Who needs Outlook?

Ein verteiltes und komponentenbasiertes Projekt zur Kontakt- und Terminverwaltung im .NET Framework

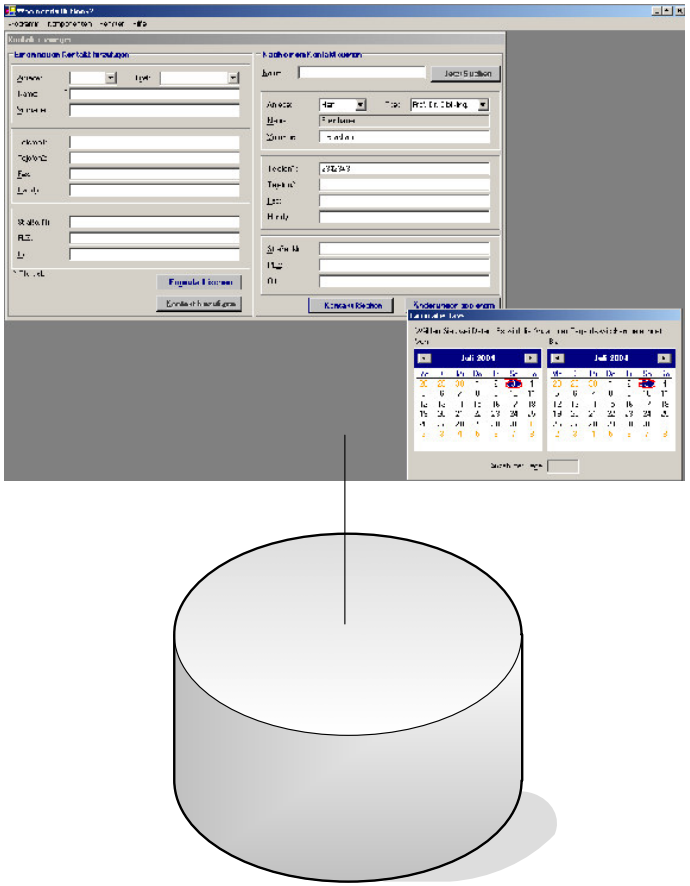
Claudia Vogel

Stefan Hüttenrauch

Tobias Queck

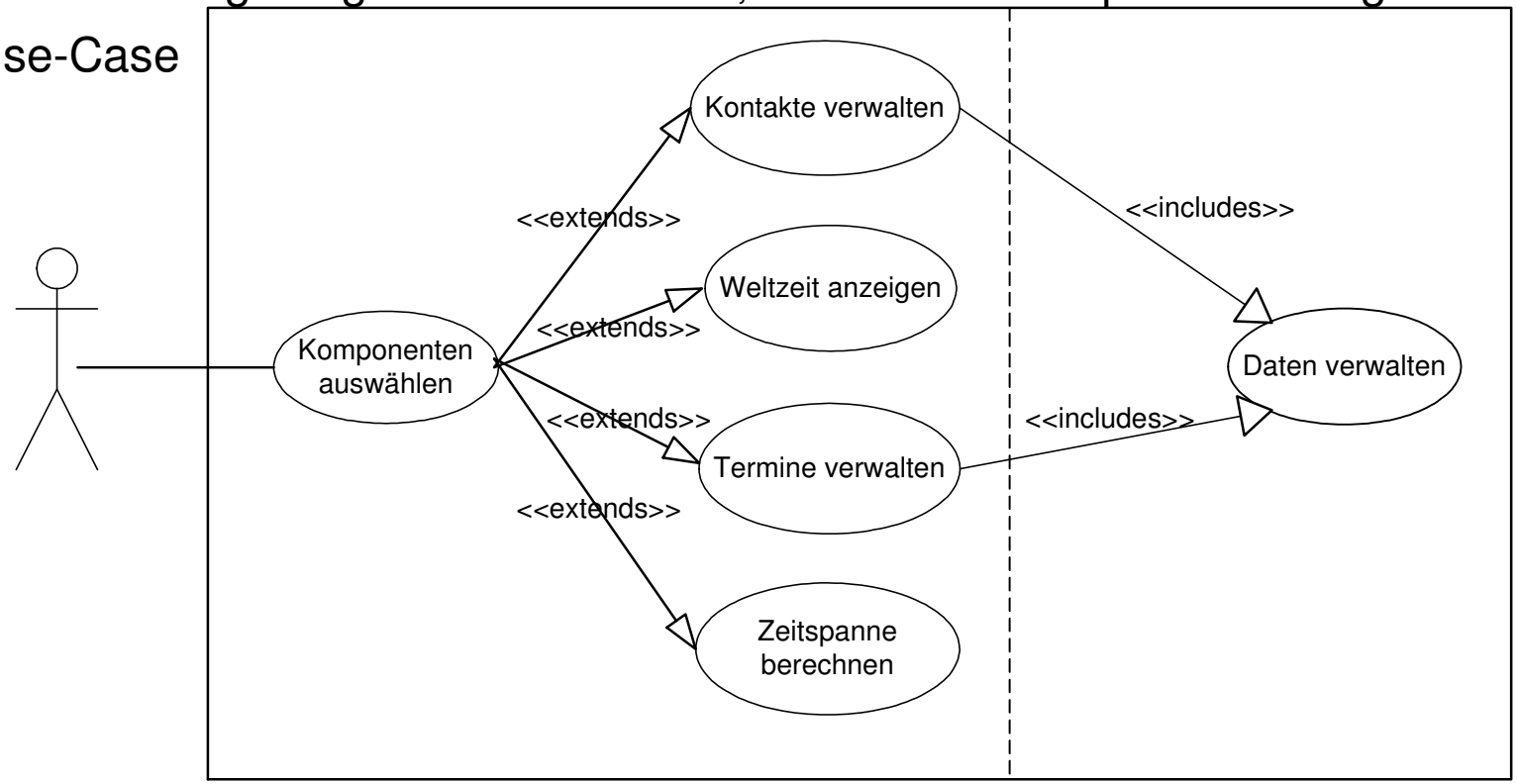
Gliederung

- Überblick über das Projekt
  - Idee
  - Architekturentscheidung
  - Verwendete Design-Pattern
- Technische Details
  - Einbinden von Assemblies
  - Kommunikation mit der Datenbank per Remoting
  - Tests mit CS-Unit / NUnit
- Präsentation
- Ausblick und Vergleich mit anderen Plattformen
- Die Arbeit im Team
  - Vorgehensweise
  - Probleme
  - Erfahrungen für zukünftige Projekte
- Literatur



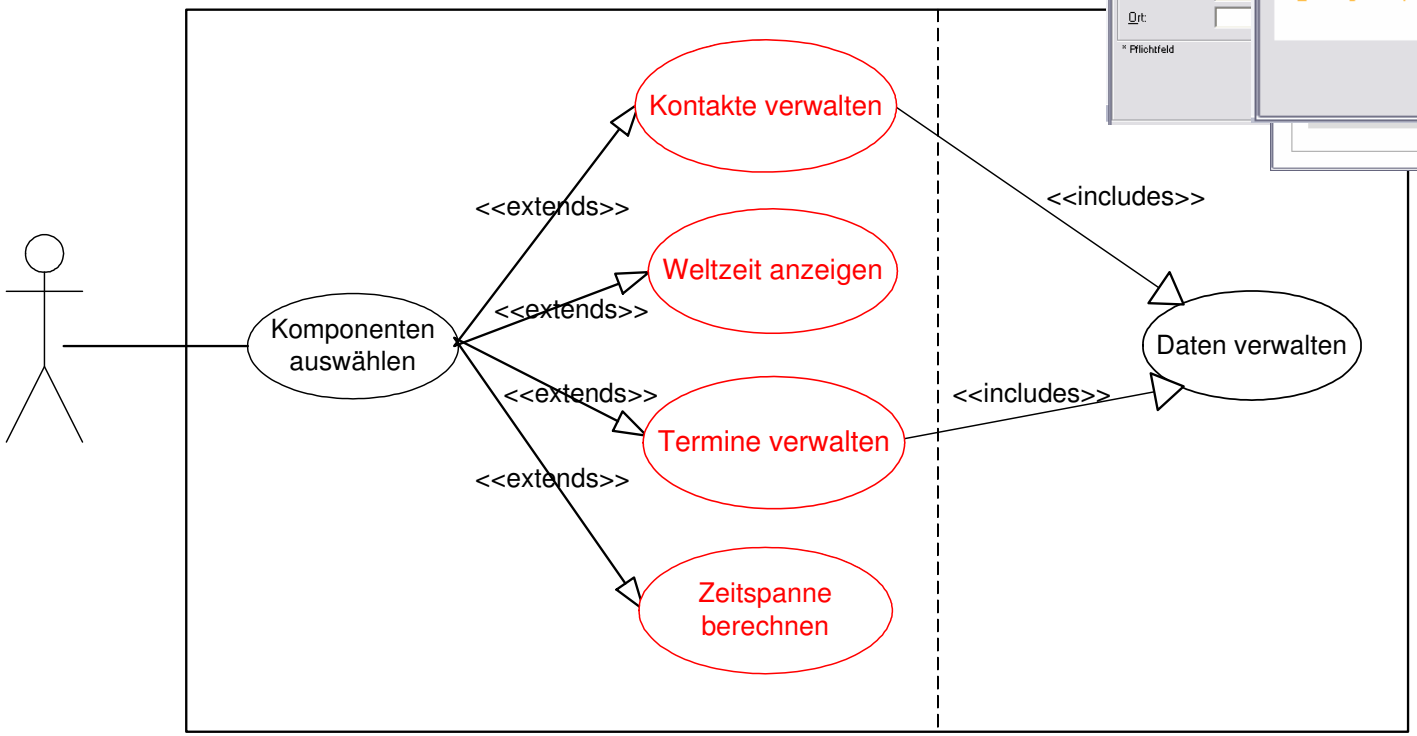
Überblick über das Projekt: Idee 1/3

- Die Idee
  - Ausgewählte Funktionen von Outlook als Komponenten realisieren
  - Datenhaltung nötig: SQL-Datenbank; Kommunikation per Remoting
  - Use-Case



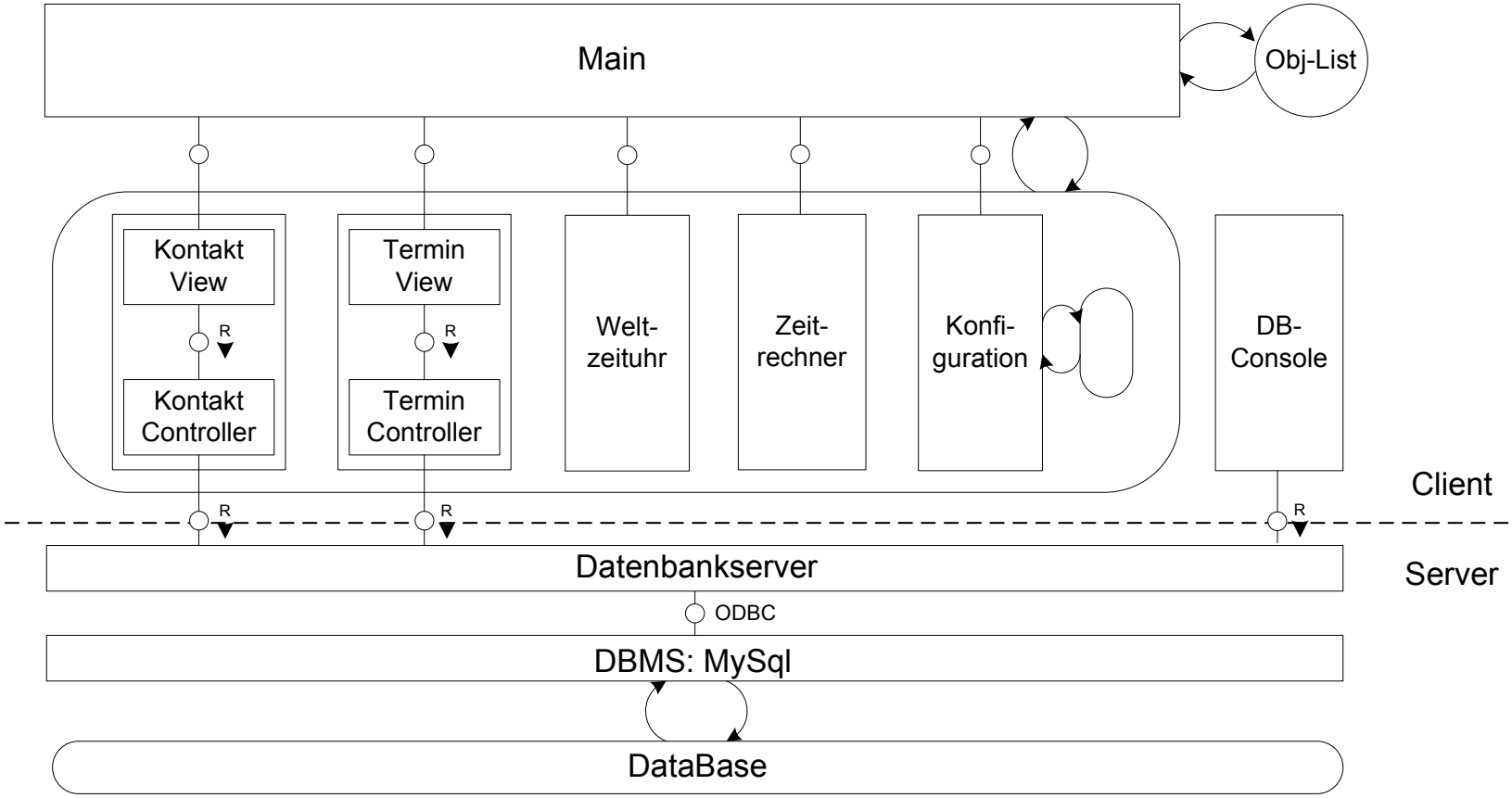
Überblick über das Projekt: Idee 2/3

Einige Komponenten und ihre Funktionen



Überblick über das Projekt: Idee 3/3

- Aufbauplan in FMC



Überblick über das Projekt: Architekturentscheidung

- Framework: Warum .NET ?
  - Zusammenwirken der verschiedenen Sprachen erfahren
  - C#, VB .NET, Visual C++, J#
  - Jeder Entwickler kann seine ‚Lieblingssprache' wählen

- Systemvoraussetzungen
  - Windowsplattform
  - .NET Framework
  - MyODBC 3.51.06
  - MySQL Server Version 1.4

Überblick über das Projekt: Verwendete Design-Pattern

- Model-View-Controller
  - Views durch die verschiedenen GUIs der Komponenten gegeben
  - Prinzipiell hat jede View einen eigenen Controller, aber auch Komponentenübergreifende Nutzung möglich
  - Modell in Form des DatabaseServer
- Façade
  - Anbindung verschiedener Datenbanken möglich
- Proxy
  - Durch Activator.GetObject()
- Singleton
  - Singleton Object auf DatabaseServer-Seite
- Reflection
  - .NET Reflection zum dynamischen Einbinden von Assemblies

Technische Details
Einbinden von Assemblies

Reflection

```
using System;
using System.Reflection;

namespace MainComponent  {
   public class Form1 : System.Windows.Forms  {
      other stuff
      private void Form1_Load(object sender, System.EventArgs e)  {
         IsMdiContainer = true;
      }
      private void loadComponent(String CompPfadAndName)  {
         Assembly aktuelleComp = Assembly.LoadFrom(CompPfadAndName);
         Type[] TypesOfComp = aktuelleComp.GetTypes();

         foreach(Type t in TypesOfComp)  {
            Object obj = Activator.CreateInstance(t);
            if (obj is System.Windows.Forms.Form)  {
               (obj as System.Windows.Forms.Form).MdiParent = this;
               (obj as System.Windows.Forms.Form).Show();
```

Technische Details
Kommunikation mit der Datenbank 1/2 – Remoting

BeispielCode

```
RemotingConfiguration.Configure("remoting.config");

SqlCommand = (IDatabase) Activator.GetObject(typeof(IDatabase), Url);
```

```
<configuration>
 <system.runtime.remoting>
  <application>
   <service>
        <wellknown  mode = "Singleton,
                    type = "Database.SqlCommands, Database"
                    url  = "tcp://localhost:1234/DatabaseService" />
   </service>
   <channels>
        <channel type="System.Runtime.Remoting.Channels.Tcp.TcpChannel,
                    System.Runtime.Remoting" />
   </channels>
  </application>
 </system.runtime.remoting>
</configuration>
```

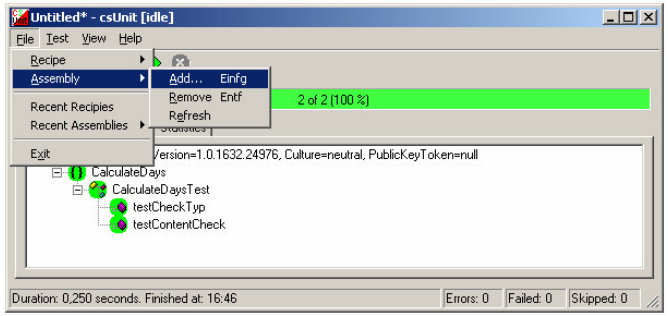
Technische Details
Kommunikation mit der Datenbank 2/2 - MySQL

- DBMS “MySQL Server Version 1.4”
  - Datenkonsistenz
  - Datenhaltung
  - Sicherheit (Authentifizierung)
  - Datenverteilung (SQL DumpFiles)
- ODBC Connector als Schnittstelle zwischen .NET und DBMS
- Beispielcode

```
OdbcConnection Connection = new
OdbcConnection(ConnectParameter);

Connection.Open();

OdbcCommand Command = new OdbcCommand(SqlQuery, Connection);
DataReader = Command.ExecuteReader();

if (DataReader != null) DataReader.Close();
Connection.Close();
```

Technische Details
Tests

- Testverfahren
  - manuelle Tests
  - CS-Unit
  - NUnit

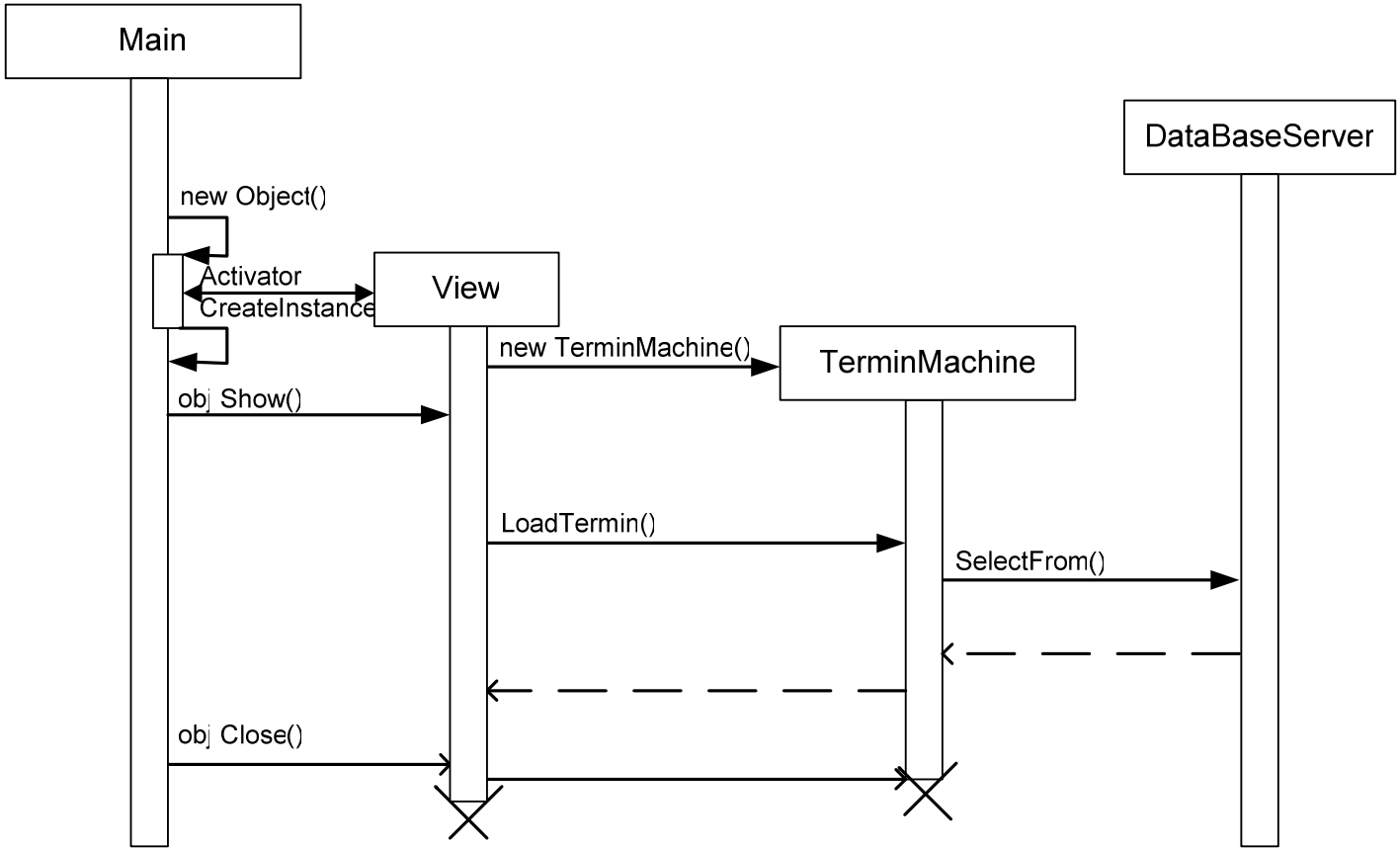


- Komponententests
  - DatabaseServer - Datenüberprüfung nach verschiedenen Befehlsfolgen
  - Komponenten - ‘TestDummies’ anstelle einer Datenbankanbindung
  - Main - Laden von DummyComponent

- Integrationstest
  - BottomUp Verfahren
  - Anbindung der Datenbank an die einzelnen Komponenten
  - Einladen der Komponenten

Präsentation

- Sequenzdiagramm des Ladens der Terminverwaltungskomponente



Ausblick

- Ausblick:
  - Funktionalität und Bedienkomfort der Komponenten erhöhen
  - Weitere Komponenten
  - Synchronisation paralleler Datenbankzugriffe
  - Portabilität

Vergleich mit anderen Plattformen

- Wie siehts aus in Corba
  - Server legt Instanz des Servant an
  - Legt Referenz in IOR-File ab
  - Client erzeugt Objekt aus IOR-File
- Wie siehts aus in Java: serialisierte JBs einbinden
  - Ähnlich .NET Reflection

```
private void loadLightBulb() {

try { ObjectInputStream is = new
        ObjectInputStream(new
        FileInputStream("lb1.ser"));

LightBulb bulb = (LightBulb)is.readObject();
```

```
ORB mySOrb = ORB.init(args, null);

IImpl EofI = new IImpl();

CORBAUtil.writeIOR (mySOrb, EofI,
"Filename");
```

```
ORB myCOrb = ORB.init(args,props);

Org.omg.CORBA.Object obj =
CORBAUtil.readIOR(myCOrb,
"Filename");
```

Die Arbeit im Team: Vorgehensweise

- Brainstorming zu Inhalten
  - Realisierungsvorstellungen festgehalten
- Getrennte Entwicklung der Komponenten und der Datenbankanbindung
- Gruppentreffen
  - Probleme und Verbesserungsvorschläge diskutiert
  - Eigene Erkenntnisse und Erfahrungen an das Team weitergegeben
- Anbindung der Komponenten an die DB
- Anbindung der Komponenten an die ‚Main‘
  - XP (meist zu zweit)
  - Teilweise jeder für sich

Die Arbeit im Team: Probleme

- Das Problem mit der Zeit
  - Alle unter einen Hut kriegen ist schwierig
- Zu viele Versionen und Änderungen gerade in der Schlussphase
  - Wer hat die aktuellste Version
  - Jeder schreibt an allem rum (Endphase)
  - Konsistenzhaltung unübersichtlich und schwierig
- Keine Dokumentation von Veränderungen

Die Arbeit im Team: Erfahrungen für zukünftige Projekte

- XP für kleine Projekte gut geeignet
  - Spart Zeit
  - Erhöht Lerneffekt
  - Ab drei Personen nicht mehr so gut geeignet, meist sitzt einer rum
- Dennoch: individuelle Arbeit ist nicht immer ersetzbar
  - Vor allem bei Einarbeitung in neue Technologien
- Versionskontrolle nötig
- Änderungsbefugnisse vergeben
- Zeitplanung aufstellen, feste Termine frühzeitig planen

Literatur

- Microsoft Visual Studio .NET 2003-Dokumentation
- RAMMER, Ingo: Advanced .NET Remoting. Springer Verlag GmbH & Co. KG Heidelberg, 2002
- WESTPHAL, Ralf: .NET kompakt. Spektrum akademischer Verlag GmbH Heidelberg/Berlin, 2001
- HOFMANN, JOBST, SCHABENBERGER: Programmieren in COM und CORBA. Carl Hanser Verlag München/Wien, 2001
- <http://www.csunit.org/index.php> (Stand 10.06.2004)
- <http://gd.tuwien.ac.at/languages/java/GoToJava2/html/k100275.html#sectlevel4id044003004002> (Stand 28.06.2004)