JLinx

# FlowSim 2004

.net

Institute for Computerapplications in Civil Engineering -
Technical University of Braunschweig.

Microsoft .net development

Studienarbeit von Jan Linxweiler WS 2003/2004

# Studienarbeit von Jan Linxweiler WS 2003/2004

Institut für Computeranwendungen im Bauingenieurswesen
Technische Universität Braunschweig

1. Aufgabenstellung
2. Microsoft .NET
3. Der FlowSim in der Anwendung
4. FlowSim Objektmodell
5. Ergebnisse
6. Ausblick
7. Fragen

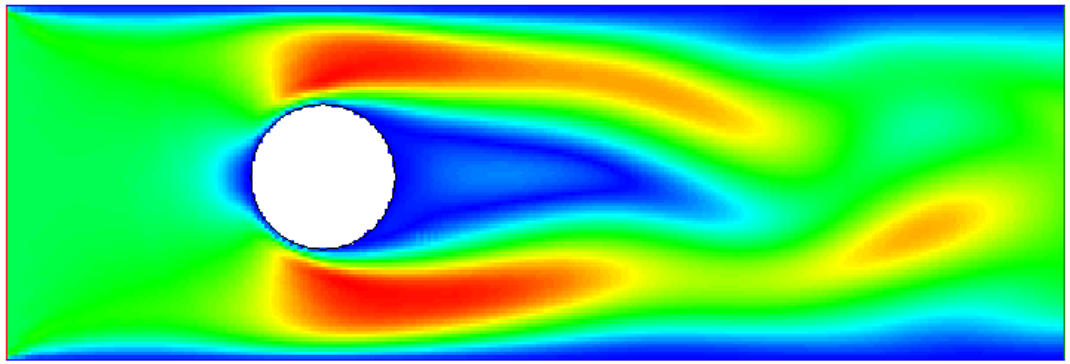
# Aufgabenstellung

Entwicklung eines auf der Lattice-Boltzmann-Methode basierenden interaktiven Strömungssimulators in C# mit verschiedenen Berechnungskernen

# 1.

# Interaktiver Strömungssimulator

*2D-Strömungsprobleme*
*uniforme Gitter*



Geometrie während der laufenden Simulation modifizierbar

# 2.

# Lattice-Boltzmann-Methode

## Numerische Algorithmen

## Simulation von Strömungs- und Transportprozessen

*Laminare, schwach kompressible Strömungen*

*Im Bauwesen i.A. nicht kompressible Strömungen*

# 3.

# C#

*Von Microsoft entwickelte Programmiersprache für*

# .NET

*Neue Plattform für Anwendungsentwicklung*

# 4.

# Verschiedene Berechnungskerne

*Warum ?*

*Numerische Simulation bedeutet:*

$$N \sim Re^3$$

Hoher Rechenaufwand

Viele Speicherbewegungen

Wichtig: performanter Berechnungskern!

# Fragen:

1. Hohe Ausführungsgeschwindigkeiten unter .NET ?
2. Optimierungsmöglichkeiten ?
3. Interoperabilität ? (vorhandener Code ?)
4. Vergleich: .NET - Nativer Code (C++)


Microsoft
.net
development

# .NET

*(managed Code)*

## *Was ist das?*

*.NET is surrounded by too many buzzwords and generalities to be understandable. I'm not sure the company knows what .NET is, or whether anybody does.*

*John Dvorak*

# .NET

*(managed Code)*

## *Was ist es nicht…*

- *Betriebssystem*
  - *Weiterhin Windows 2003, XP, 2000, Me, 98*
- *Programmiersprache*
  - *Weiterhin C++, VB, Java, Delphi, Fortran,….*
  - *Neu: C#  („C sharp“)*

# Next Generation Web Services

*(NGWS)*

Sommer 2000

*Beginn der .NET Initiative…*

# Microsoft .NET

- .NET Web Services
  - Ständig verfügbare Internetdienste
  - Messenger, Code-Updates, Suchmaschienen
- *.NET Enterprise Server*
  - SQL Server 2000, Exchange Server 2000, …
  - Integration in .NET Anwendungen
- .NET Framework

  ....

# .NET Framework

*Komponenten und Richtlinien*

*Anwendungen erstellen, kompilieren und ausführen*

# .NET Framework

- *Common Language Runtime (CLR)*
- *Framework Class Library (FCL)*
- *Common Type System (CTS)*
- *Common Language Specification (CLS)*

# .NET Framework

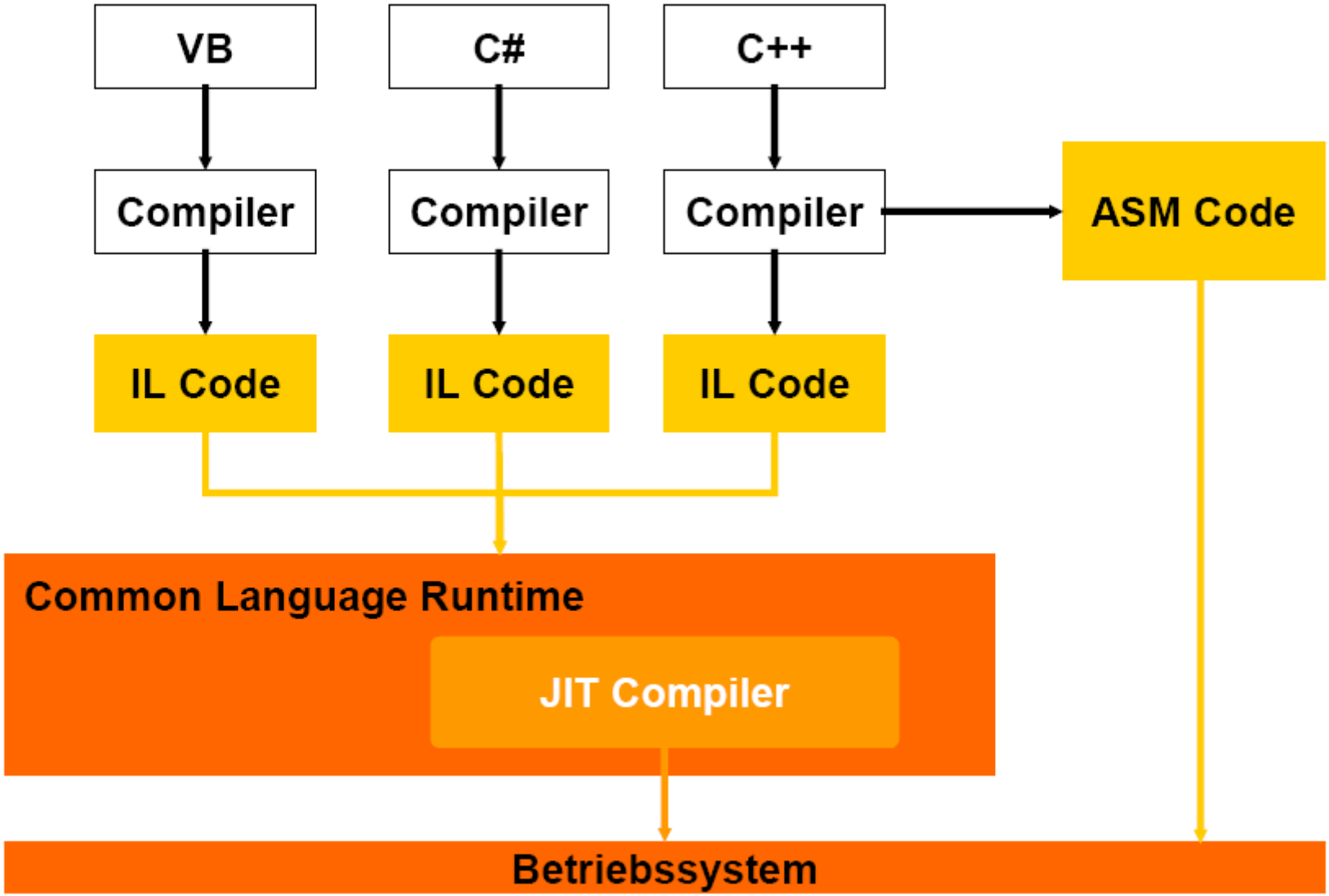
- *Common Language Runtime (CLR)*
- *Framework Class Library (FCL)*
- *Common Type System (CTS)*
- *Common Language Specification (CLS)*

# Common Language Runtime (CLR)

*(Laufzeitumgebung – Kern von .NET)*

*Vergleichbar mit der Java-VM*

*Führt managed Code aus…*


VB
C#
C++
Compiler
Compiler
Compiler
ASM Code
IL Code
IL Code
IL Code
Common Language Runtime
JIT Compiler
Betriebssystem

# Common Intermediate Language (CIL)

- *Compiler erzeugen keinen nativen Code sondern eine prozessorunabhängige Zwischensprache*

CIL wird oft auch als IL bezeichnet

# Managed Code

- *Sämtlicher Code wird unter Aufsicht der Common Language Runtime ausgeführt*
  - Runtime führt Sicherheitsüberprüfungen aus
  - Runtime übernimmt Speicherverwaltung und Fehlerbehandlung
  - Runtime führt Versionsüberprüfungen aus
  - Dieser Code wird als **Managed Code** bezeichnet

# Common Language Runtime (CLR)

*(Laufzeitumgebung – Kern von .NET)*

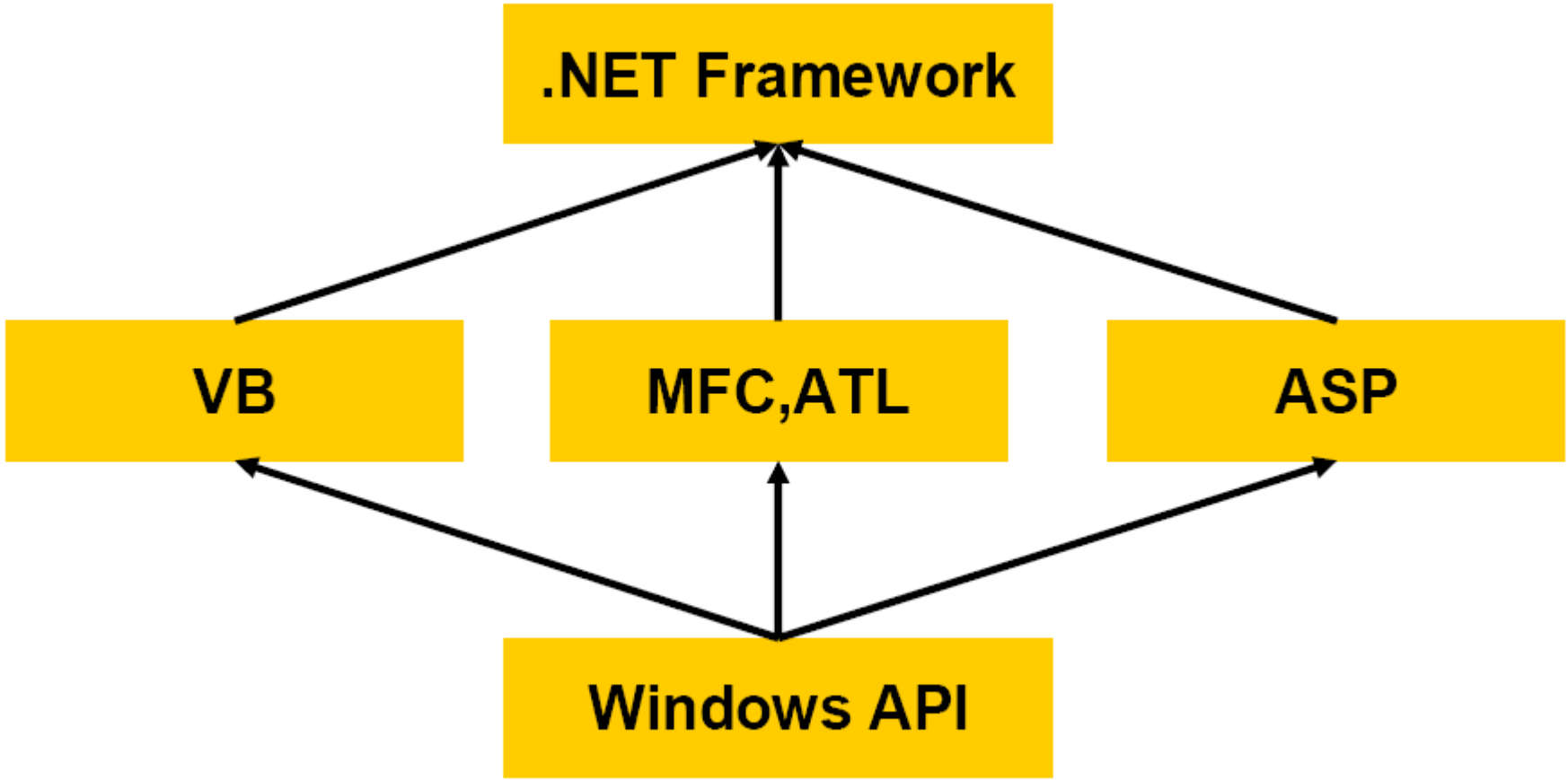
- *JITer* - Just in Time Compiler
- *Garbage Collector* - automatische Speicherverwaltung
- *Type Checker* - Überprüfung Typkonvertierungen
- *Class Loader* - Klassen der Laufzeitumgebung laden

…

- *Versions-, Prozessraum-, Sicherheitsmanagement*

# .NET Framework

- *Common Language Runtime (CLR)*
- *Framework Class Library (FCL)*
- *Common Type System (CTS)*
- *Common Language Specification (CLS)*


.NET Framework
VB
MFC,ATL
ASP
Windows API

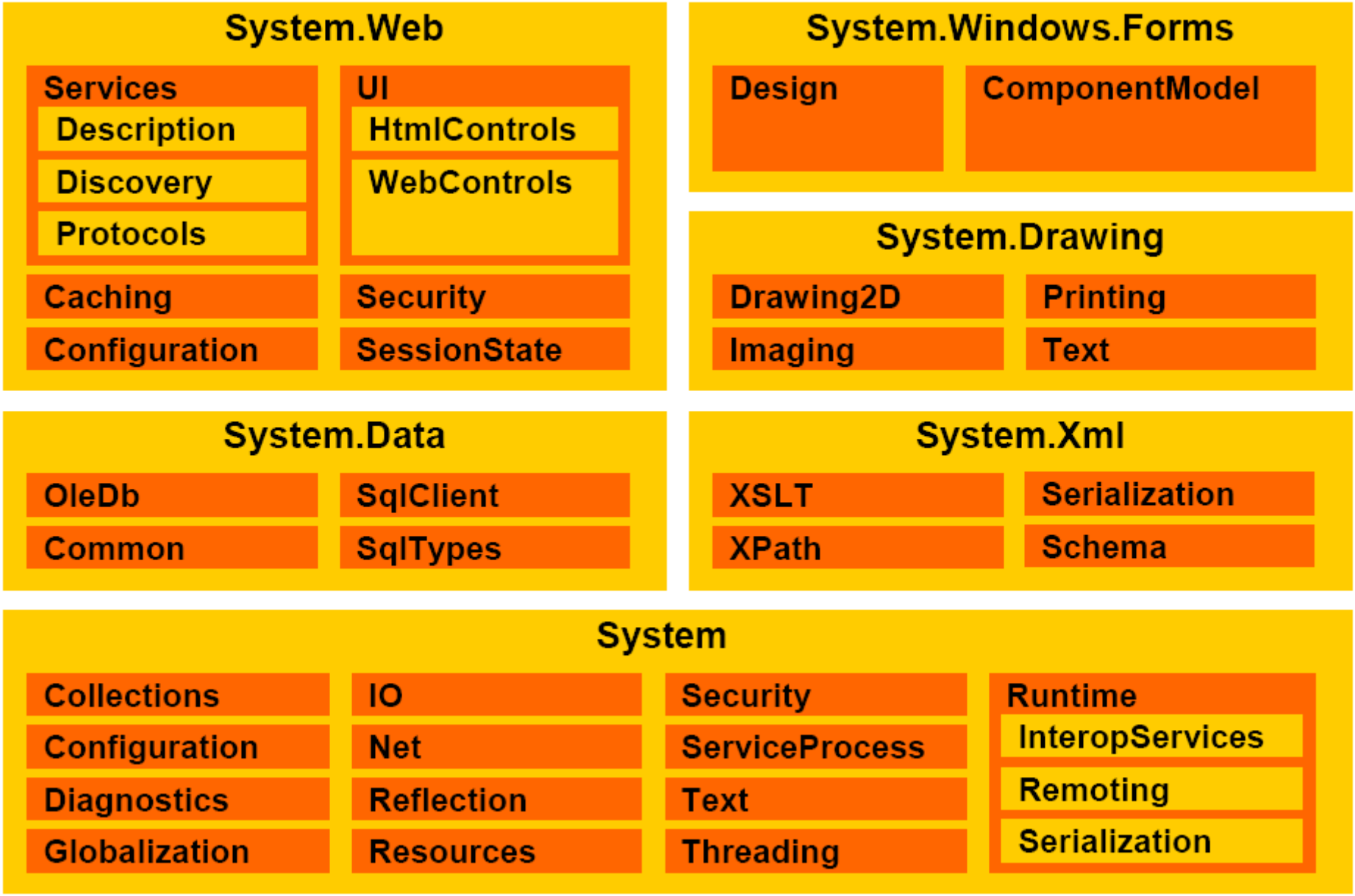
# Framework Class Library (FCL)

*(.NET-Klassenbibliothek)*

- durchgehend objektorientiert - Basistyp *Object*
- mehr als 2400 Klassen
- in Namespaces organisiert

- GUI - Windows Forms
- ASP.NET - Web Forms
- ADO.NET - Datenbanken

*Longhorn: Win32-API → WinFX*


System.Web
Services
Description
Discovery
Protocols
UI
HtmlControls
WebControls
Caching
Security
Configuration
SessionState
System.Windows.Forms
Design
ComponentModel
System.Drawing
Drawing2D
Printing
Imaging
Text
System.Data
OleDb
SqlClient
Common
SqlTypes
System.Xml
XSLT
Serialization
XPath
Schema
System
Collections
IO
Security
Runtime
InteropServices
Configuration
Net
ServiceProcess
Remoting
Diagnostics
Reflection
Text
Serialization
Globalization
Resources
Threading

# .NET Framework

- *Common Language Runtime (CLR)*
- *Framework Class Library (FCL)*
- *Common Type System (CTS)*
- *Common Language Specification (CLS)*

# Common Type System (CTS)

- enthält alle verfügbaren Datentypen
- definiert Vorschriften für eigene Typen
- Basis für Sprachunabhängigkeit

- C# (Systemsprache) unterstützt alle verfügbaren Datentypen

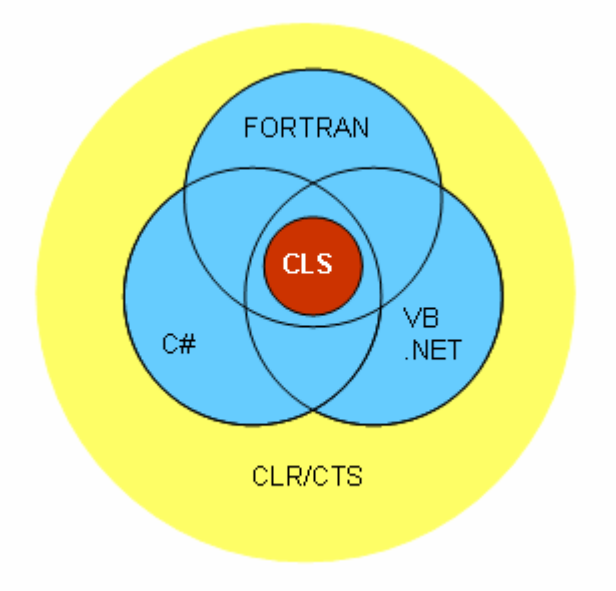
- muß nicht von allen Sprachen unterstützt werden → CLS

# .NET Framework

- *Common Language Runtime (CLR)*
- *Framework Class Library (FCL)*
- *Common Type System (CTS)*
- *Common Language Specification (CLS)*

# Common Language Specification (CLS)

- Untermenge des CTS




- muß von allen Sprachen unterstützt werden um Interoperabilität zu gewährleisten

# Allgemeine Vorteile

- *Automatische Speicherverwaltung (keine Zeiger)*
- *Einheitliche (komfortable) Klassenbibliothek*
- *Plattformunabhängigkeit*
- *Sprachunabhängigkeit*
- *Ausführungssicherheit*
- *XCOPY-Deployment*
- *Hoher Entwicklungskomfort*

# Spezielle Vorteile

- *Automatische Speicherverwaltung*
- *CPU-spezifischer Code*
- *Typsicherheit*
- *Range Checking*
- *CLR - interne Optimierungsfunktionen*
- *Vorhandener Code kann integriert werden*

# Nachteile

*Zusätzlicher Verwaltungsaufwand durch die CLR*

Ausführungsgeschwindigkeit!?

*(Bindung an Windows)*

numerische Simulation:

# Nativer Code

*(unmanaged Code)*

- sequenziell (objektorientiert)
- C, Fortran, (C++)
- Plattformspezifisch
  - Unix, Windows
  - Pentium, Athlon
- Zeiger
- Speicherverwaltung
  - Aufgabe des Programmieres

# Nativer Code

*(unmanaged Code)*

- sequenziell (objektorientiert)
- eine Programmiersprache
- plattformspezifisch
- Speicherverwaltung!!!!
  - Programmierer
- Zeiger

# .NET

*(managed Code)*

- objektorientiert
- sprachunabhängig
- (plattformunabhängig)
- Speicherverwaltung
  - automatisch
- Referenzen (Typsicherheit)

# C#

# C#

*(„C sharp“)*

- Von Anders Hejlsberg bei Micosoft entwickelt
- .NET Systemsprache
- Vollständig objektorientiert
- An C++ und Java angelehnt
- Setzt auf dem CTS auf

# Neues in C#

- Interfaces
  - Ähnlich einer abstrakten Klasse
  - Interface: Definition – Klasse: Implementierung
  - Eine Art „Vereinbarung“
- Eigenschaften
  - Kontrollierter Zugriff auf private Felder
  - Getter-, Setter-Methoden

# Hallo Welt in C#

```
namespace Studienarbeit
{
    public class MyFirstClass
    {
        public MyFirstClass()
        {
        }

        public void WriteHallo()
        {
            System.Console.WriteLine("Hallo, Welt !");
        }
    }
}
```

```
using Studienarbeit;

class HelloWorld
{
    static void Main(string[] args)
    {
        MyFirstClass firstClass = new MyFirstClass();
        firstClass.WriteHallo();
        System.Console.ReadLine(); //Auf Enter warten
    }
}
```

# FlowSim 2004 .NET

*Vorführung*

# FlowSim 2004 .NET

*Ausgangspunkt*

1. FlowSim 2003
2. sequenzieller LBGK-Code

# Rad nicht neu erfinden…

*Details*

- LBGK-Methode
- Affine Abbildungen
- Bresenham Algorithmus
- Formfunktionen

# FlowSim Objektmodell

## Modularer Aufbau

### Komponenten

- Leicht austauschbare Programmteile
- Unabhängig zu implementieren
- Kommunikation über Schnittstellen

Komponentenbasierte Programmierung

*.NET Steuerelemente*

# Aufgabenteile:

## 1. **GUI** *(Graphical User Interface)*

- Darstellung der Geometieobjekte
- Interaktives Erstellen & Modifizieren von geometrischen Objekten
- Steuerung & Darstellung der Simulation

## 2. **Datenmodell**

- Erstellen, Speichern und Modifizieren von Geometrieobjekten

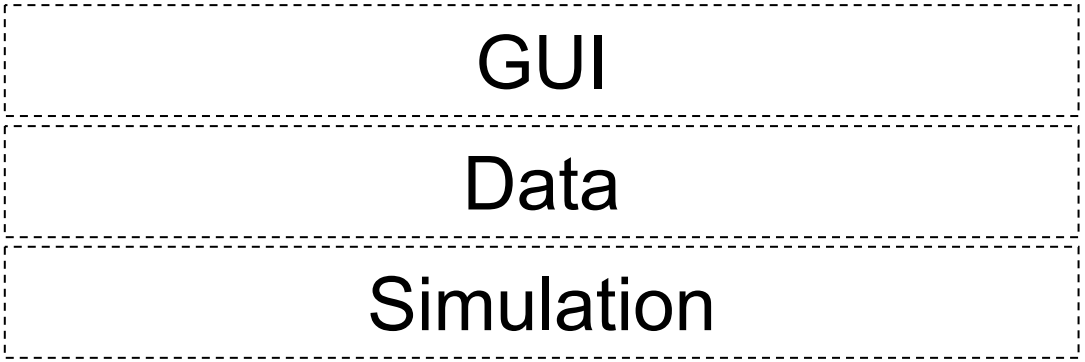
## 3. **Berechnungskerne**

- Lattice-Boltzmann-Methode
- Unterschiedliche Technologien (.NET vs. Native C++)

Ziel…

# FlowSim Objektmodell

*Drei Schichten*



# *Drei Schichten*

## GUI

*Darstellung und Benutzerinteraktion*

- *Variierte Darstellung von Geometrieobjekten*
- *Interaktives Erstellen und Modifizieren von Geometrieobjekten*
- *Darstellung & Steuerung der Simulation*

## Data

## Simulation

# *Drei Schichten*

GUI

## Data

*Datenhaltung und -modifikation*

- *Erstellen, Speichern, Modifizieren und Persistieren von Daten*
- *Geometrieobjekte, Geometriematrix*

Simulation

# *Drei Schichten*

GUI

Data

Simulation

*Berechnungskerne*

- *LBGK-Methode*
- *Momentenmodell*

# Namespaces

- *FlowSim*
- *FlowSim.Core*
- *FlowSim.Dialogs*
- *FlowSim.GUI*

} GUI

- *FlowSim.Document*
- *FlowSim.Document.Editing*

} Data

- *FlowSim.LB*
- *FlowSim.LB.Core*
- *FlowSim.LB.MC*
- *FlowSim.LB.Qt*
- *FlowSim.LB.Remoting*

} Simulation

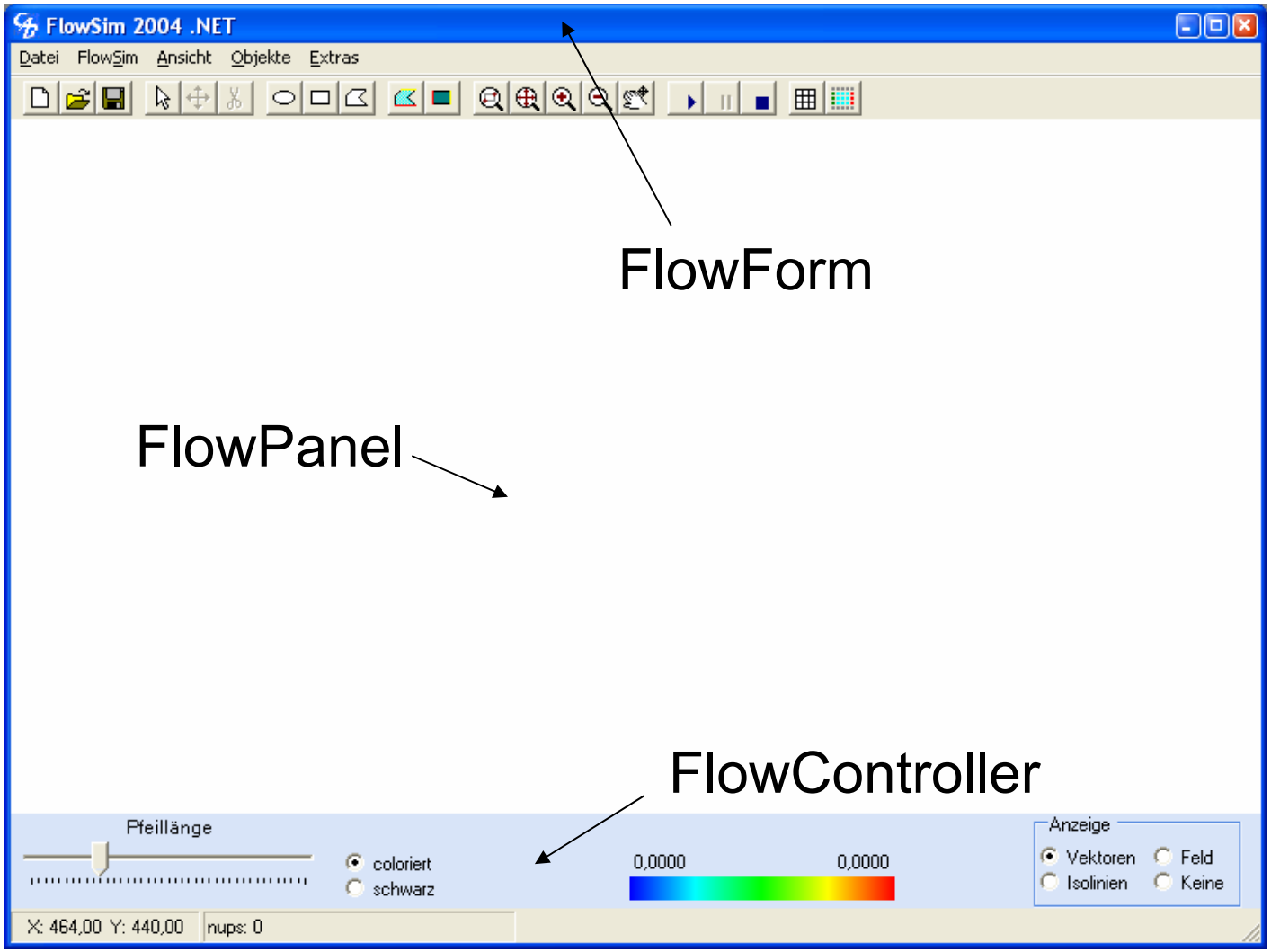
- *FlowSim.Geometries*
- *FlowSim.Geometries.GeoData*
- *FlowSim.Geometries.GeoVisual*

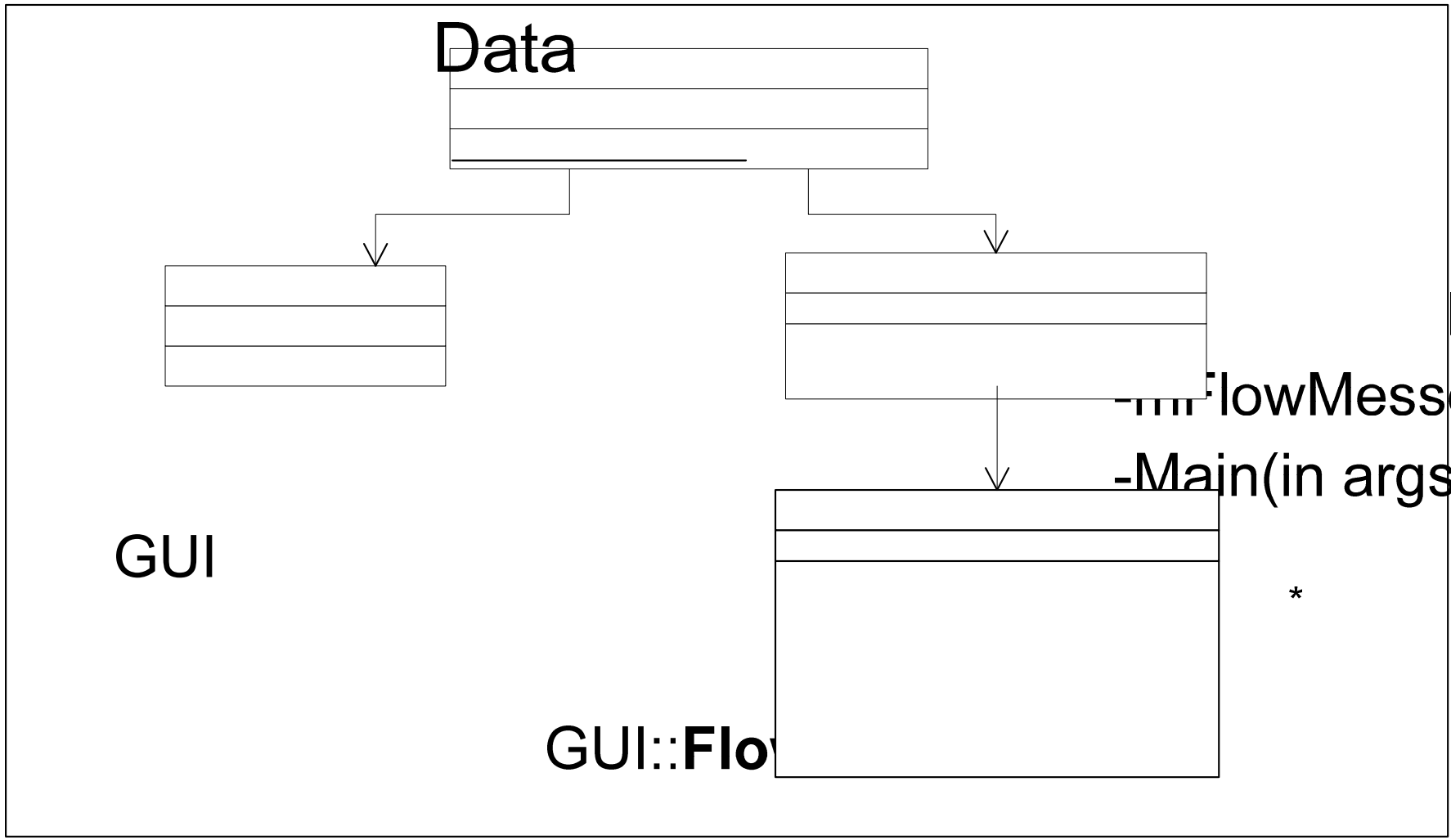
} Elementare Datentypen

# GUI

- *FlowForm*
  - *Einstiegspunkt für die Anwendung*
  - *Menü, Toolbar, Statusleiste*
- *FlowPanel*
  - *Darstellung, Benutzerinteraktion*
  - *Verwendet WorldToPanelTransformer*
- *FlowController*
  - *Steuerung der Simulation*
- *Dialoge*

# GUI

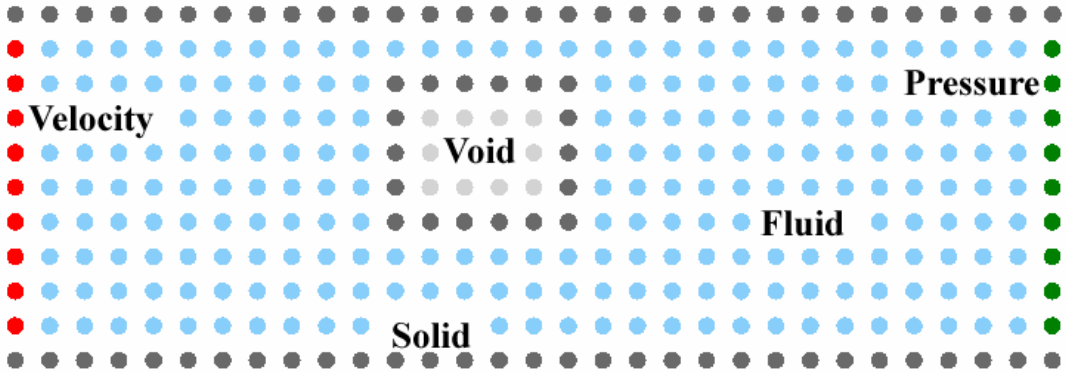

FlowSim 2004 .NET
Datei FlowSim Ansicht Objekte Extras
FlowForm
FlowPanel
FlowController
Pfeillänge
coloriert
schwarz
0,0000
0,0000
Anzeige
Vektoren
Feld
Isolinien
Keine
X: 464,00 Y: 440,00
nups: 0


Data
FlowF
-InitFlowMessenger :
-Main(in args : string
GUI
*
GUI::Flo

# Elementare Datentypen

*Namespace Geometries*

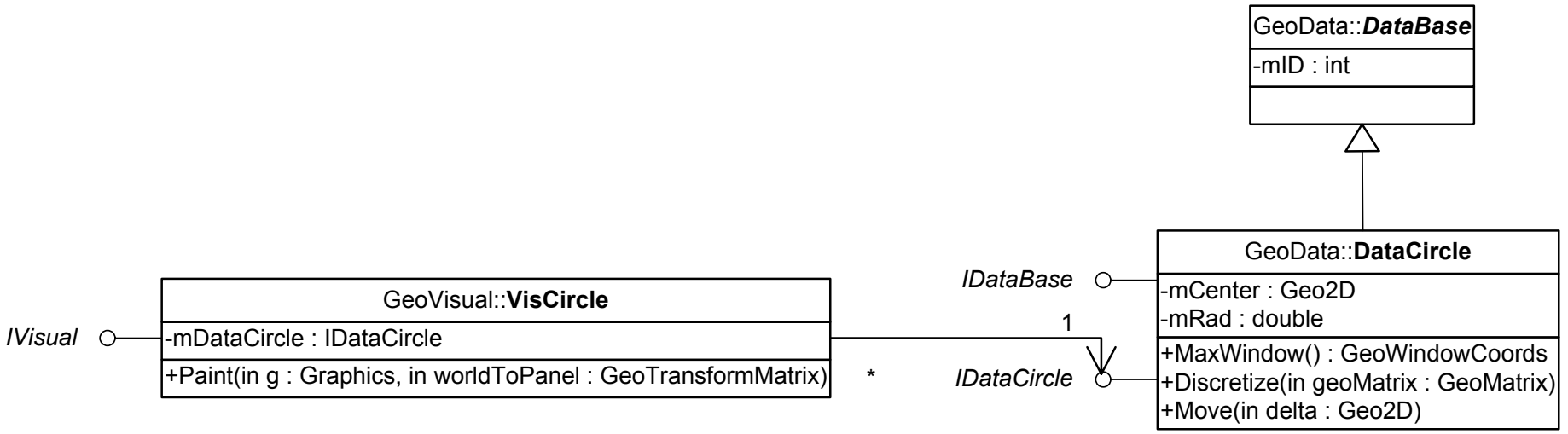
- Punkt - *Geo2D, GeoWindowCoords*
- Affine Abbildungen - *GeoTransformMatrix*
- Geometrieobjekte - *(DataCircle, VisCircle, DataRectangle…)*
- Listen - *DataGeoList, VisGeoList*
- Matrix der Geometrieobjekte - *GeoMatrix*



# Trennung: Visualisierung – Daten

| | | |
|---|---|---|
| VisCircle | - | DataCicle |
| VisRectangle | - | DataRectangle |
| VisPolygon | - | DataPolygon |
| VisGeoList | - | DataGeoList |
| GeoVisMatrix | - | GeoMatrix |

# Trennung: Visualisierung – Daten

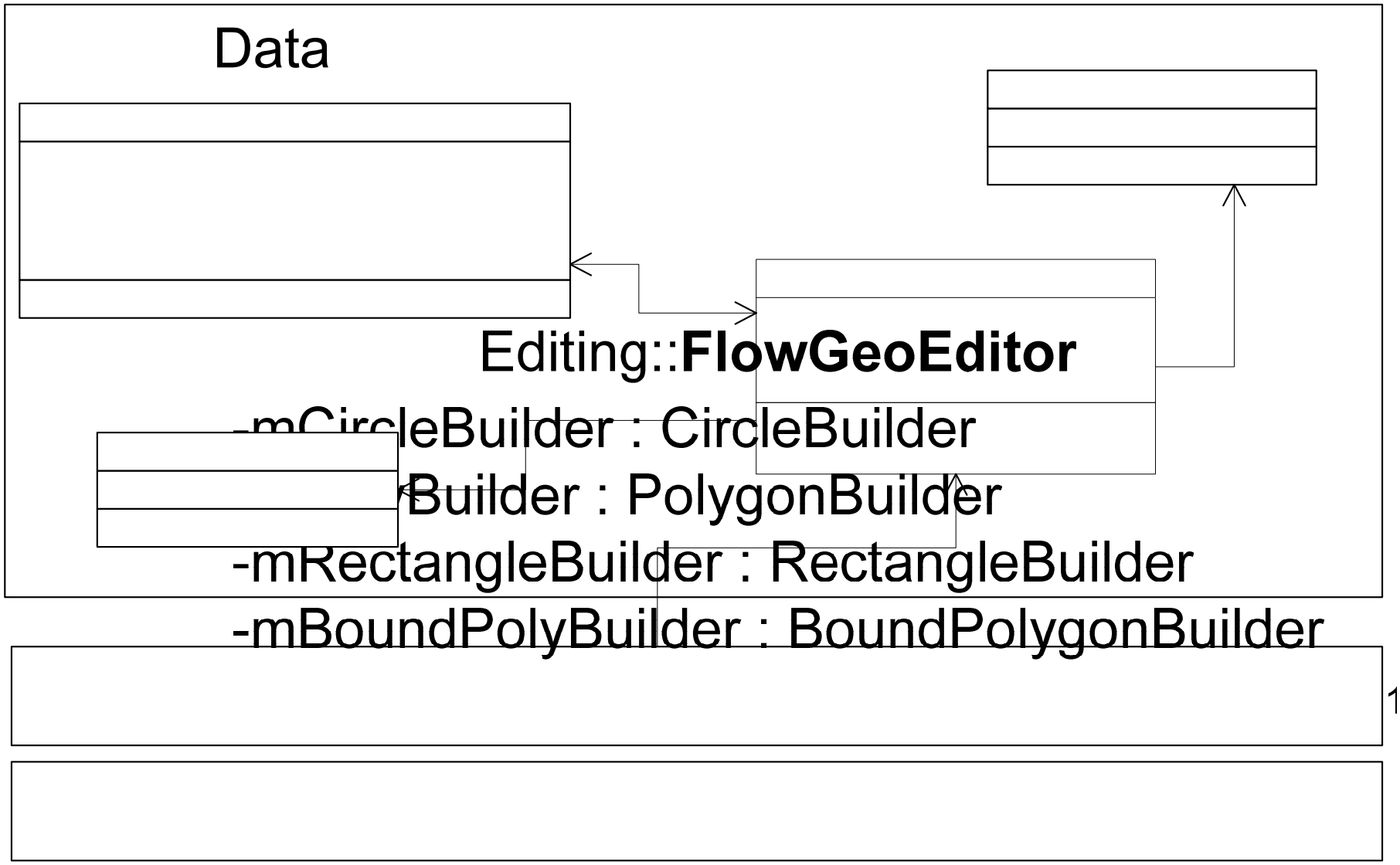


# Data-Layer

## Namespace Document

- *FlowDocument*
  - **Projektdaten**
    - *FlowProject (speichern - öffnen)*
  - **Datenhaltung**
    - *DataGeoList, VisGeoList*
    - *GeoMatrix, GeoVisMatrix*
  - **Datenmodifikation**
    - *FlowGeoEditor (Geo-Objekte modifizieren)*
      - *CircleBuilder*
      - *RectangleBuilder*
      - *PolygonBuilder*
      - *….alle IGeoBuilder*

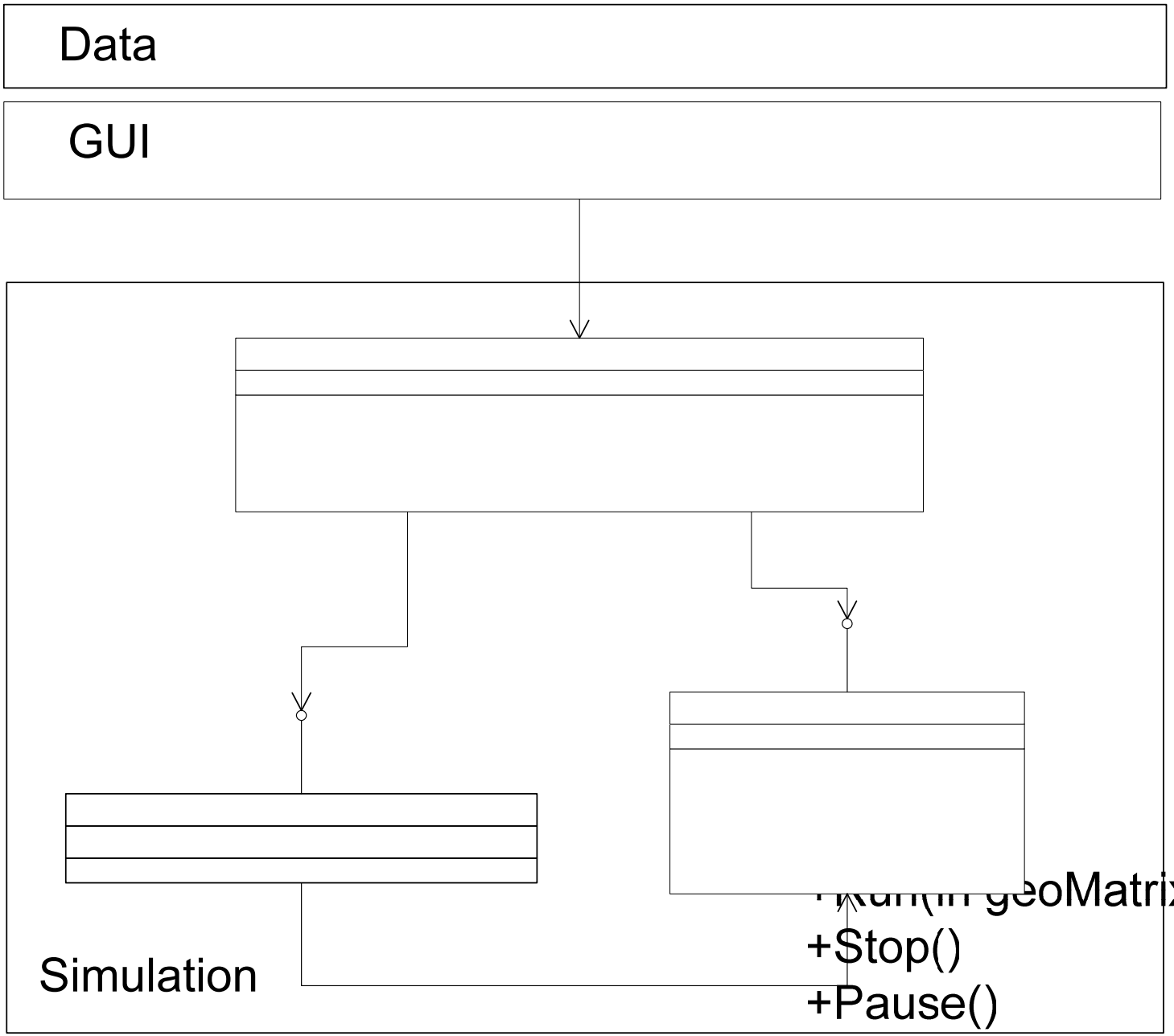
} *(Geo-Objekte erstellen)*


Data
Editing::FlowGeoEditor
-mCircleBuilder : CircleBuilder
-Builder : PolygonBuilder
-mRectangleBuilder : RectangleBuilder
-mBoundPolyBuilder : BoundPolygonBuilder
1

# Simulation-Layer

- *FlowSimulation*
  - *Steuerung der Simulation*
  - *Run(), Stop(), Pause(), Resume(), UpdateMatrices()*
  - *LBComputationXXX*
    - *ILBComputation*
    - *RunLBGK(), RunMRT()*
    - *calculation(), propagate(), collide(),…*
  - *LBVisComputationXXX*
    - *ILBVisComputation*
    - *PaintField()*
    - *PaintVectors()*
    - *…*

} mehrere Varianten




*

1

LB::**Flow**

eoMatrix : GeoMatrix,

*

# Varianten der Berechnungskerne

- *Verschiedene Array Typen*
  - *Arrays hohe Bedeutung im Berechnungskern*
- *Zeiger in .NET*
  - *Können in C# verwendet werden*
- *Interoperabilität*
  - *Zusammenspiel von verwalteter & unverwalteter Welt*

# Arrays

- *Rechteckige Arrays*
  - *int[,,]*
  - *Zusammenhängender Speicherbereich*

  → *1. Berechnungskern*

- *Verschachtelte Arrays*
  - *Int[ ][ ][ ]*
  - *Unzusammenhängende Speicherbereiche*

  → *2. Berechnungskern*

# Zeiger in .NET

- *Interoperabilität & Performancesteigerung*
- *Keine Sicherheitsüberprüfungen durch CLR*
  - *Typsicherheit*
  - *Range Checking*
  - *Schlüsselwort: **„unsafe“***
- *Zeigerarithmetik*
  - *Schlechte Lesbarkeit des Quelltextes*
  - *rho=pf[m_ny*9*i+9*j+0]+pf[m_ny*9*i+9*j+1]+...*
  - *(rho = m_f[i,j,0]+m_f[i,j,1]+m_f[i,j,2]+…)*

→ *3. Berechnungskern*

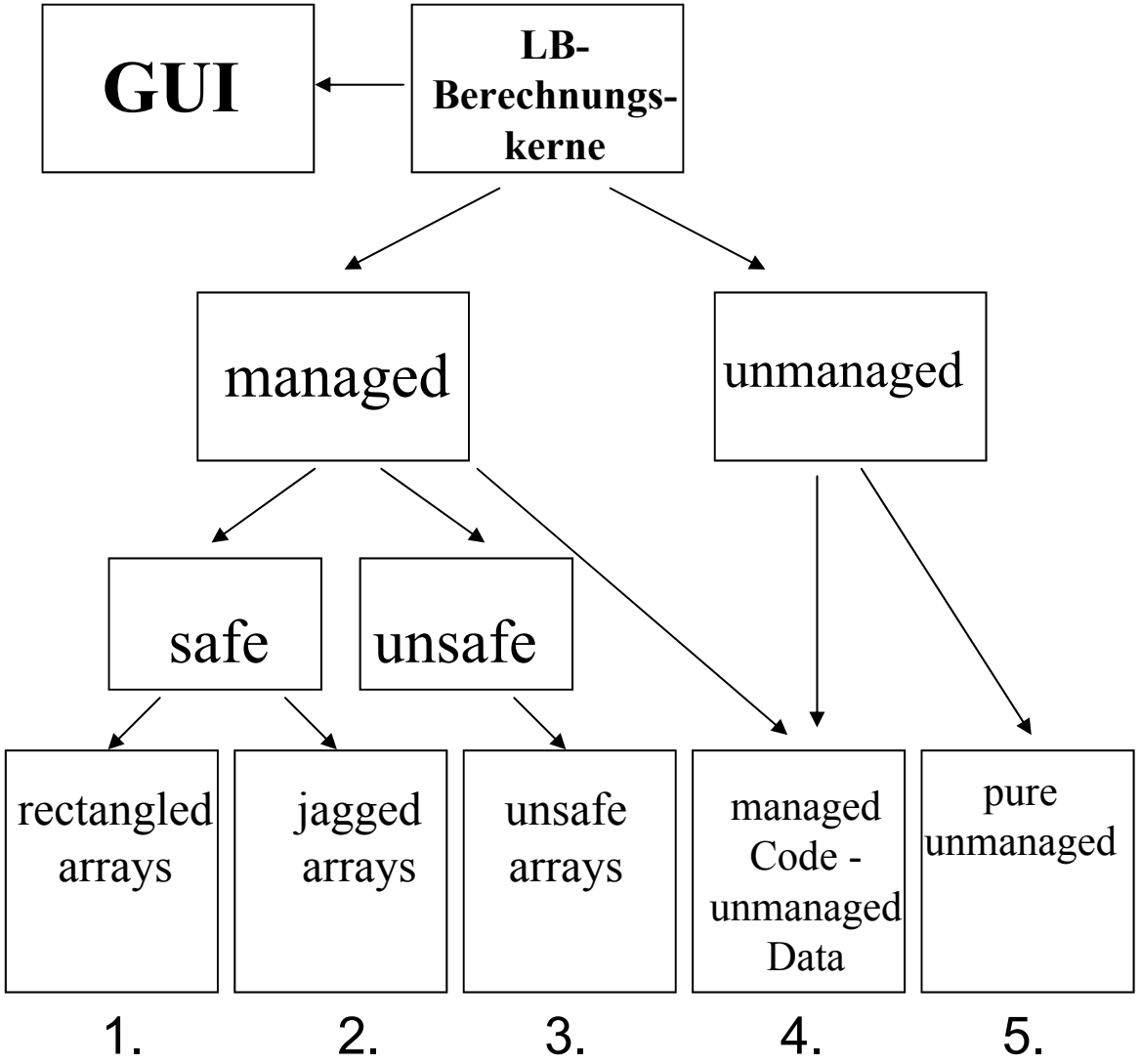
# Interoperabilität

- *C++ Managed Extensions*
  - *Erweiterung der Sprache C++*
- *„Mischen“ von managed & unmanaged Code*
- *Unverwaltete Daten & Verwalteter Code*

  → *4. Berechnungskern*
- *Rein Unverwalteter Code*

  → *5. Berechnungskern*

# Berechnungskerne



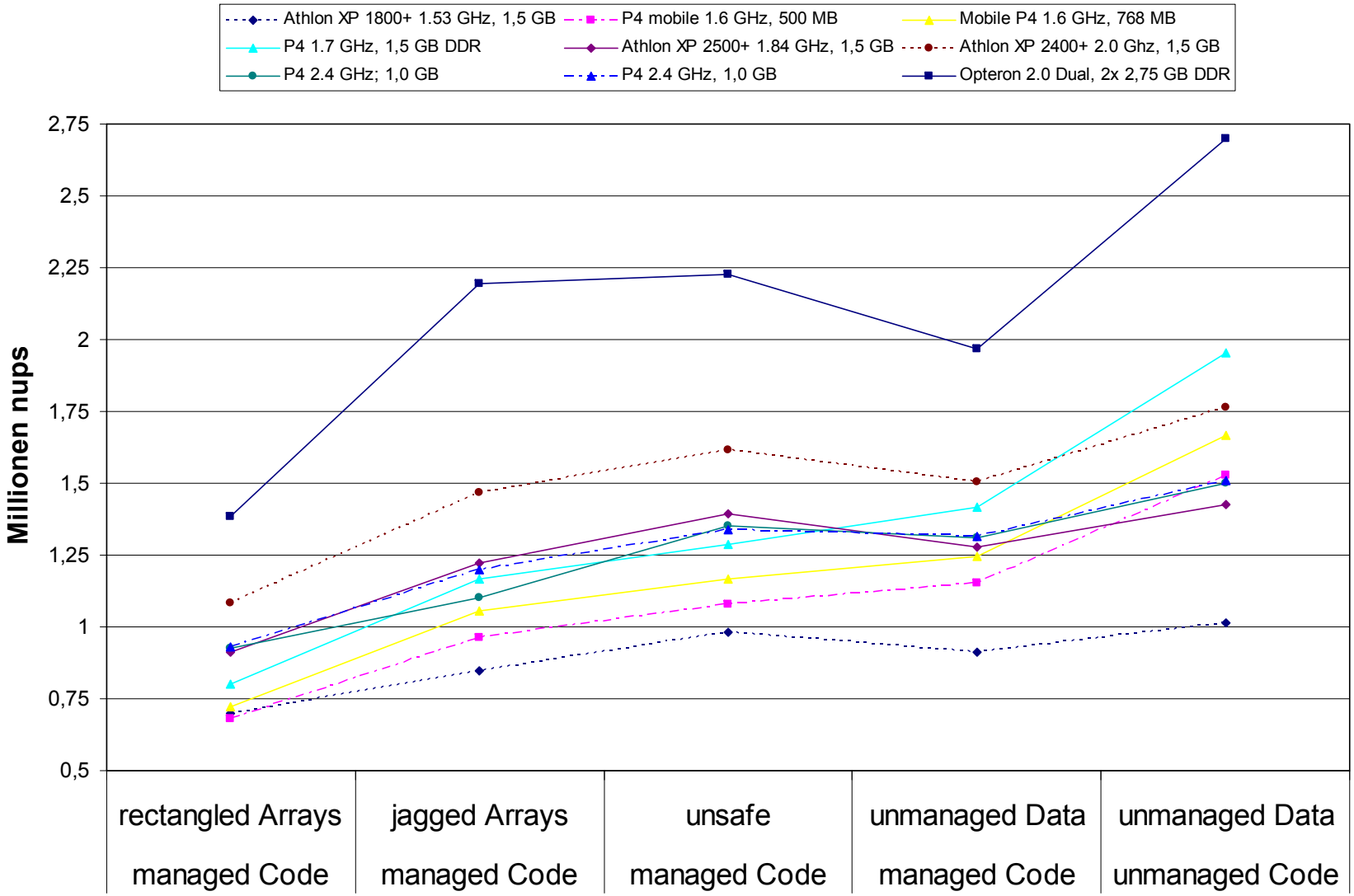
# Berechnungkerne

- *regulärer Managed Code*
  1. *Rectangled Arrays*
  2. *Jagged Arrays*
- *Managed Code + Zeiger*
  3. *Unsafe Managed Code*
  4. *Managed C++*
- *Nativer Code*
  5. *Unmanaged Code*

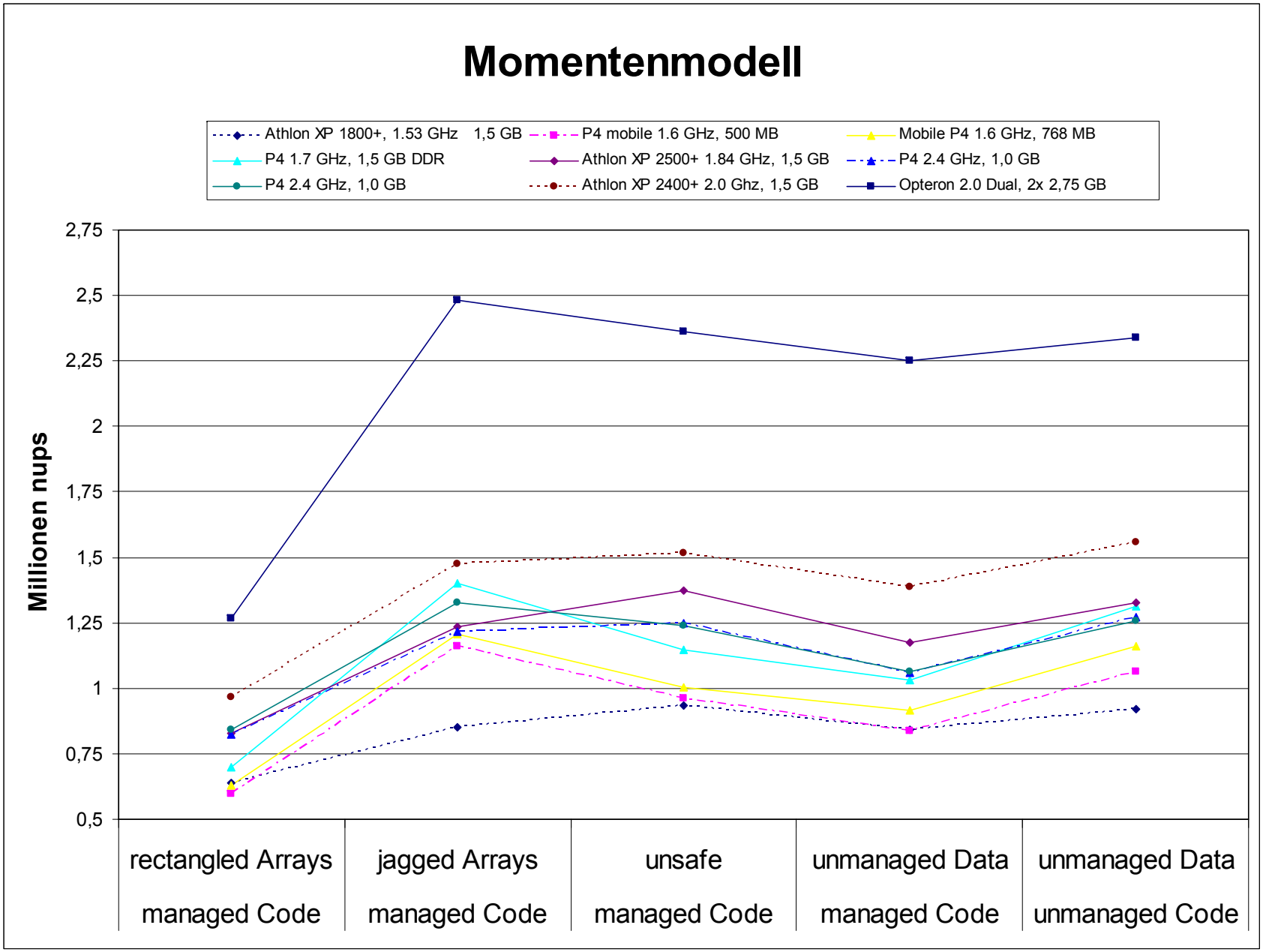
# Testläufe

- *5 Berechnungskerne*
- *Strömungsgebiet mit 30.000 Knoten*
- *9 Rechner*
- *LBGK & Momentenmodell*

# LBGK-Methode



# Momentenmodell



# Ergebnisse

- *Jagged Arrays zu bevorzugen*
  - *CLR kann Range Checking unterdrücken*
- *Unmanaged Code i.A. gute Ergebnisse*
- *u.U. Managed Code sehr gute Ergebnisse*
- *Es gibt nicht DIE Performanceoptimierung*
- *Unsafe Code unverhältnismäßig viel Aufwand*
- *Managed C++ sehr flexibel*

# FlowSim 2004 .NET

*2. Vorführung*

Danke für Ihre Aufmerksamkeit.

Fragen?