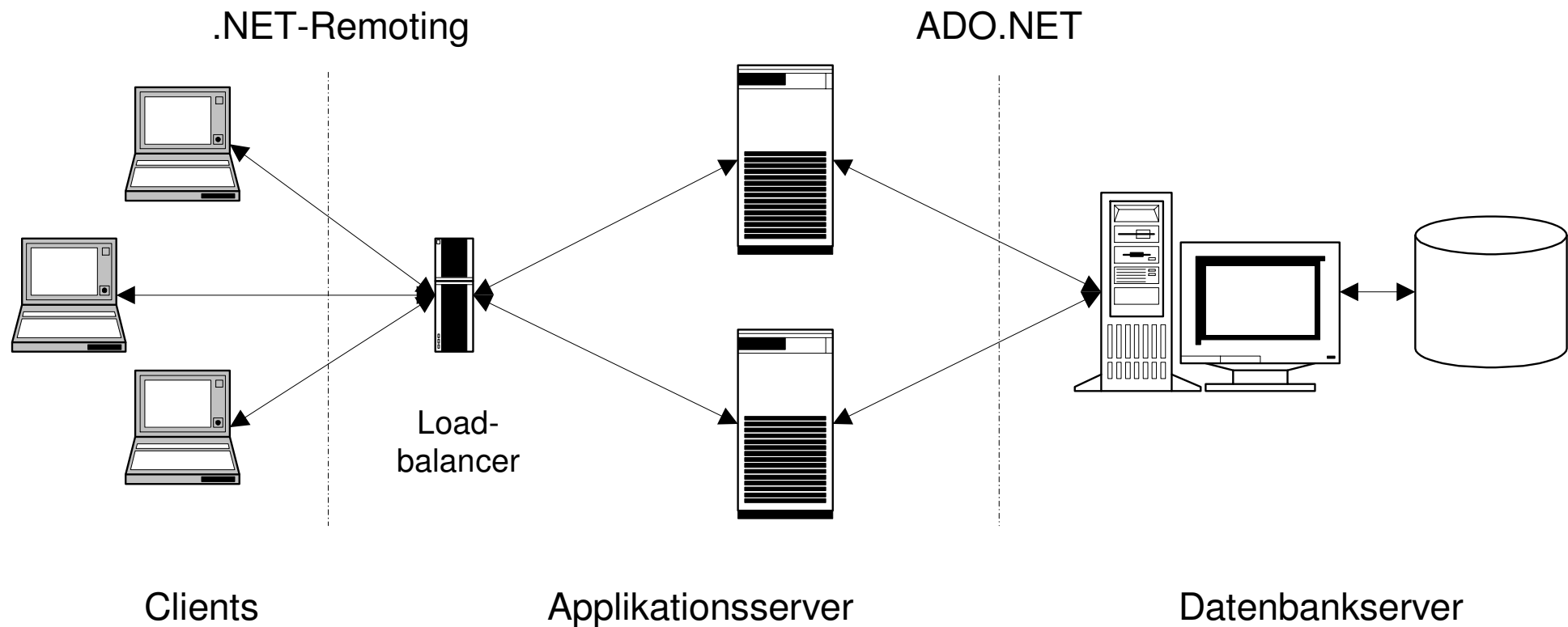
Aufwand

Zusammenfassung

Zusätzlich zum Framework wird einiges benötigt

- .NET bietet
 - .NET Framework Class Library mit Klassen für nahezu jeden Zweck
 - Windows Forms für native Clients
 - Web Forms für Web-Clients
 - Remoting zum Anbinden des Clients an den Server
 - ADO.NET für Datentransport und zur Anbindung an die Datenbank
- Eine Architektur ist nötig
 - Trennung von Zuständigkeiten (Schichten, Ebenen, Komponenten)
 - Einheitlichkeit
 - Treffen wichtiger Designentscheidungen

Verteilung auf mehrere Rechner garantiert Ausfallsicherheit und Skalierung



Beim Entwurf der Anwendungsarchitektur werden frühzeitig wesentliche Grundlagen gelegt

Fragestellung

Oberflächentyp

Wo Client-/Server-Schnitt ?

Protokoll der Middleware

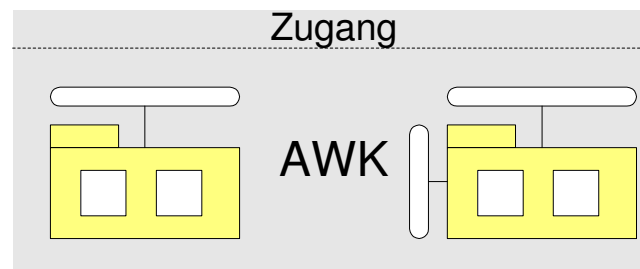
Komponentenmodell

Persistenzschnittstelle

Datenbankzugang

Präsentation

Dialogsteuerung



Datenzugriff



Lösung

Windows Forms

Native Clients

.NET Remoting

QUASAR

Assemblies

Zugriffsobjekte

ADO.NET

Unsere Sichtweise zu Komponenten

■ Was ist eine Komponente?

Ein Stück Software

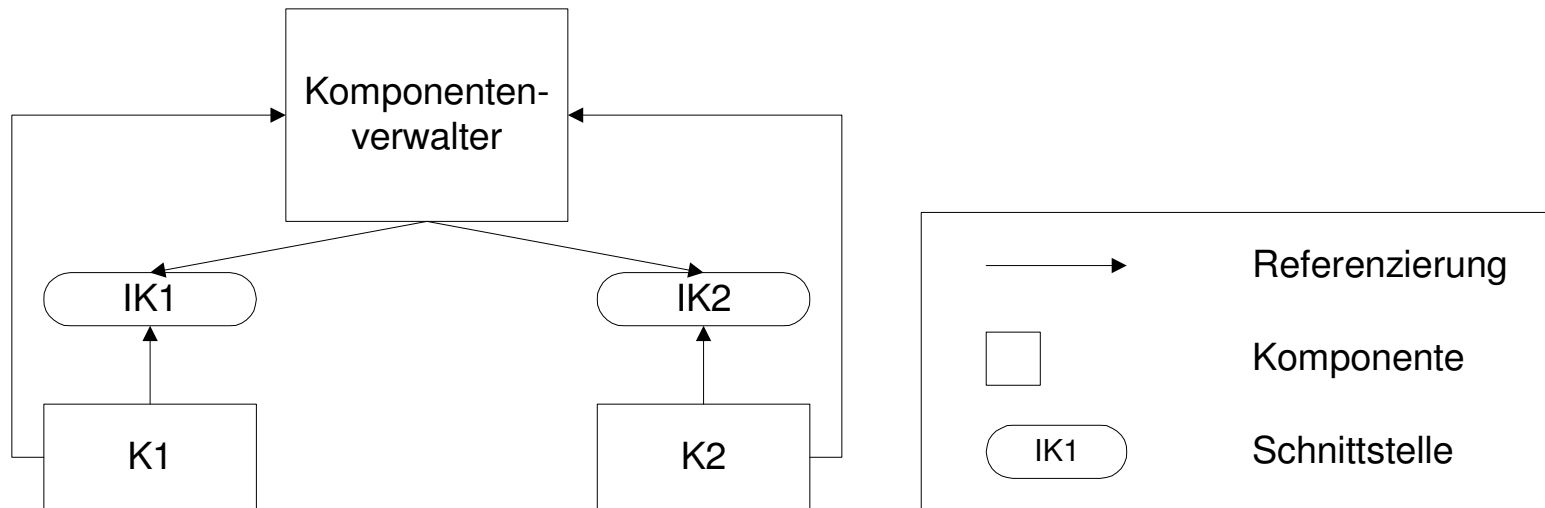
- klein genug, um es in einem Stück zu erzeugen und zu pflegen
- groß genug, um eine sinnvolle Funktionalität zu bieten und eine individuelle Unterstützung zu rechtfertigen
- ausgestattet mit Schnittstellen, um mit anderen Komponenten zusammenzuarbeiten

■ Warum setzen wir Komponenten ein?

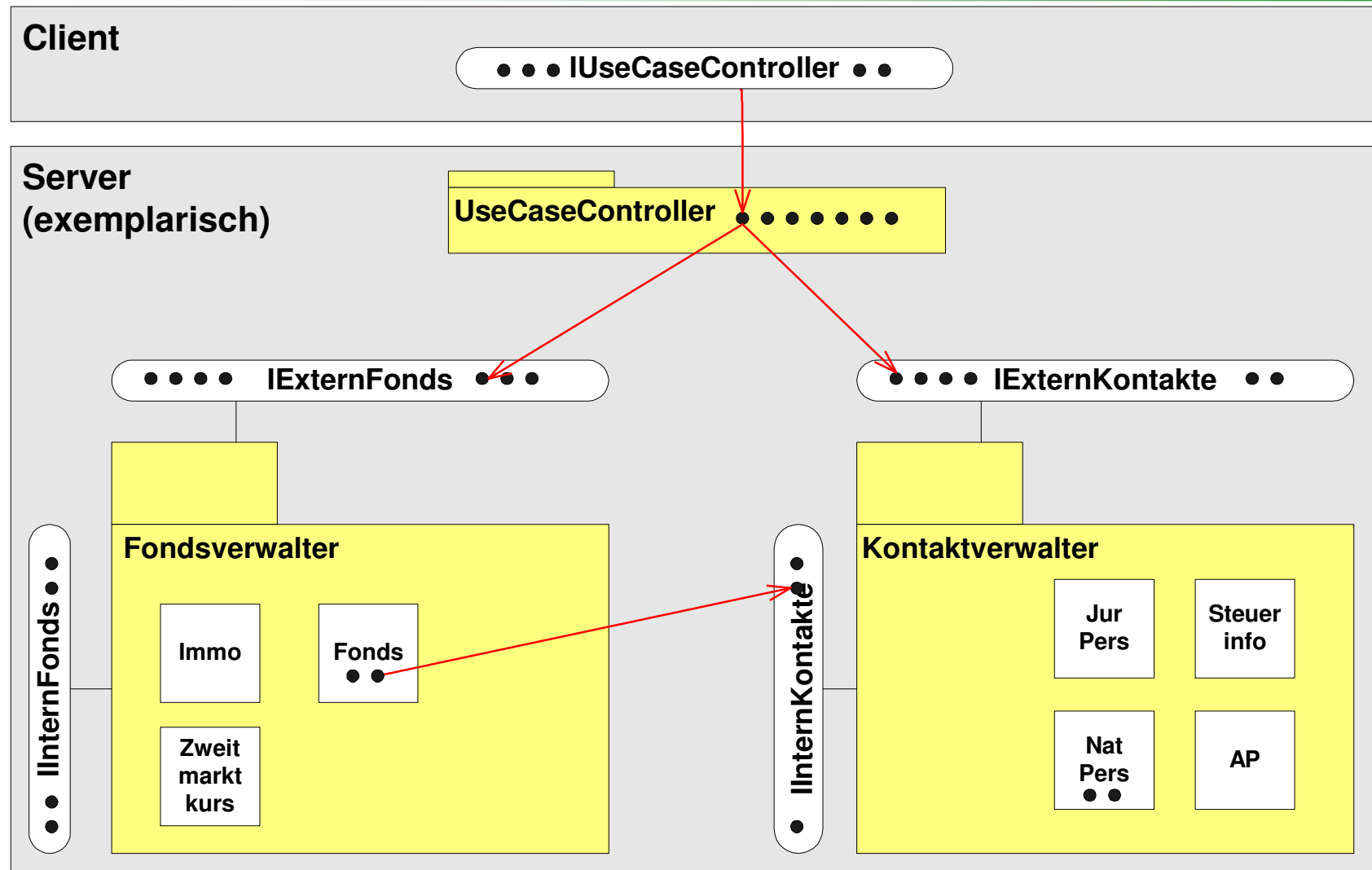
- Vermeidung von Abhängigkeiten
- Erreichen von Änderbarkeit
- Unterstützung von paralleler Entwicklung

Der Komponentenzugriff erfolgt über Verwalter

- Wie?
 - Nutzen einer Komponente **nur** über deren Schnittstellen
 - Allgemeiner Komponentenverwalter als Singleton kennt die Schnittstellen aller öffentlicher Komponenten
 - Keine zirkulären Abhängigkeiten



Komponenten verbergen Entitätsklassen, der Zugriff erfolgt nur über Schnittstellen



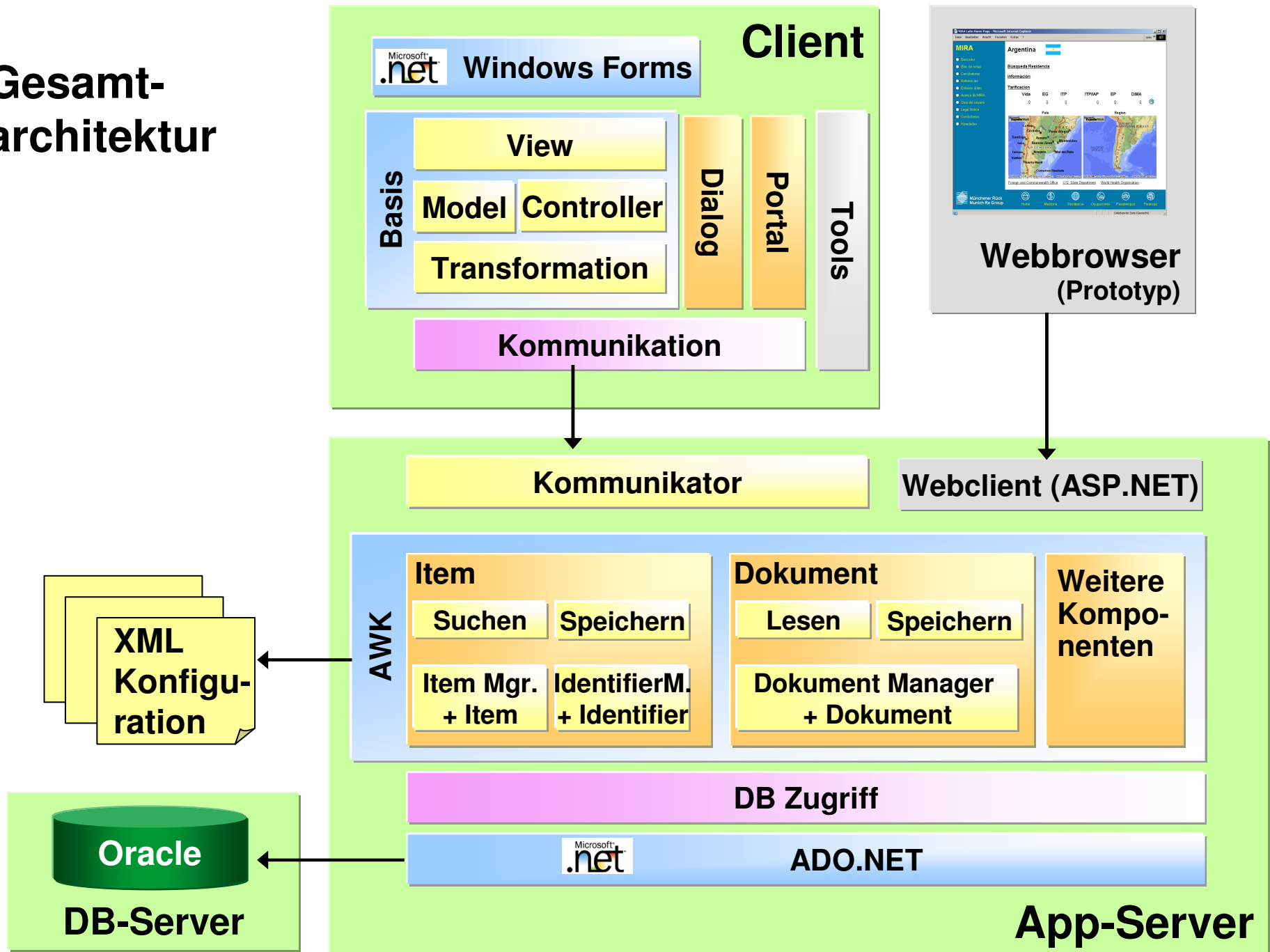
Komponenten – Codebeispiel

```
namespace leonardo.server.awk.fonds
{
    internal class Fondsverwalter :
        IKAwkFondsExtern, IKAwkFondsIntern
    {
        private Fondsverwalter() {}
        public static void RegistrierenInstanz() {
            if( Fondsverwalter.Instanz == null )
                Fondsverwalter.Instanz = new Fondsverwalter();

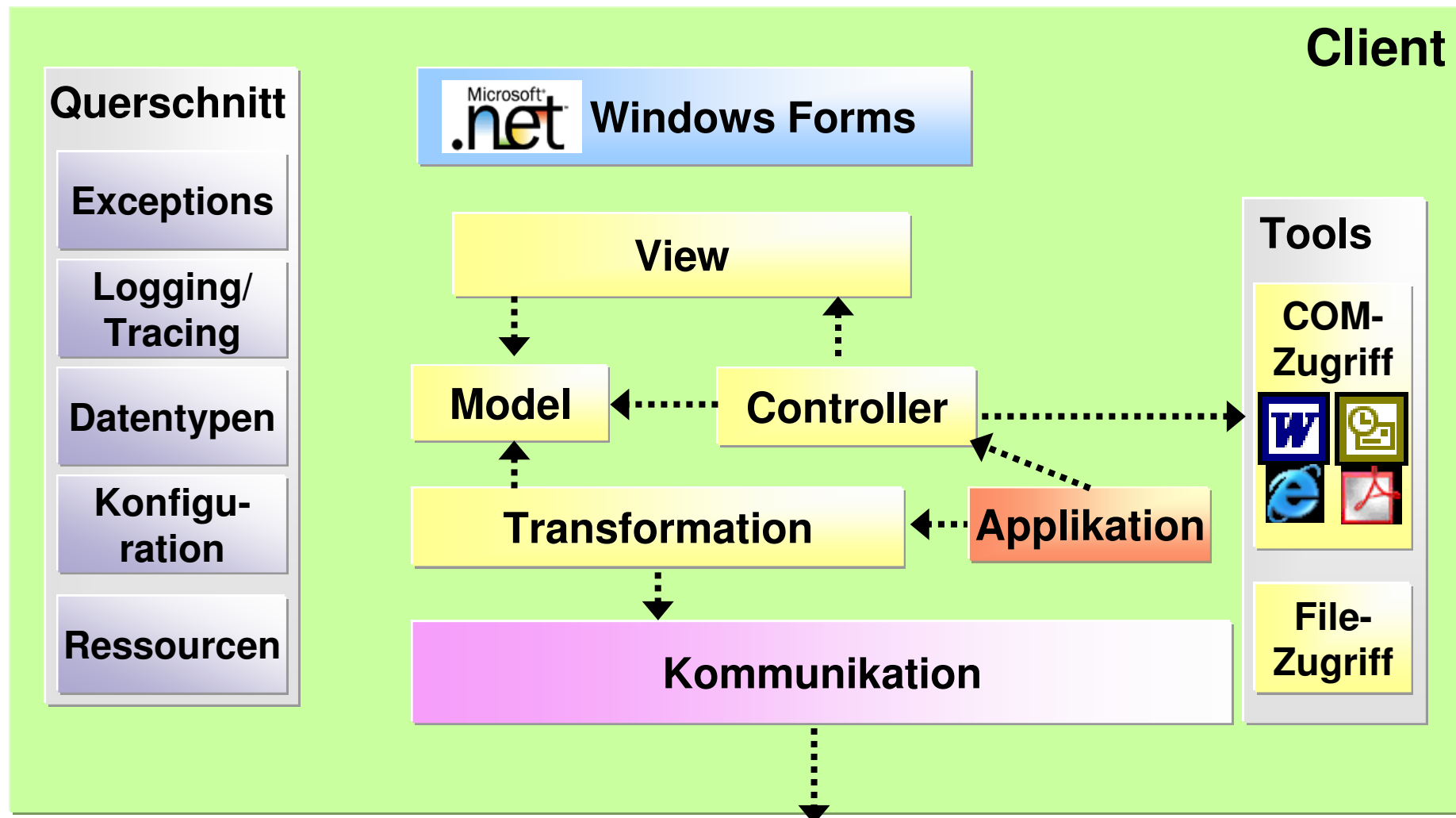
            Komponentenverwalter.Instanz.IKAwkFondsExtern =
                Fondsverwalter.Instanz;
            Komponentenverwalter.Instanz.IKAwkFondsIntern =
                Fondsverwalter.Instanz;
        }
    }

    internal class Fonds : IFonds { ... }
```

Gesamt- architektur



Das Grunddesign eines .NET-Clients unterscheidet sich nicht von Clients in anderen Technologien



Das übliche MVC-Design lässt sich mit .NET umsetzen

.NET bietet mit Windows Forms ein gutes Framework

■ Windows Forms (.NET Framework)

- Framework Komponenten, beinhalten GUI-Controls

■ View

- Verantwortlich für den sichtbaren Teil des GUIs mit Oberflächen und Steuerungskomponenten
- Kapselt die Windows Forms Controls aus dem .NET Framework

■ Model

- Datenhaltung

■ Controller

- Verarbeitet die Benutzeraktionen
- Beinhaltet die Dialogsteuerung mit Zustandsautomat (Aktivierung und Deaktivierung von Controls)
- Triggert die Serveraktionen
- Öffnet parametrisierte Subdialoge

Weitere Client-Komponenten lassen sich mit .NET leicht implementieren

■ Applikation

- Initialisierung des Clients
- Laden der lokalen Konfigurationen und der Ressourcen vom Server
- Initialisierung aller Factories und zentraler Client Komponenten mit ihren Verbindungen
- Initialisierung zentraler Dienste

■ Transformation

- Datentransformation zwischen Client- und Server-Datenmodell

■ Kommunikation

- Benutzt .NET Remoting

■ Ressourcen

beinhalten

- Daten für die Konfiguration der Dialoge
- Lokalisierungen für verschiedene Sprach- und Marktversionen
- Bilder für die Oberflächen

Im Rahmen der Projekte konnten wir in verschiedenen Themenbereichen wertvolle Erfahrungen sammeln

- Dialogsteuerung
- Datenübergabe
- Berechtigungen und Sicherheit
- Speicher- und Ressourcenverwaltung
- Dokumentenmanagement
- Workflow-Unterstützung
- Hilfesystem
- Clipboard

lassen sich analog zu bisherigen Technologien (J2EE) entwerfen und umsetzen

Bei der Verwendung von Windows Forms sind vorbereitenden Maßnahmen sinnvoll

- Wrapper für .NET-GUI-Controls
 - Einheitliche Methoden für die Steuerung
 - Zusätzliche Methoden für Prüfungen
- Funktionserweiterungen z.B. für
 - Tabellen
 - Internationalisierung
- .NET Methoden überschreiben
 - Zur korrekten Ausgabe [ToString()]
 - Zum Kopieren von Objekten [Clone()]
 - Für Loadbalancing und korrekte Identifikation [Equals(), GetHashCode()]

MIRA Maintenance - [International - Occupations - Edit - Dentist]
Subject Edit Window Help

In work
Market Versions
International
General items
Underwriting
What's New
Medical
Residential
Occupations
Items
Submenus
Additional docs
Allowed Rates
Shortcut texts
Pastimes
Financial
Calculators
References
About MIRA
User Guide
Claims
UK

Item: **Dentist**

Rating Tree
Information
Help Link Texts
Synonyms
Menu
Additional Info
Questionnaires
What's New
How To

Tree	Life	CI	ADB	TPD own	TPD any	WF	IP 4	IP 13	IP 26
Age < 45 yrs									
Own practice	0	0	0	Dec	3	3	Dec	3	3
Clinic	2%	0*1	0*1	Dec	3	Dec	Dec	3	3
Age 45 yrs and above									
Own practice	1%		0	Dec	3	3	Dec	3	3
Clinic	2%	0*1	0*1	Dec	3	Dec	Dec**2	3	3

Unindent
Indent
Preview

Save
Reset

Last updated: 12/23/01 09:11 by Wowra
Last published: 12/09/01 16:28 by Mückenhausen

Operation successfully terminated.

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

■ **Remoting**

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

Betrieb

Aufwand

Zusammenfassung

.NET Remoting ermöglicht den einfachen Austausch von Objekten

- Remoting ist ein Framework für den nahtlosen Austausch von Objekten zwischen
 - verschiedenen Anwendungen
 - verschiedenen Prozessen
 - verschiedenen Rechnern und Domänen
- Framework
 - bietet notwendige Funktionalität
 - ist erweiterbar
 - ist leicht integrierbar

Die Struktur der zu übertragenden Objekte lässt sich variieren

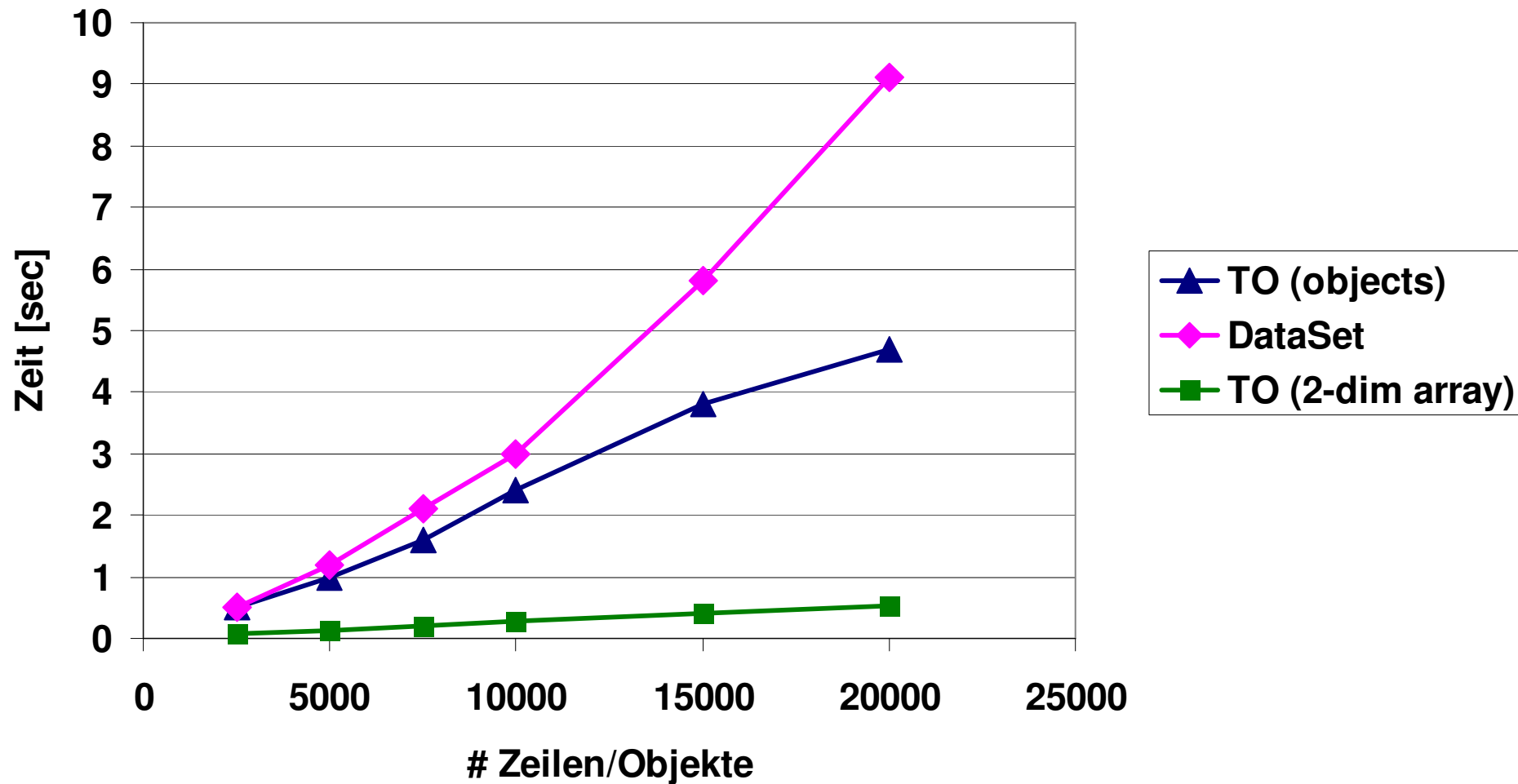
Ein Beispiel

- Es sollen folgende Daten transportiert werden:

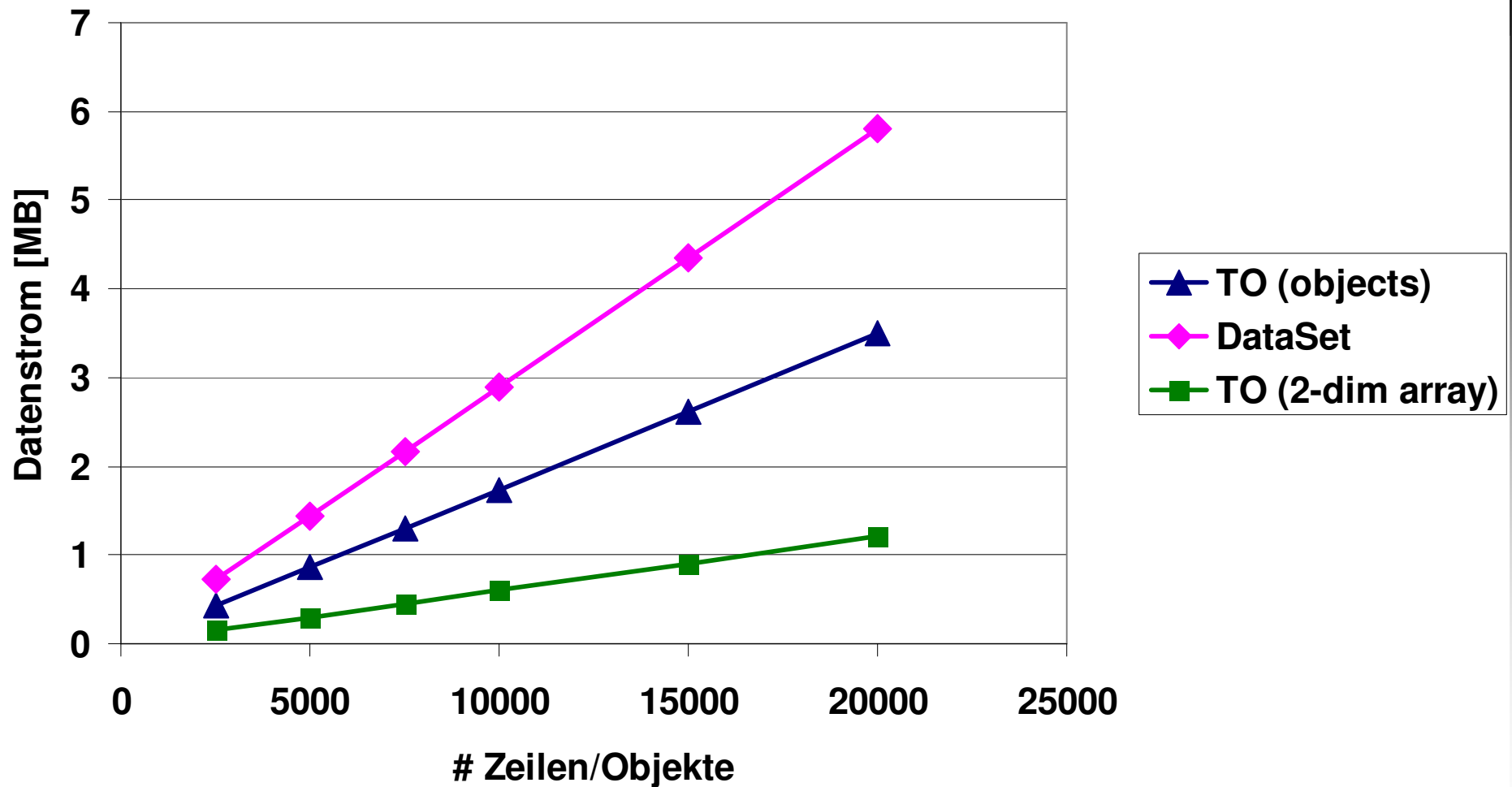
Country	Capital	Area	Zahl 1	Zahl 2	Zahl 3
Germany	Berlin	Europe	8234	3575	634,52
China	Beijing	Asia	2345	243	265,34
...	...	...	...	...	...

- Datenhaltung in Datencontainer/Transportobjekt (TO):
 - **DataSet** mit einer Tabelle
 - **TO (Objects)**
Ablage in Hashtable:
Schlüssel ist Zeilennummer, Inhalt ist Objekt mit genau einer Zeile
 - **TO (2-dim. Array)**
Ablage der Daten in einer String-Matrix (String[][])

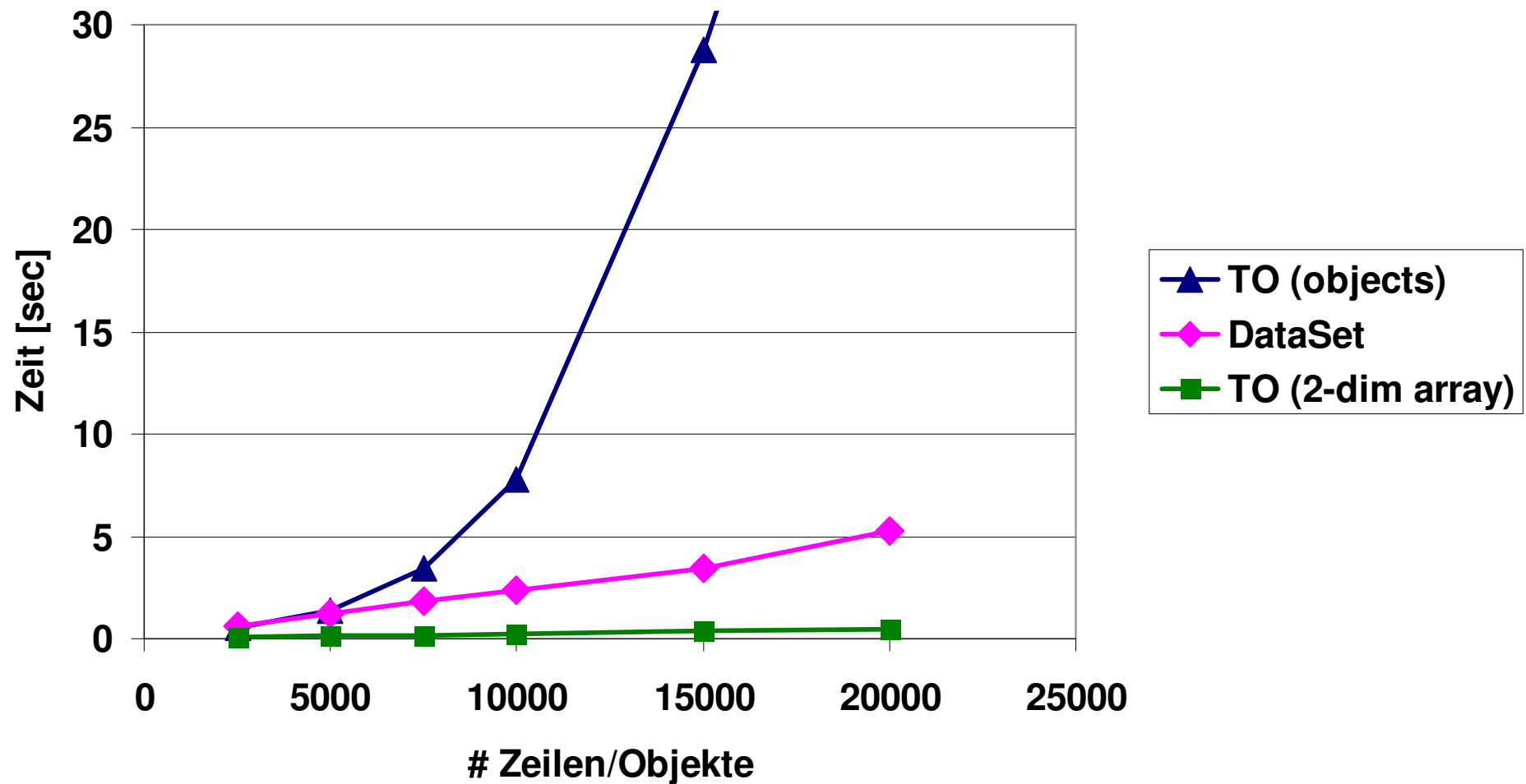
Die Zeiten für die Serialisierung variieren stark in Abhängigkeit von der Struktur der Transportobjekte



... auch die Netzlast variiert stark



Bei der Deserialisierung hängt die Performance ebenfalls stark von der Struktur der Transportobjekte ab



Damit Remoting kein Performanz-Killer wird, sind einige Aspekte zu beachten

- Keine komplexen Strukturen transportieren
(DataSet mit verlinkten Tabellen noch teurer
als Transportobjekt mit Strukturen)
- Binäre Serialisierung ist schneller
als XML/SOAP-Variante via HTTP-Channel
 - In Projektbeispielen:
XML/SOAP-Serialisierung ca. 3-fache Kommunikationszeiten
(Extremwerte: Faktor 10)
- Varianten für den Datenaustausch testen und individuell entscheiden
- ➔ Für Erzeugung der Transportobjekte in der Zugriffsschicht:
Empfehlung: DataReader statt DataSet

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

■ **COM-Anbindung**

Internationalisierung

Entwicklungsumgebung

Betrieb

Aufwand

Zusammenfassung

Für unsere Anwendungen werden COM-Zugriffe benötigt



Word

- Aufruf aus dem Client zur Bearbeitung von Word-Dokumenten (Verlinkung, HTML-Umwandlung, Verschlagwortung)
- Callback von Word für Zusatzfunktionen mit Verbindung zur Hauptanwendung
- Verarbeitung des Worddokuments zur Integration in das Gesamtsystem



Outlook

- Versenden von Mail über Outlook-Client

Anzeige als Plug-In auf dem Client von



HTML-Dokumenten mit Internet Explorer



PDF-Dateien mit Acrobat Reader

MIRA Maintenance - [UK - Residential - Edit - Russia]
Subject Edit Approve Tools Window Help

General
Market Versions
International
UK
In work
Ready for approval
Contents
Underwriting
What's new
Medical
Residential
Items
Submenus
Additional doc
Occupations
Questionnaires
Pastimes
Calculators
References
Claims
China

Item Russia

RatingTree
Information
Help Texts
Synonyms
Special Info
Menu
Supplementary Info
Questionnaires
What's New

Source

☒ All Help Texts
☐ Referenced Help Texts

Permanent Residence
Without dangerous re
With dangerous regio
Temporarily only

Help Texts

Help Texts
DANGER
TEMPORARY

Edit
Add
Delete

Preview

☒ Enable Preview

Russia

Dangerous regions in [Russia](#) include

Region	Security risks
- the Chechen Republic, - Ingushetia,	High
- Dagestan and North Ossetia, - Karachai-Cherkessia,	Considerable
- Kabardino-Balkaria (including the Elbrus area) - and to eastern and southern parts of Stavropol Krai, particularly where it borders Chechnya and North Ossetia.	Moderate

Save
Reset

Last updated: 04.12.2002 11:30:40, walthes
Last published:

Welcome to MiraMaintenance Client

Word lässt sich einfach integrieren

```
using Word;
...
try{
    Application wordApp = new ApplicationClass();
    Document doc        = wordApp.Documents.Open(...);
    Selection sel        = wordApp.Selection;
}
catch( Exception e ){ ... }
}
```

COM-Zugriffe sind leicht zu implementieren

- COM-Anbindung
von Word, Internet Explorer, Outlook und Acrobat Reader
problemlos bzgl. Performanz und Stabilität
- Word derzeit nicht direkt als Plug-In integrierbar
- Vorsicht:
Bibliotheken von Office 2000 und Office XP unterscheiden sich
 - Änderung der Aufrufe notwendig
 - Kapselung über Schnittstellen und Factory



Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

■ **Internationalisierung**

Entwicklungsumgebung

Betrieb

Aufwand

Zusammenfassung

.NET ermöglicht die einfache Umsetzung einer internationalisierten Anwendung

Anforderungen

- Oberfläche zunächst nur in Englisch
- Daten in diversen Sprachen

Anzupassen sind:

- Sortierungen
- Zeichensätze
- Format für Zahlen, Daten und Uhrzeit
- Schreibrichtung
- Fachliche Regeln

.NET bietet

- Lokalisierungsmechanismus
- Ressourcen für Lokalisierung
- lokalisierte Behandlung von Datum, Uhrzeit und Zahlen
- Unicode-Support
- Konvertierungen zwischen Zeichensätzen
- Sortierung und Stringvergleich sprachabhängig
- Sprachabhängige Schreibrichtung

Verwendung des Frameworks

- Datenhaltung vollständig in UNICODE
- Komponenten werten CultureInfo aus
- .NET Funktionen konnten weitgehend genutzt werden

Anpassungen

- Länder-Sprachen-Kulturzuordnung in Projekt und Framework unterschiedlich
- CultureInfo nicht anwendungs-, sondern dialog- bzw. feldspezifisch
- Unterschiedliche Schreibrichtungen selbst programmieren

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

■ **Entwicklungsumgebung**

Betrieb

Aufwand

Zusammenfassung

Die integrierte Entwicklungsumgebung: Viel Licht birgt auch Schatten

Tools von Microsoft

- Visual Studio.NET
- Visual Source Safe

Tools von Drittanbietern

- Rational XDE
- Test Framework NUnit

- Make Utility NAnt
- Dokumentationsgenerator NDoc
- DLL-Viewer
- Profiler, Monitoring Tool (AQTime.NET)
- Setupgenerator (MSI, InnoSetup)

Vorteile

- Gute Integration von Visual Studio und Source Safe
- Integrierter Windows Forms Designer
- Syntax Highlighting und Intellisense Codevervollständigung
- Ein- und Ausblenden von Details mittels *Regions*
- Generierung von XML-Kommentaren
- Einfache Integration weiterer Tools (ILDASM, Shellskripten etc.)

Probleme / Anpassungen

- Entwicklung im großen Team
 - Global Build nötig
 - Konventionen (globales Assembly-Verzeichnis für Komponentenreferenzierung, Registryeintrag für Assembly-Pfad)
- Fehler in einzelnen Tools (teilweise BETA)
 - Debugger
 - Visual Designer

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

■ **Betrieb**

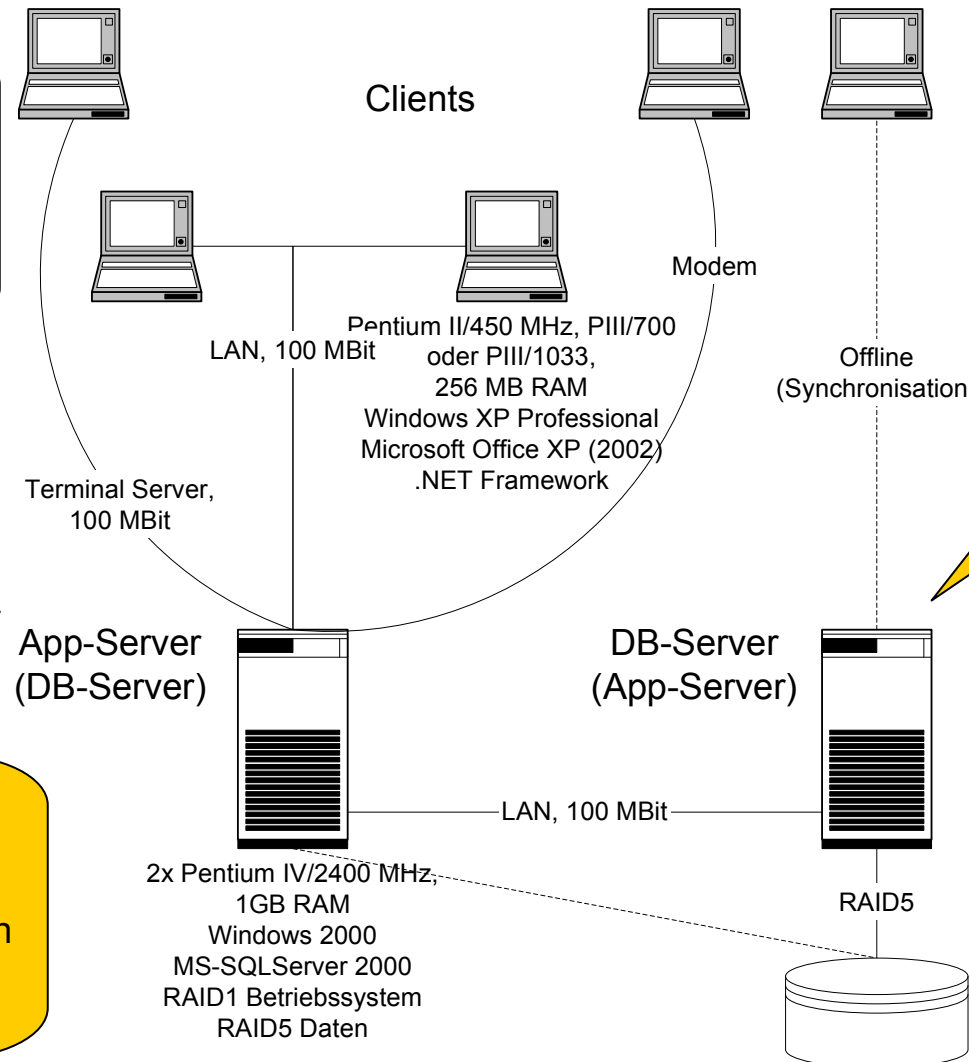
Aufwand

Zusammenfassung

Im Betrieb läuft unsere Anwendung bis heute ohne Probleme

Installation:

- Framework und SP2 problemlos
- Crystal Reports und Netinstall problematisch
- Anwendung: DLLs kopieren, Dienst installieren



Stabilität:

- bisher keine Probleme mit CLR und Framework
- täglicher Neustart des Servers
→ Garbage Collection ok

Performance:

- Antwortzeit i.d.R. < 1 sec
- etwas Verzögerung beim Laden
- Hauptlast auf DB-Server

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

Betrieb

■ **Aufwand**

Zusammenfassung

Mit geringem Einarbeitungsaufwand ist .NET schnell einsetzbar

- Einarbeitung in C# (ca. 3 Bearbeitertage (BT) pro MA bei Java-Erfahrung)
- Einarbeitung in Klassenbibliothek (je nach Thema)
- Entwicklungsumgebung
 - Installation Basiskomponenten (1 BT pro MA)
 - Einarbeitung in Tools (ca. 2 BT pro MA)
 - Nutzungskonzept (3 BT)
 - Eventuell globaler Build (ca.5-20 BT)
 - Interne Bugs
- Fehlende bzw. instabile Tools von Drittanbietern
 - Tabellen-Control
 - O-R-Mapper
- Effizienz der Entwicklung bei C#/.NET analog JAVA
 - Deutliche Vorteile von .NET bei
Office-Integration, Webservices und Web-Clients

Agenda

Projektvorstellung

Entscheidung für .NET

Vorgehen

Architektur

Erfahrungen

Remoting

COM-Anbindung

Internationalisierung

Entwicklungsumgebung

Betrieb

Aufwand

■ Zusammenfassung

Lessons Learned

Fazit: .NET trägt

- Stabilität bzgl. Last und Verfügbarkeit
 - Performanz
 - .NET Framework bietet gute Basis
 - Anpassungen bzw. Erweiterungen sind notwendig (Basisarchitektur)
- **Gesamtarchitektur** ist Standard-OO-Architektur, im Detail Anpassungen für .NET sinnvoll
 - **Frameworkklassen** sind in Einzelfällen anzupassen und zu erweitern
 - Beim **Remoting** Struktur der Objekte genau überlegen
 - **Internationalisierung** wird in .NET sehr gut unterstützt
 - **COM-Zugriffe** sind leicht integriert
 - Die **Entwicklungsumgebung**: Viel Licht birgt auch Schatten
 - **Betrieb** unserer Anwendungen bis heute problemlos

Fazit

- Beim Einsatz von .NET
 - Referenzarchitektur entwerfen und erproben
 - Entwicklungsprozesse definieren
 - Aufwand für Know-how-Aufbau und Problemlösung einplanen
- Aufwand entspricht dem für Java (J2EE), sofern
 - die Kinderkrankheiten beseitigt und
 - Basiskomponenten vorhanden sind
- Sehr gut unterstützt wird die Entwicklung von
 - Webservices,
 - Web Clients,
 - Office Integration

Kontakt



Harald Haller
Tel. 089/63812-431
haller@sdm.de

Alexander Ramisch
Tel. 089/63812-386
ramisch@sdm.de

... oder auf dem Stand hier auf der Konferenz

sd&m AG
Thomas-Dehler-Straße 27
81737 München
Telefon (089) 63812-0
Fax (089) 63812-220
info@sdm.de