

Objektorientierte Entwicklung von Domino Anwendungen

Dipl. Ing. (FH) Manfred Meise

IBM Certified Advanced Developer - Lotus Notes and Domino R3 – R8.5

IBM Certified Advanced Administrator - Lotus Notes and Domino R3 – R8, R8.5

IBM Certified Advanced Instructor – Lotus Notes and Domino R3- R8, R8.5



zu meiner Person: Manfred Meise

- Studium Elektrotechnik (Dipl. Ing.)
- Arbeit als Softwareingenieur seit mehr als 30 Jahren bei verschiedenen Computerherstellern und Softwarehäusern
- Gründer und Geschäftsführer der mmi consult gmbh
- Erfahrungen mit Lotus Notes/Domino seit 1992 - Markteinführung in Europa
(als Leiter Strategische Projekte bei Lotus Development Deutschland)
- IBM Zertifizierungen als Anwendungsentwickler, Systemadministrator, Trainer für die Produktversionen R3 bis R8.5
- Tätigkeitsschwerpunkte im Entwicklungsbereich:
CRM, Workflow, Objektorientierte Anwendungsarchitekturen, XPages
- Tätigkeitsschwerpunkte als Systemadministrator:
Domänenzusammenführungen und -trennungen, Betriebshandbücher und Administrationsstandards, Versionswechsel, Infrastruktur-Audits, Client-Rollouts
- Erreichbar unter:
 - **manfred.meise@mmi-consult.de**
 - **<http://www.mmi-consult.de>**
 - **<http://www.mmi-consult.de/faq>**



Meine Themen heute ...

Entwicklung von Domino Anwendungen – gestern heute und morgen

Einführung in UML

Einsatz von CASE – Tools auch für die Domino Entwicklung

Erstellung der Systemdokumentation

Domino Designer 8.5.1 – Erweiterungen für Objektorientierung

Testverfahren und -tools für objektorientierte Anwendungen

Resumee und Ausblick



Entwicklung von Domino Anwendungen – gestern und heute

- Viele unerfahrene Entwickler fahren noch lange Strecken auf dem Rapid Development Zug
- Linearer Code kann sehr unübersichtlich werden
- Funktionen in verschiedenen Skriptbibliotheken lassen sich nicht unkompliziert nutzen, da Bibliotheken Abhängigkeiten aufweisen, die einer Kombination widersprechen
- Komplexere Aufgabenstellungen führen oftmals zur Code, der auf globale Variablen arbeitet
 - ▶ welcher fehlerhafte Code hat schließlich die Variableninhalte verändert?
 - ▶ warum ist die Variable nicht initialisiert?
 - ▶ leider kann der Lotus Debugger keine Variableninhalte überwachen
- Dokumentation → oft Mangelware
- Lösungsbausteine oft wenig strukturiert → Spaghetti-Code
- Änderungsfreundlichkeit → Katastrophal



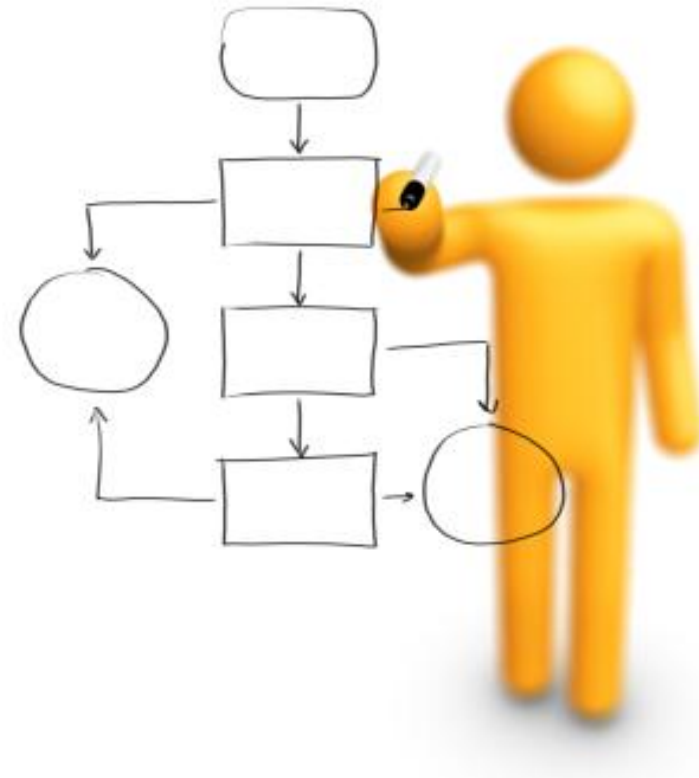
Entwicklung von Domino Anwendungen – gestern und heute

- Objektorientierte Codierung adressiert viele Schwachstellen
 - ▶ gekapselte Daten
 - ▶ klare Schnittstellen (Properties und Methoden), über die Werte gelesen oder gesetzt werden (man kann Breakpoints im Debugger setzen)
 - ▶ Entstehung eines modularen Baukastens, dessen Bausteine kombiniert werden können
 - ▶ Überlagerung von Methoden erlaubt es, mit abgeleiteten Klassen eine angepasste/erweiterte Funktionalität zu implementieren
 - ▶ einfache Fehlersuche und Wartung, weil Code weniger komplex ist. Fehlerkorrekturen einmal durchführen und überall Fehler beseitigen
 - ▶ robuster Code, da Klassen stets wiederverwendet werden
 - ▶ wiederverwendbarer Code
- Leider muss man etwas planen und konzipieren
- Entwicklerdokumentation noch wichtiger, da Code universell einsetzbar sein könnte/sollte
- Code weniger effizient, da unnützer Overhead stets abgearbeitet wird



Entwicklung von Domino Anwendungen – heute und morgen

- Vorteile des Rapid Prototyping primär für UI Entwürfe verwenden
- Anwendungscode (LotusScript, Serverside JavaScript, Java) objektorientiert planen und implementieren
- Auf Wiederverwendbarkeit und Wartbarkeit sowie implizite Dokumentation Wert legen
- Eingesetzte Methoden und Werkzeuge sollte vergleichbar denen der klassischer Anwendungsentwicklung sein
- Klare Beschreibung der Anforderungen führen zu einem Top-Down Ansatz mit
 - ▶ nachvollziehbaren Systemmerkmalen
 - ▶ Testplanung als Abfallprodukt
 - ▶ Versionierung von Modulen
- Gemeinsame „Sprache“ für alle Fachabteilungen, Projektmanager, Entwickler finden: UML



EINFÜHRUNG IN UML



Was ist UML ?

- Standard Modellierungssprache
- Dient zur Visualisierung
- Spezifizierung
- Konstruktion
- Dokumentation von Modellen für Softwaresysteme

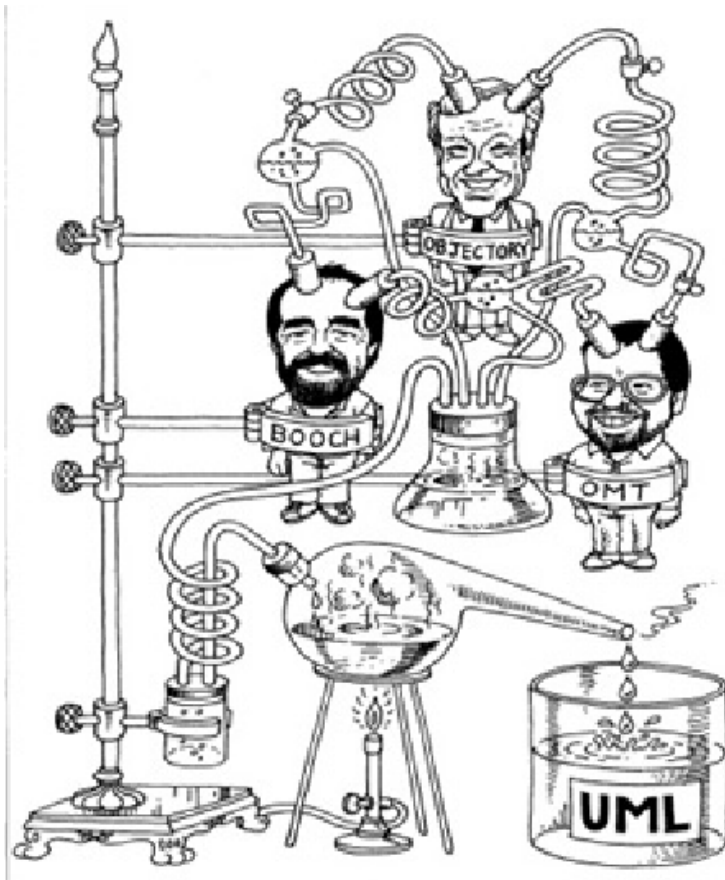


Geschichte der UML

- Erste Publikationen vor ca. 30 Jahren.
- IT-Gurus entwickeln verschiedene, meist nur auf einen Anwendungsbereich spezialisierte Methoden. (80er Jahre)
- Erste Bücher über objektorientierte Analyse – und Designmethoden seit Anfang der 90er Jahren.



3 Amigos (Godfather's of UML)



Grady Booch

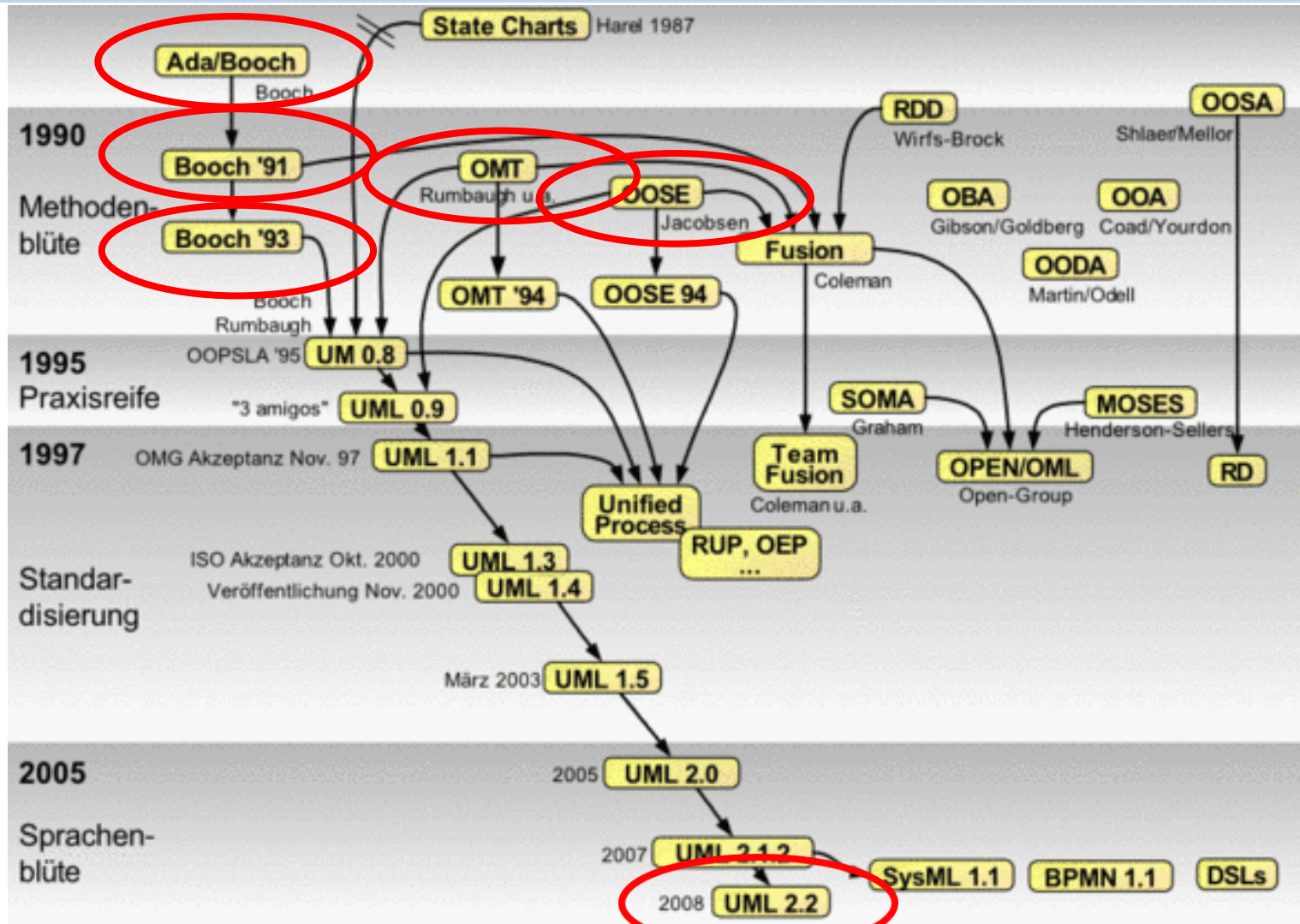
James Rumbaugh

Ivar Jacobson



Quelle Grafik: „Introduction to the Unified Modeling Language, Terry Quatrani

Historische Entwicklung objektorientierter Methoden und der UML



Quelle: http://de.wikipedia.org/wiki/Unified_Modeling_Language

UML = Sprache zur Beschreibung von Softwaresystemen

- Grundgedanke: Einheitliche Notation für alle Softwaresysteme!
 - ▶ UML entstand aus mehreren bestehenden Notationen

- Verschiedene Diagrammtypen, die sich gegenseitig ergänzen können (sollen!) und verschiedene Systemaspekte hervorheben
 - Bsp: Analogie Bauplan für Haus – Grundriss, Aussenansichten, Werkpläne für versch. Handwerker...



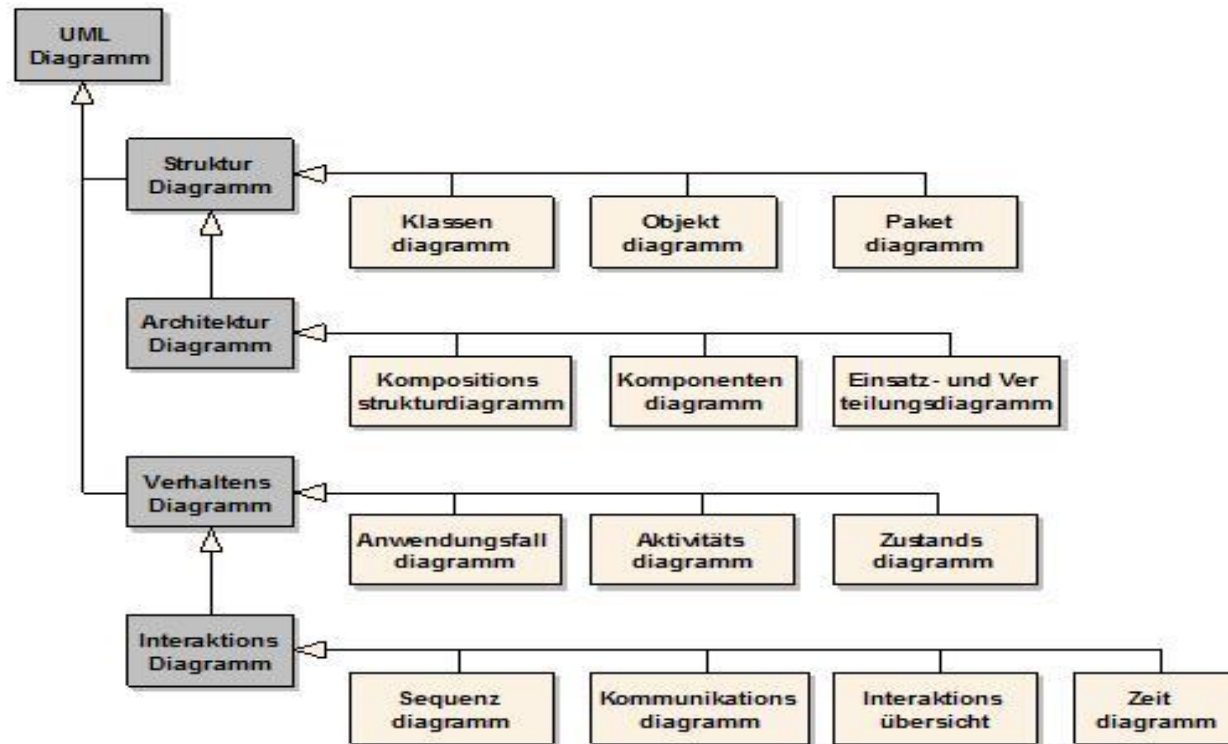
UML für was?

- UML ist ein Werkzeug für die Systemanalyse und beim Design
- abstrakte Beschreibungssprache
 - ▶ ermöglicht Kommunikation zwischen Entwicklern und Benutzern
- etabliertes Hilfsmittel bei OO-Analyse und -Design, sowie auch bei der Dokumentation
 - ▶ Unterstützung durch diverse Softwarewerkzeugen



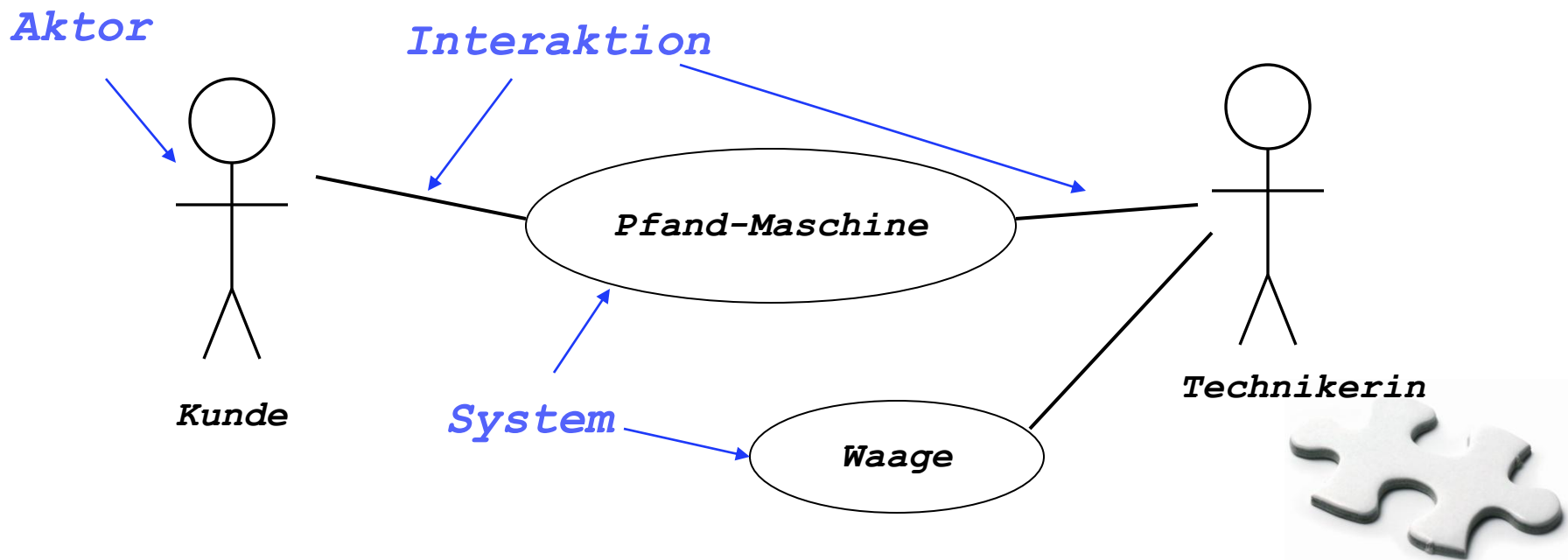
UML 2.1 – Diagrammtypen

- UML ≠ Klassendiagramme
 - Klassendiagramme sind nur ein Teil von UML
 - UML ist mehr!



Beispiel: Use-Case-Diagramm

- Beschreiben das Zusammenwirken von Personen [allg.: Aktoren] mit einem System
- Einsatz: Anforderungen, Festlegung, Übergabe



Klassendiagramme

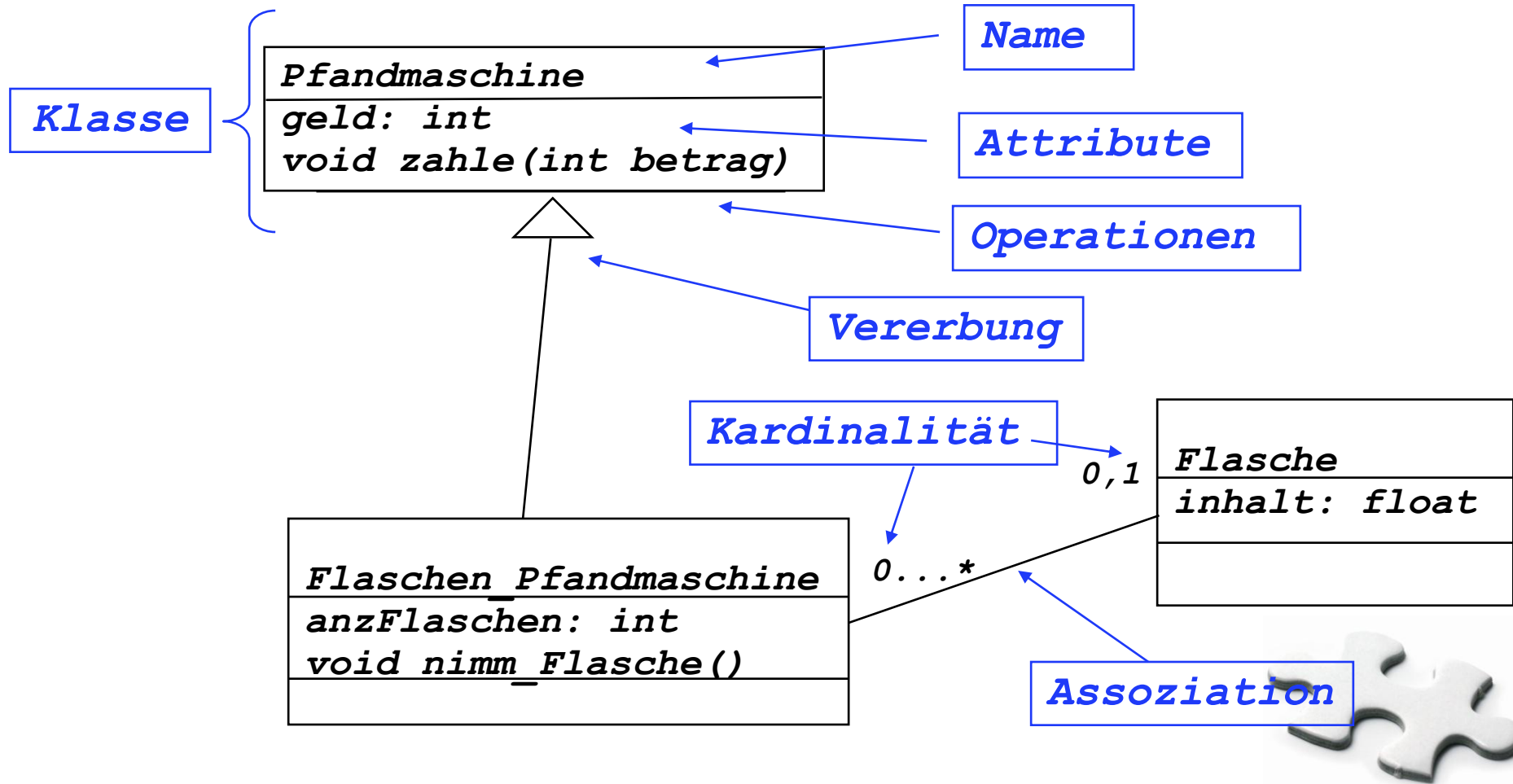
- Beschreiben **statische Struktur** der Objekte und ihre **Beziehungen** untereinander

- Ermittlung der Klassen ist nicht Sache der Klassendiagramme!
 - ▶ anderer Hilfsmittel z.B.:
 - CRC Cards (Class, Responsibility and Collaboration)
 - Use Cases

- Zentraler Bestandteil der UML
 - ▶ aber nicht einziger!



Beispiel: Klassendiagramm



Beispiel: Interaktionsdiagramm

- Beschreiben **zeitliche** Abläufe (Aufrufsequenzen) zwischen (bekannten) Objekten
 - ▶ Dynamische Sicht im Gegensatz zu statischen Klassendiagrammen

- Zwei semantisch äquivalente Darstellungen:
 - ▶ Einfach andere Darstellen, aber dieselbe Sicht

Sequenzdiagramme

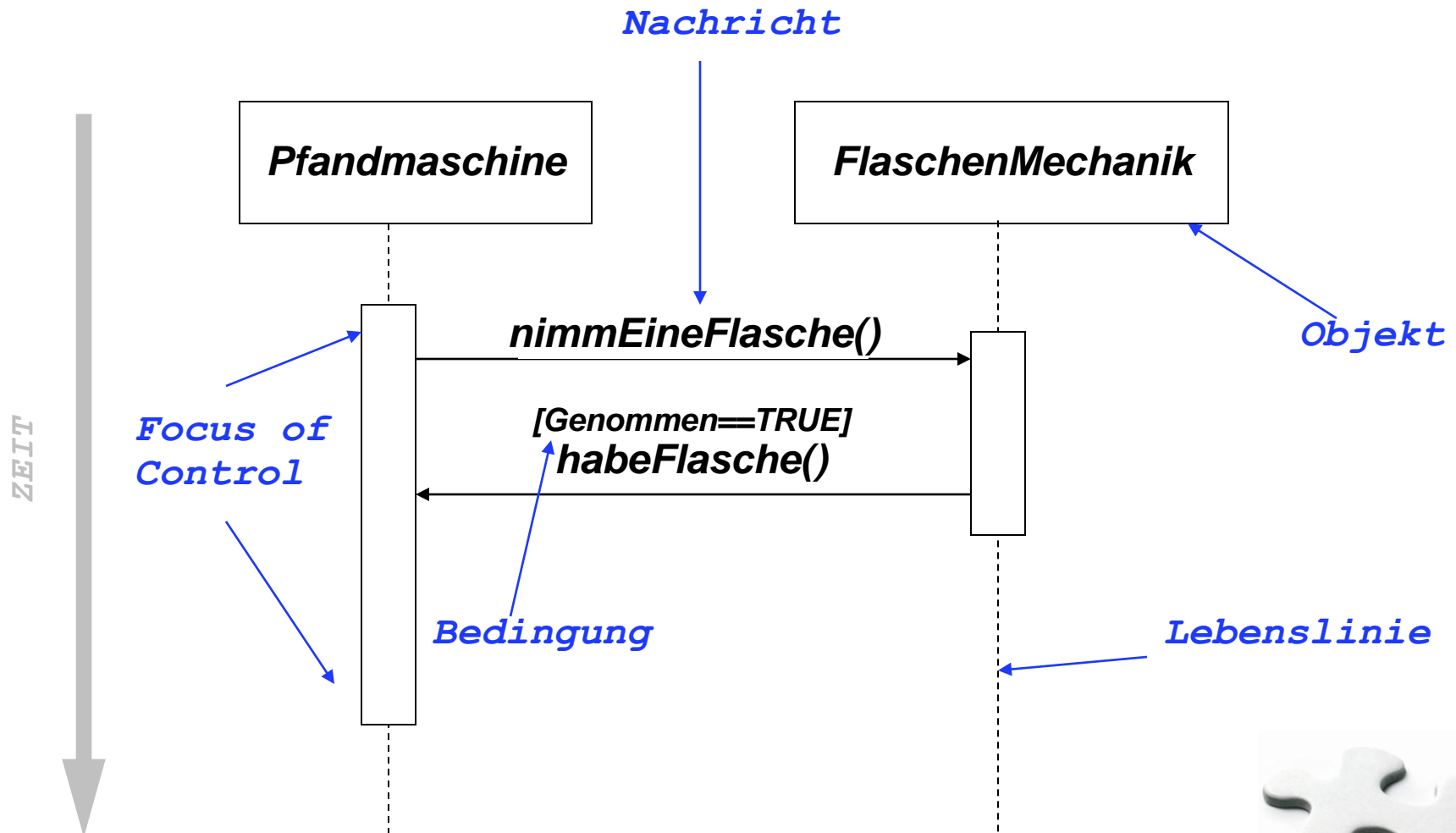
- Verwendung bei wenigen Klassen
- Zeitablauf klar ersichtlich

Kollaborationsdiagramme

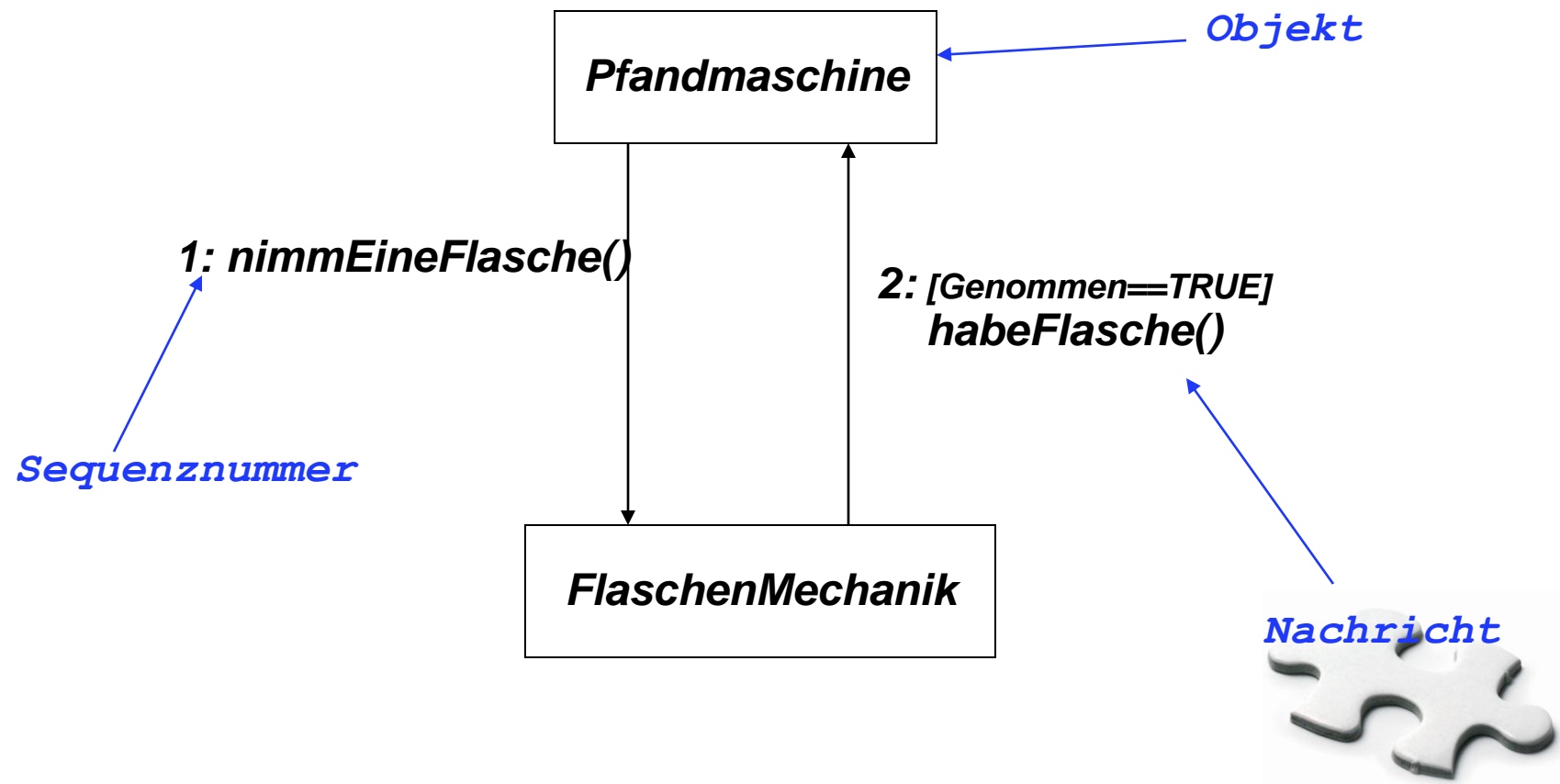
- Verwendung bei wenigen Nachrichten
- Zeitablauf weniger klar ersichtlich



Beispiel: Sequenzdiagramm

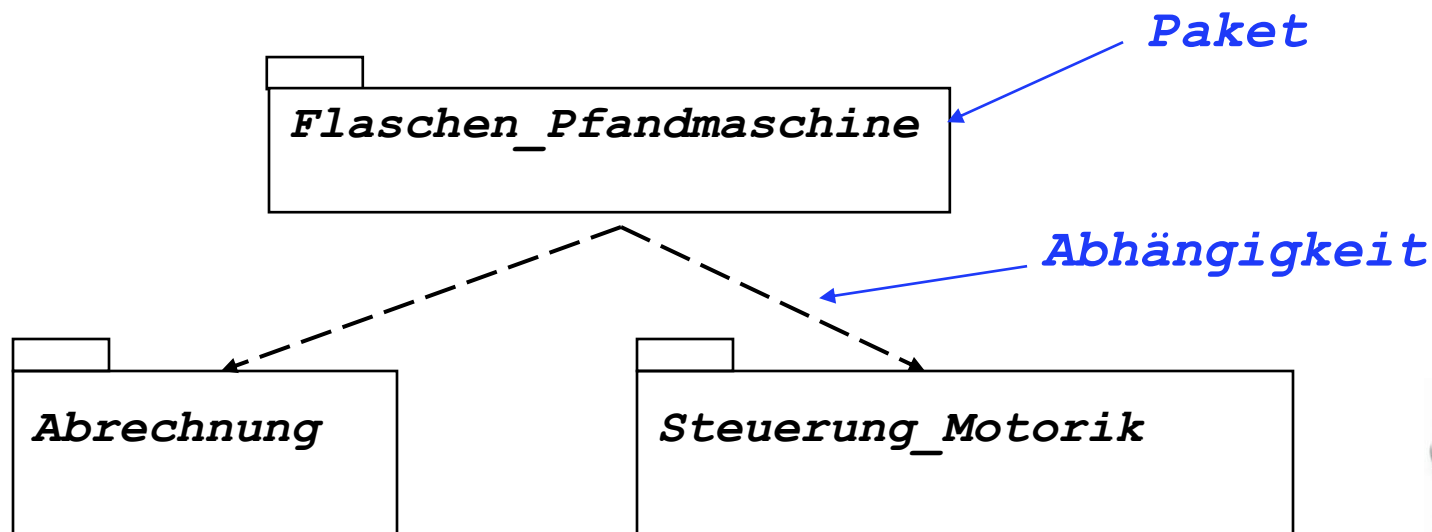


Beispiel: Kollaborationsdiagramm



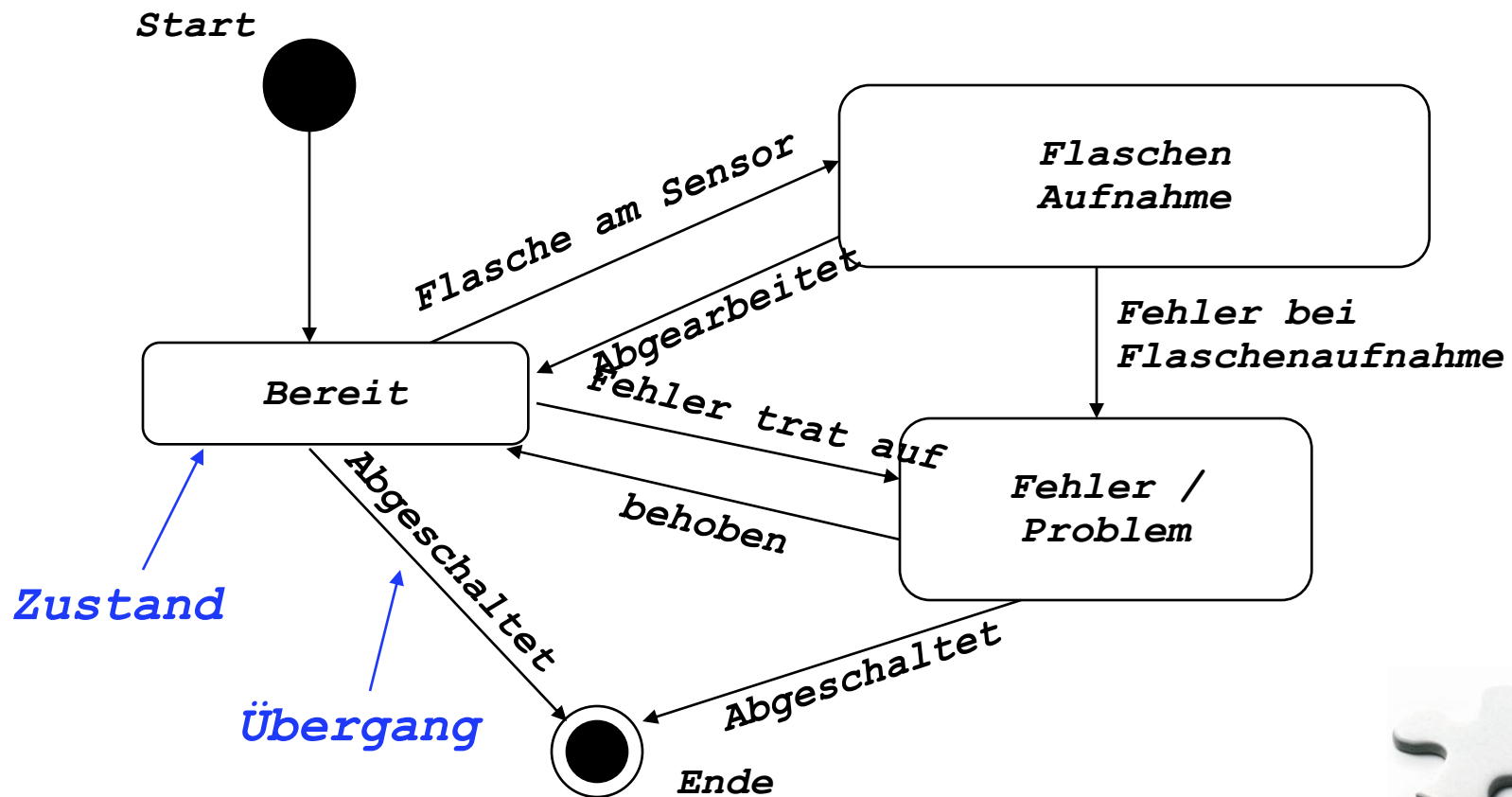
Beispiel: Package-Diagramm

- Strukturierung der verschiedenen Darstellung
- Zusammenfassung von Gruppen von Diagrammen oder Elementen
 - ▶ Zusammenfassung: Strukturell oder Thematisch
 - ▶ Besserer Gesamtüberblick



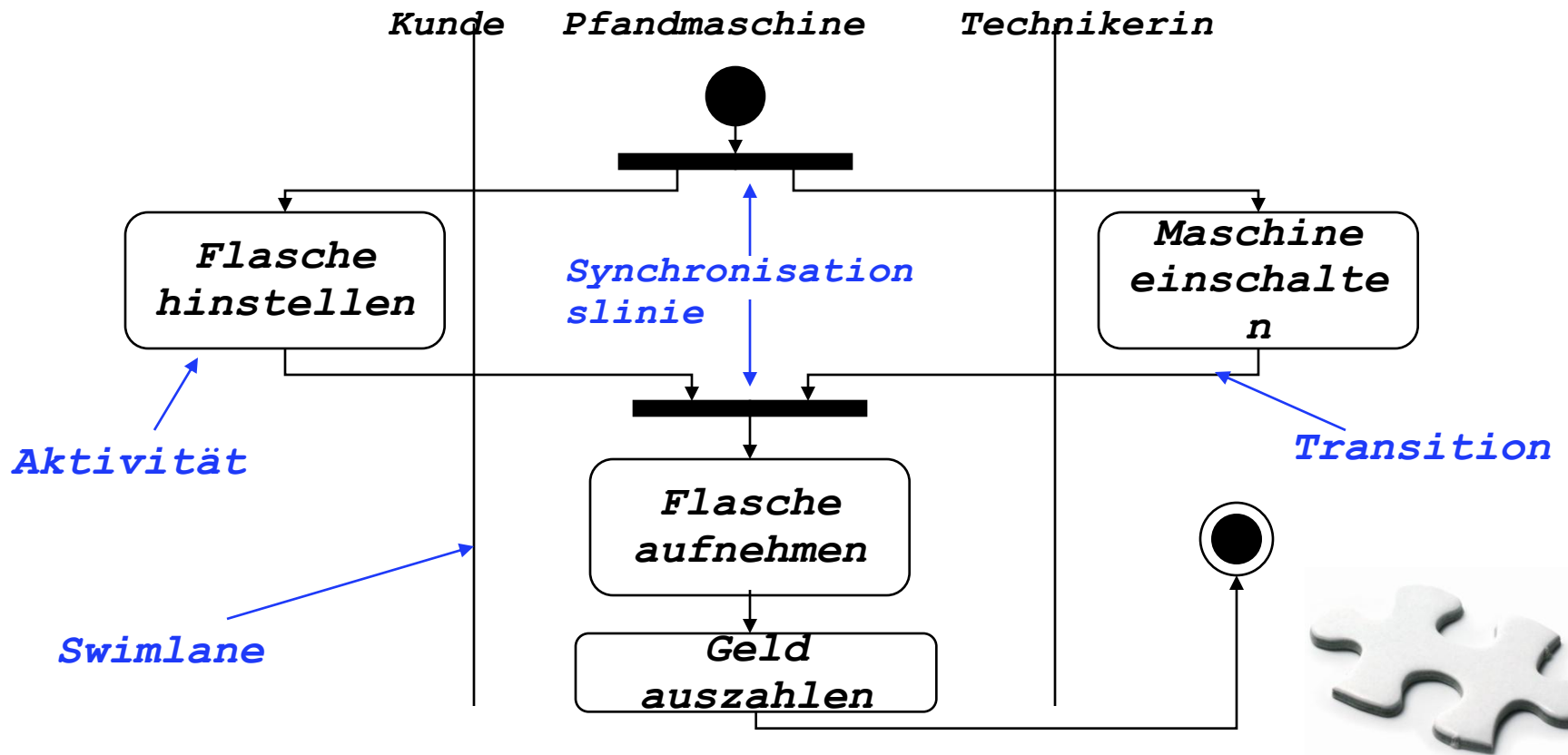
Beispiel: Zustandsdiagramm

- Beschreiben das **Verhalten** eines (Teil-)Systems



Beispiel: Aktivitätsdiagramm

- Beschreiben **nebenläufiges** Verhalten
 - Grundlagen: Zustandsdiagramme, Flussdiagramme & Petrinetze



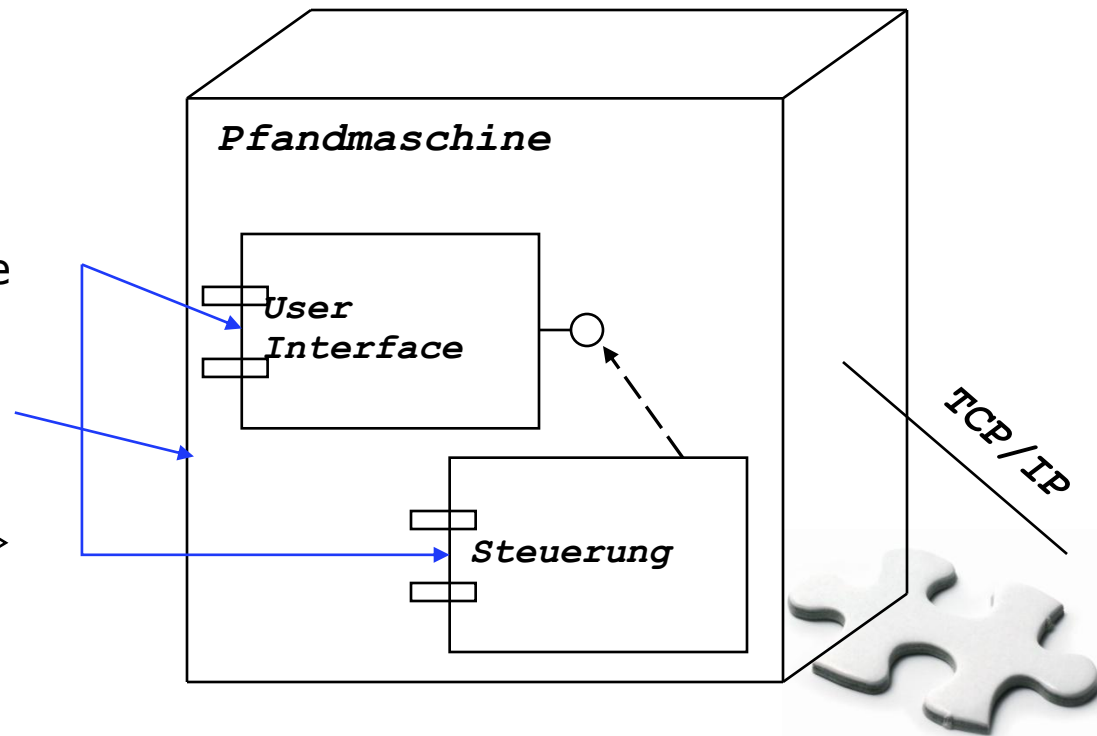
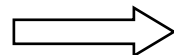
Beispiel: Implementierungsdiagramm

- Beschreiben Aufteilung von logischen Komponenten auf physikalische Komponenten
 - ▶ Einsatz: Aufzeigen von Unterschieden der logischen zur physikalischen Struktur

- Zwei Formen:

- ▶ Komponentendiagramme
- ▶ Deploymentdiagramme

- Kombinierbar!



Zusammenfassung

UML = Sprache zur Beschreibung von Softwaresystemen

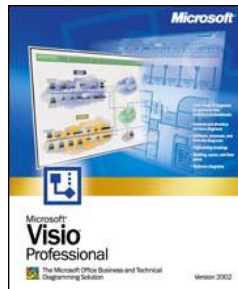
- Verschiedene Diagrammtypen, die sich gegenseitig ergänzen können (sollen!) und verschiedene Systemaspekte hervorheben
- UML ist ein Werkzeug für die Systemanalyse und beim Design
- abstrakte Beschreibungssprache
 - ▶ ermöglicht Kommunikation zwischen Entwicklern und Benutzern



EINSATZ EINES CASE – TOOLS AUCH FÜR DIE DOMINO ENTWICKLUNG



Einige Tools



Rational. software



Erweiterte Übersicht:

<http://www.oose.de/service/uml-werkzeuge.html>

Anbieter: IBM Corporation

Rational. software

- Umfangreiche Produktfamilie
 - ▶ Nahezu alle Aspekte des Systementwurfes, der Softwareentwicklung, des Systemtests, der Geschäftsprozeß- und Datenmodellierung
 - ▶ Größte Angebotspalette am Markt

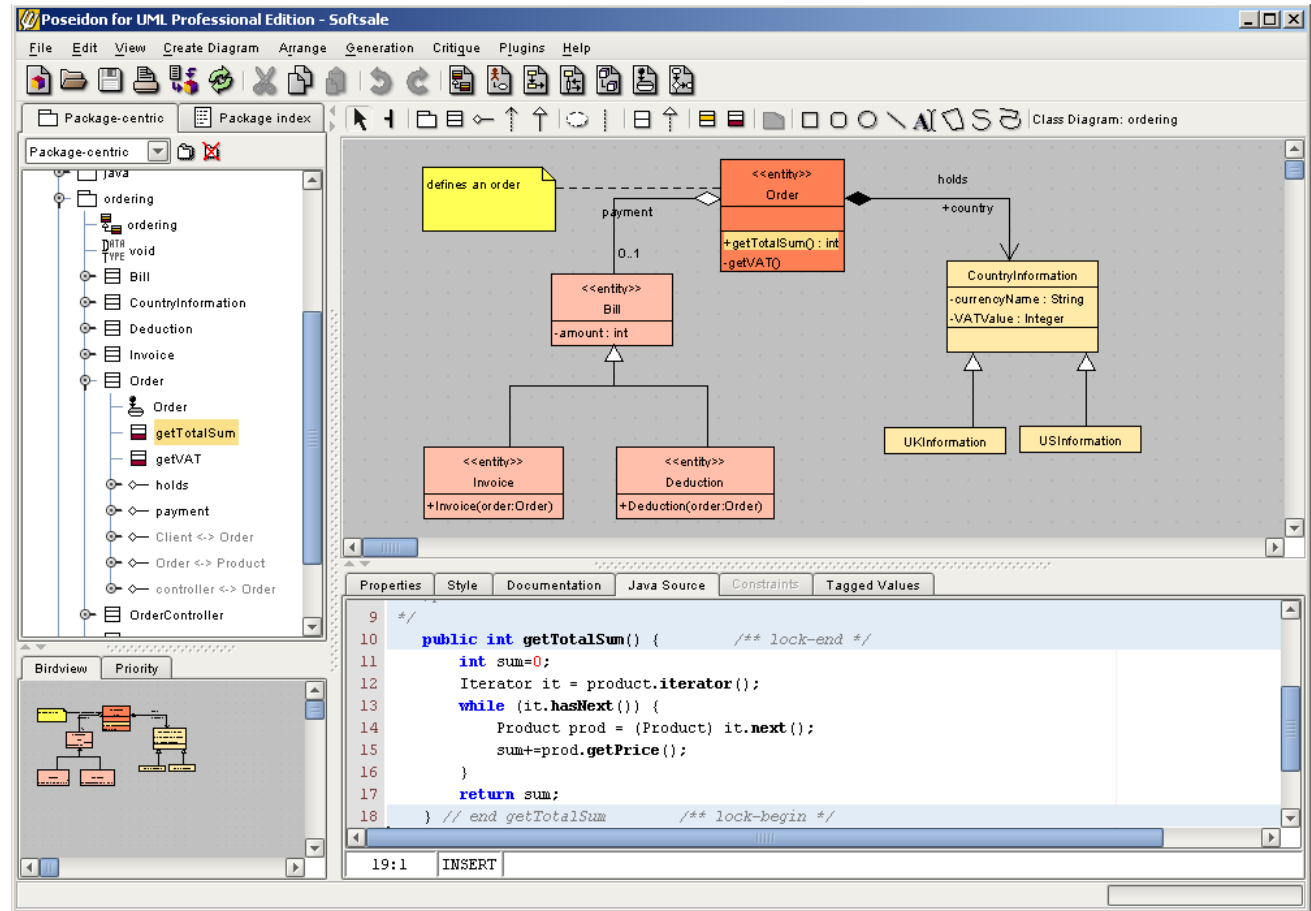
- Profil-Tools, die ihren Preis und Funktionsumfang haben

- Umfangreiche Systemvoraussetzungen

Weitere Informationen:
<http://www.ibm.com/Software/Rational>



Anbieter: Gentleware



Quelle Grafik: Screenshot aus der Professional Edition,
<http://www.gentleware.com>



Poseidon for UML

Professional Edition 1.6

- deutscher Toolanbieter
- übersichtlich und einfach
- zahlreiche Features
- Preis: von Freeware bis 700\$ (Professional Edition)



Anbieter: microTOOL

 objectiF®

- führender deutscher Toolanbieter
- professionelles Werkzeug (teuer 2500 €)
- mittlere und grosse Projekte



..mein Favorit

- Anbieter „SparxSystems“ aus Australien
 - ▶ Deutschsprachiger Raum wird von der Niederlassung in Wien betreut (inkl. Schulungen / Konferenzen etc.)
- Preisgünstige Varianten (nach Bedarf) lizenzierbar
- 4 Produkttypen
 - ▶ Standalone Product
 - ▶ Eclipse Integration
 - ▶ Visual Studio Integration
 - ▶ Kostenloser Viewer für alle Diagramme
- 3 Edition
 - ▶ Desktop
 - ▶ Professional
 - ▶ Corporate
- Deckt alle Phasen des Software-Engineerings ab
 - ▶ MDG mit Code-ReEngineering für Kernsprachen
- Erlaubt es CodeTemplates für nicht unterstützte Sprachen zu erstellen (z.B. LotusScript)
- Preisgünstig (-wert) von 135\$ - 699\$



Design Patterns (Entwurfsmuster)

- Wiederverwendbare Vorlage zur Problemlösung
- Langjährige Sammlung von „Best Practices“ ausgehend von GoF (Gang of Four) und Anderer

<http://de.wikipedia.org/wiki/Entwurfsmuster>

Erzeugende Muster

- Abstract Factory (Abstrakte Fabrik)
- Builder (Erbauer)
- Factory Method (Fabrikmethode)
- Prototype (Prototyp)
- Singleton (Einzelstück)

Strukturelle Muster

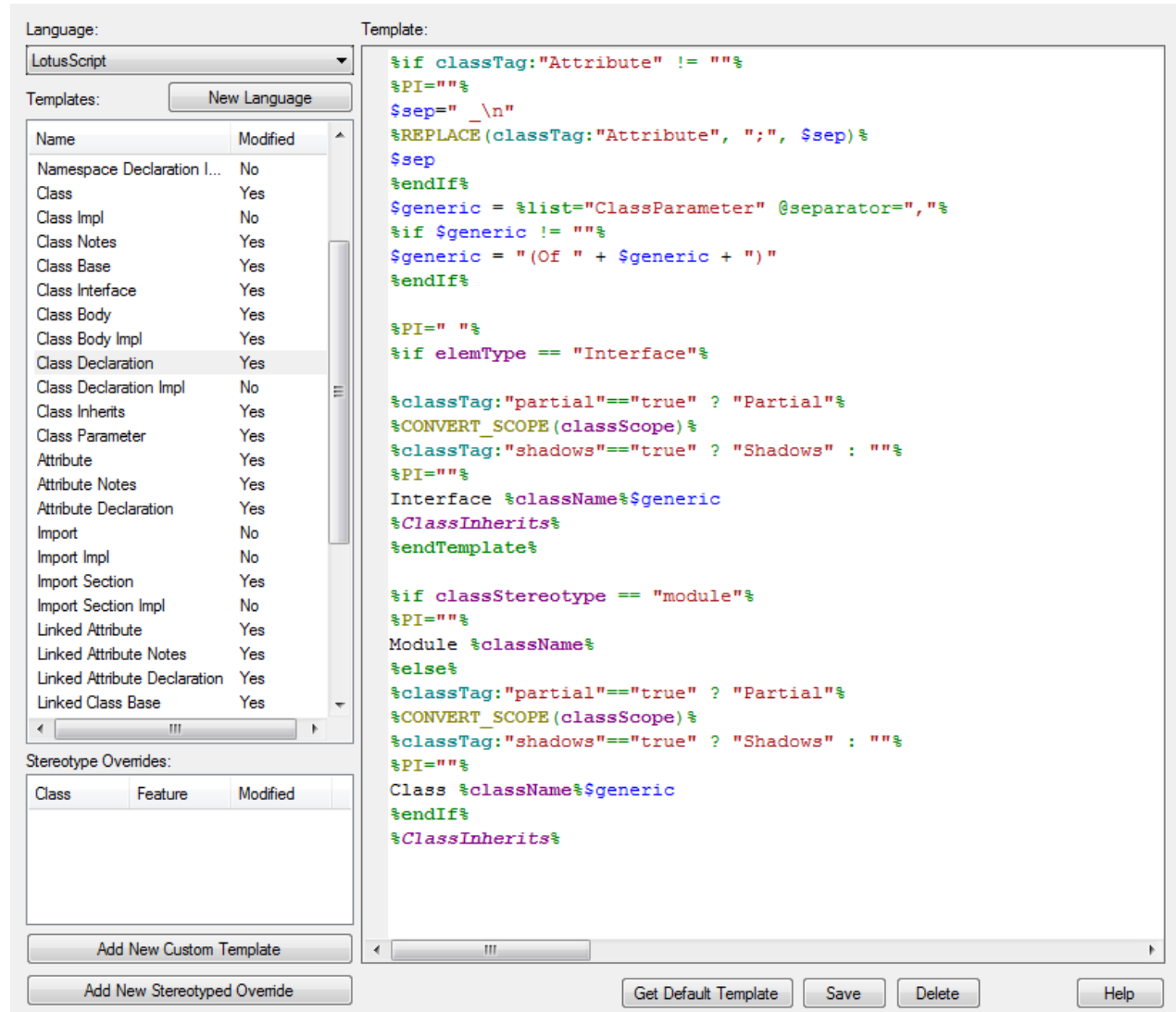
- Adapter
- Composite (Kompositum)
- Bridge (Brücke)
- Decorator (Dekorierer)
- Facade (Fassade)
- Flyweight (Fliegengewicht)
- Proxy (Stellvertreter)

Strukturelle Muster

- Chain of Responsibility (Zuständigkeitskette)
- Command (Kommando)
- Interpreter
- Iterator
- Mediator (Vermittler)
- Memento
- Observer (Beobachter)
- State (Zustand)
- Strategy (Strategie)
- Template Method (Schablonenmethode)
- Visitor (Besucher)

Code Generation Templates

- Definition von Code und Diagrammelementen
- Platzhalter und konditionale Generierung
- Templates für Standardsprachen enthalten
 - ▶ ActionScript
 - ▶ C
 - ▶ C#
 - ▶ C++
 - ▶ Delphi
 - ▶ Java
 - ▶ PHP
 - ▶ Python
 - ▶ Visual Basic
 - ▶ VB.Net
- Template für LotusScript selbst entworfen (Beta-Qualität)



Language: LotusScript

Templates:

Name	Modified
Namespace Declaration I...	No
Class	Yes
Class Impl	No
Class Notes	Yes
Class Base	Yes
Class Interface	Yes
Class Body	Yes
Class Body Impl	Yes
Class Declaration	Yes
Class Declaration Impl	No
Class Inherits	Yes
Class Parameter	Yes
Attribute	Yes
Attribute Notes	Yes
Attribute Declaration	Yes
Import	No
Import Impl	No
Import Section	Yes
Import Section Impl	No
Linked Attribute	Yes
Linked Attribute Notes	Yes
Linked Attribute Declaration	Yes
Linked Class Base	Yes

Stereotype Overrides:

Class	Feature	Modified

```

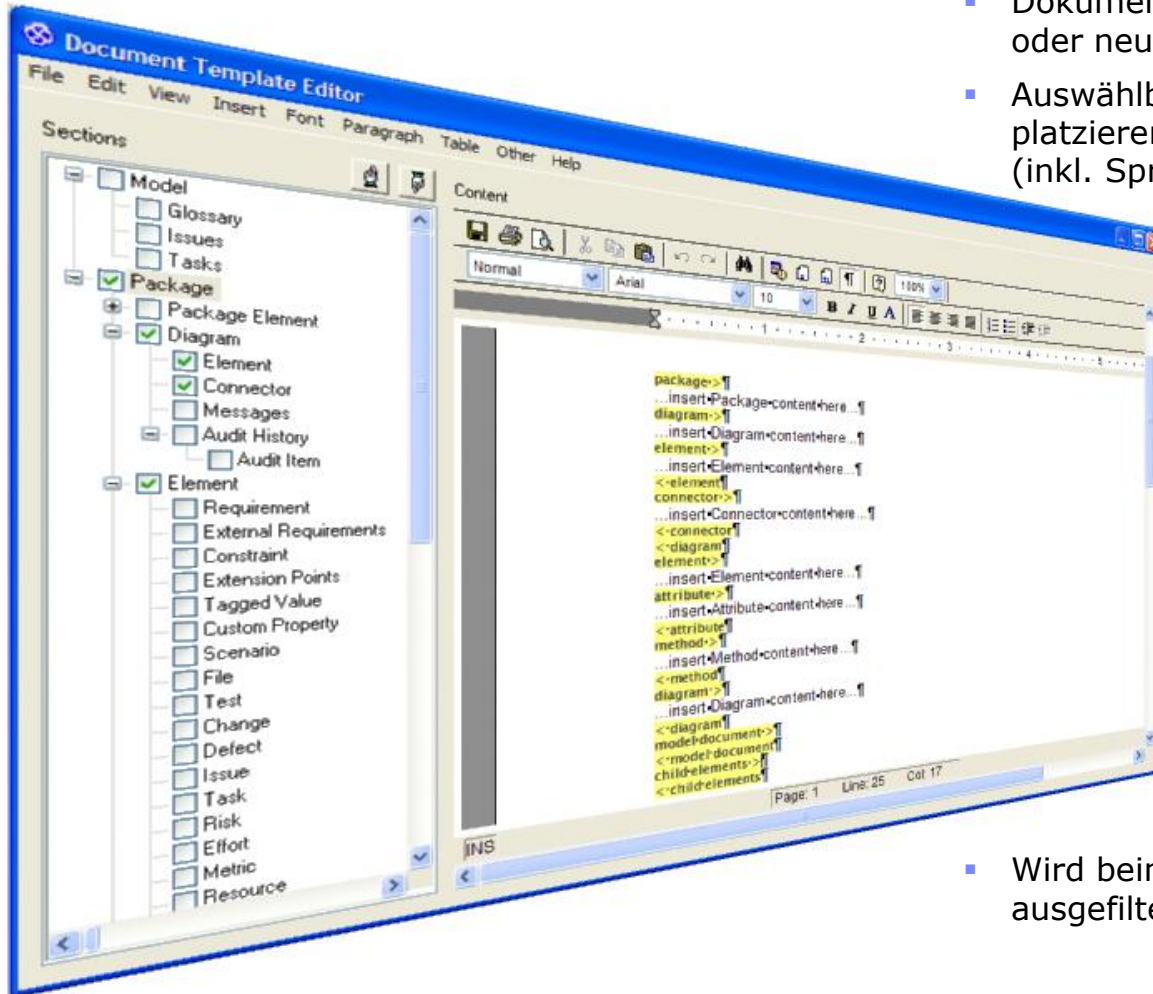
Template:
@if classTag:"Attribute" != ""$
$PI=""$
$sep="_\n"
$REPLACE(classTag:"Attribute", ";", $sep)$
$sep
$endif$
$generic = $list="ClassParameter" @separator=","$
@if $generic != ""$
$generic = "(Of " + $generic + ")"
$endif$

$PI=" "$
@if elemType == "Interface"$

$classTag:"partial"=="true" ? "Partial"$
$CONVERT_SCOPE(classScope)$
$classTag:"shadows"=="true" ? "Shadows" : ""$
$PI=""$
Interface $className$ $generic
$ClassInherits$
$endTemplate$

@if classStereotype == "module"$
$PI=""$
Module $className$
$else$
$classTag:"partial"=="true" ? "Partial"$
$CONVERT_SCOPE(classScope)$
$classTag:"shadows"=="true" ? "Shadows" : ""$
$PI=""$
Class $className$ $generic
$endif$
$ClassInherits$
    
```

Documentation Templates



- Dokumentationsvorlagen können verändert oder neu erstellt werden
- Auswählbare Platzhalter in eine RTF Vorlage platzieren und somit das eigene Layout (inkl. Sprache) festlegen

- Wird beim Generierungslauf iterativ mit ausgefilterten Daten befüllt



Diagramme und Code Generation Templates



Live-Demonstration



ERSTELLUNG DER SYSTEMDOKUMENTATION



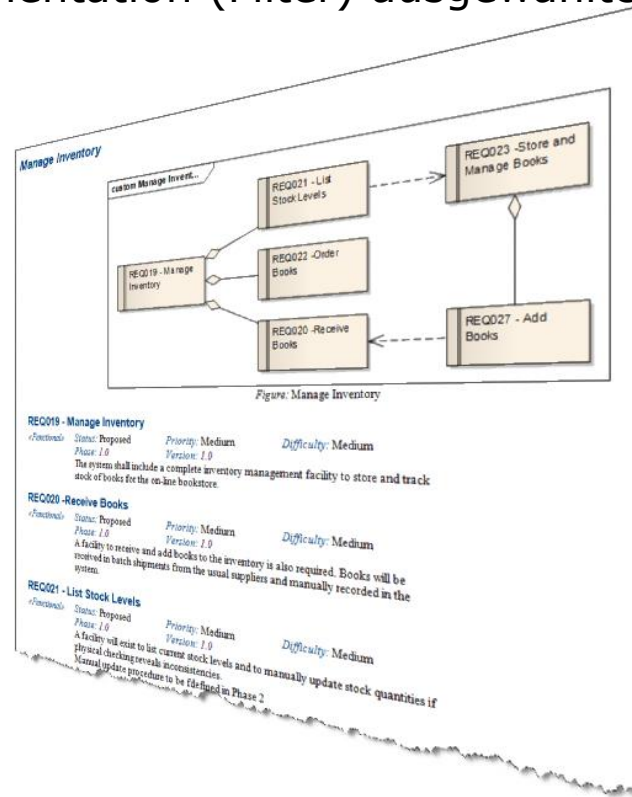
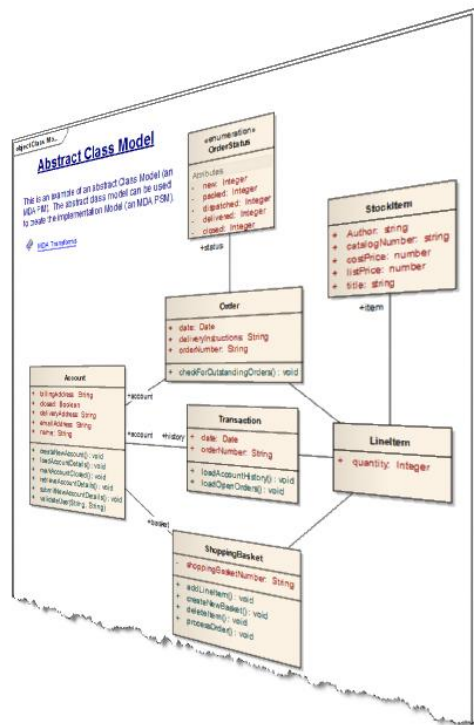
Projektdokumentation

- Erzeugt mit EA
- Anpassbare Dokumentationsschablonen
- Erspart doppelte Erstellung (Dokumentation als Abfallprodukt)
- Dokumentation der Klassen gelangt bis in den Code als Kommentar
- Kommentare aus dem gesamten Code extrahieren und als HTML bereitstellen



Systemdokumentation mit Enterprise Architect

- Angepasste Berichte in RTF oder HTML Format
- Verwendung von Dokumentationsvorlagen
- Begriffsübersetzungen
- Selektive Dokumentation (Filter) ausgewählter Bereiche



List of Project Issues	
Description	Resolution
The test server builds have been delayed because the particular (unusual) memory requirements to match the customer's site are not available on shore. They are being sourced from Singapore but it will delay the builds and delivery of the machines.	Resolved on: 13.10.2005 Resolved by: The machines will be built and delivered using standard memory and the proprietary memory will be added later. All performance tests will be delayed until the memory is available.
This model is yet to be completed.	Resolved on: 13.10.2005 Resolved by:
A number of the training CD's have not been replaced and training has had to be delayed.	Resolved on: 13.06.2005 Resolved by:
A number of the developers have downloaded different version of a number the compilers. This has lead to unpredictable builds impacting on testing.	Resolved on: 13.06.2005 Resolved by:
The schedule includes staff working on public holidays. A number of staff have indicated that contrary to what they stated earlier they will not be available.	Resolved on: 13.06.2005 Resolved by:


- Erstellt von Mikkel Heisterberg (im Rahmen eines Presseartikels)
- OpenSource Werkzeug
 - ▶ erstellt in LotusScript und Java mit zahlreichen eigenen Klassen
 - ▶ Separate Datenbank, die mit zyklischem Agent konfigurierte Datenbanken analysiert und dokumentiert
- Inline Kommentare im zu dokumentierenden Code sind an JavaDoc Syntax angelehnt
- Dokumentation wird zyklisch aus Klassendefinitionen und Inline Kommentaren als HTML erstellt
 - ▶ Stets aktuelle Produktdokumentation
 - ▶ Nachschlagewerk bei Programmierung und Debugging
- Eignet sich für alle Stellen, an denen LotusScript verwendet werden kann

Weitere Information und Download: <http://www.lsdoc.org>

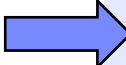


Kommentare im LotusScript Code ...

▪ Syntaxbeispiel



```
' /**
' * Adds a recipient to the e-mail.
' * <p></p>
' * @param recipient_type Use the XXX_RECIPIENT constants
to signal the type of recipient (To, CC, BCC).
' * @param username The address to send to
' * @error 9999 If the supplied type isn't one of the
allowed XXX_RECIPIENT constants
' */
Public Sub AddRecipient(recipient_type As Integer, username
As String)
```



▪ Zusätzliche @-parameter

- ▶ @param
- ▶ @return
- ▶ @depends
- ▶ @see
- ▶ @error
- ▶ @version
- ▶ @author



..erzeugt Website mit aktueller Dokumentation

[LS.Doc](#)
Inherits from: LotusScript.doc 2.0.0

Script libraries

[AppSLDesign7](#)
[CLASS: Collections](#)

Agents

All Classes

- [DesignBase](#)
- [Enumeration](#)
- [Hashtable](#)
- [HashtableEnumeration](#)
- [HashtableNode](#)
- [Queue](#)
- [QueueNode](#)
- [SortedVector](#)
- [Stack](#)
- [TypeEnumerationWrapper](#)
- [Vector](#)
- [VectorEnumeration](#)
- [memoryManager](#)

AppSLDesign7
Class DesignBase

Property Summary	
Public	GET formDocuments As Variant
Public	GET subformDocuments As Variant
Public	GET pageDocuments As Variant
Public	GET imageDocuments As Variant
Public	GET javaResourceDocuments As Variant
Public	GET viewDocuments As Variant
Public	GET folderDocuments As Variant
Public	GET navigatorDocuments As Variant
Public	GET framesetDocuments As Variant
Public	GET scriptLibraryDocuments As Variant
Public	GET sharedActionDocuments As Variant
Public	GET agentDocuments As Variant
Public	GET databaseScriptDocuments As Variant
Public	GET outlineDocuments As Variant
Public	GET stylesheetDocuments As Variant

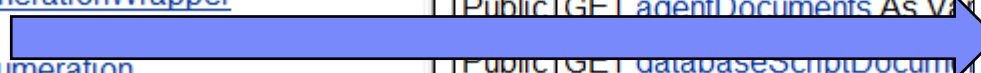
CLASS: Collections
Class Vector

Known sub-classes:

- [SortedVector](#)

Method Summary	
Public	new ()
Public	copyInto (outArray As Variant)
Public	trimToSize ()
Public	setCapacityIncrement (increment As Integer)
Public	ensureCapacity (minCapacity As Integer)
Public	addElement (element As Variant)
Public	addElement (elements As Variant)
Public	removeAllElements ()
Public	unique ()

Function Summary	
Public Variant	setSize (newSize As Integer)
Public Integer	size ()
Public Integer	capacity ()



LotusScript.Doc



Live-Demonstration





Lotus Domino Designer 8.5

DOMINO DESIGNER 8.5.1

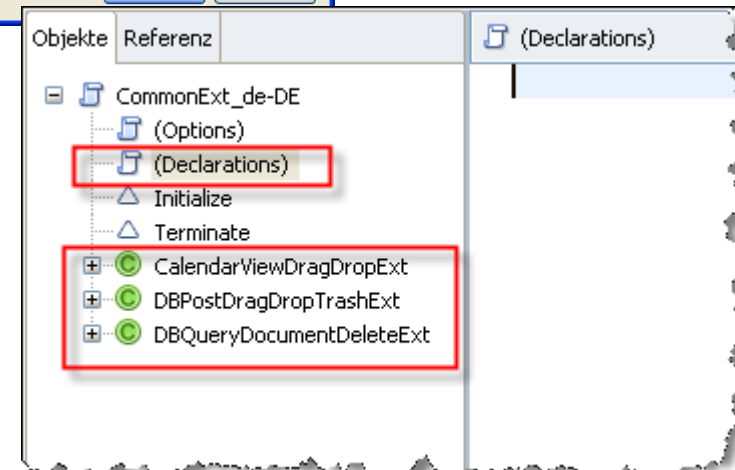
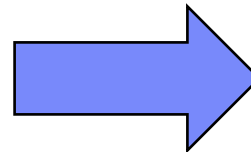
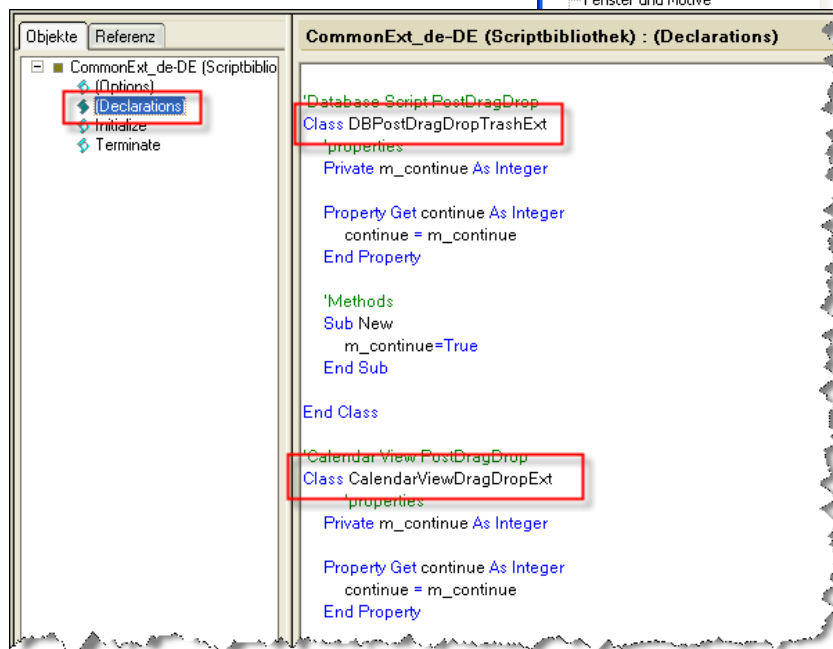
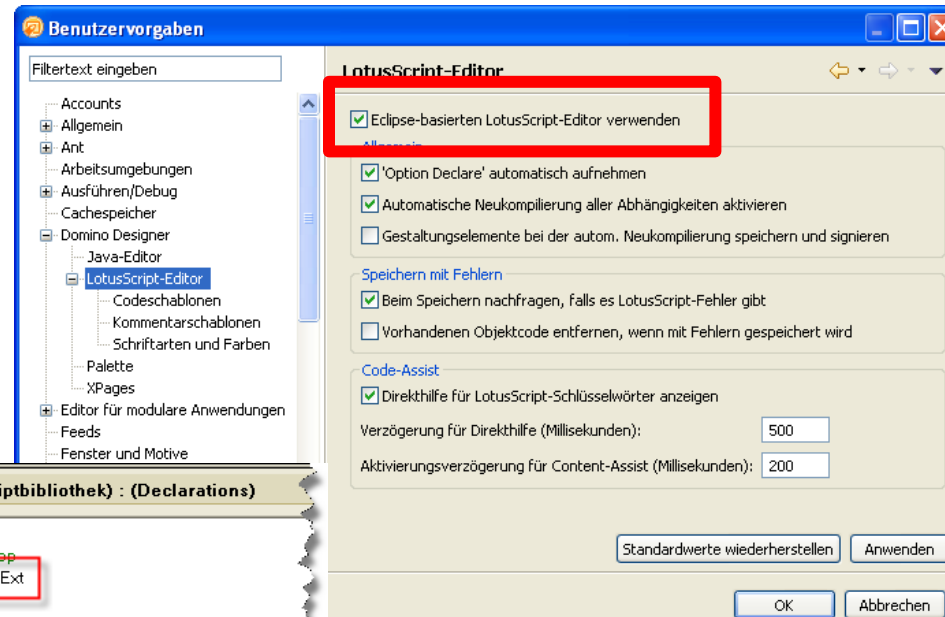


LotusScript Editor (8.5.1)

- jetzt auch Eclipse basiert
- aktivierbar und einstellbar über Vorgabendialog
- für Skriptbibliotheken und Agenten
- enthält Klassenbrowser (auch für eigene Klassen)
- Code- und Kommentarschablonen
- Auto-Completion (auch für eigene Klassen)
- Zeilennummern
- Syntax-Farbcodierungen
- Hyperlinking
- Speichern mit Fehlern
- automatische Neu-Kompilation abhängiger Komponenten
- automatische Sortierung von Functions, Subs, Classes



Vorgaben einstellen



Vorgaben einstellen

Benutzervorgaben

zeilen

- Allgemein
 - Editoren
 - Texteditoren**

Texteditoren

Protokollgrößenfestlegung rückgängig machen: 200

Angezeigte Tabulatorbreite: 4

Leerzeichen für Tabulatoren einfügen

Aktuelle Zeile hervorheben

Druckrand anzeigen

Spalte für Druckrand: 80

Zeilennummern anzeigen

Bereichsanzeiger anzeigen

Leerzeichen anzeigen

In K...

Bei Bew...

Zieh...

Vor...

Inte...

Farbop...

Zeilen...

Hervor...

Druck...

Suchbe...

Vorder...

Hinter...

Hinter...

Vorder...

Hyperl...

Weitere...

Objekte **Referenz**

- CommonExt_de-DE
 - (Options)
 - (Declarations)
 - Initialize
 - Terminate
 - CalendarViewDragDropExt
 - continue Abrufen von
 - m_continue : Integer
 - New
 - DBPostDragDropTrashExt
 - DBQueryDocumentDeleteExt

CalendarViewDragDropExt

```

1
2 'Calendar View PostDragDrop
3 Class CalendarViewDragDropExt
4   'properties
5   Private m_continue As Integer
6
7   Property Get continue As Integer
8     continue = m_continue
9   End Property
10
11 'Methods
12 Sub New
13     m_continue=True
14 End Sub
15 End Class
  
```

Code und Kommentarschablonen

Benutzervorgaben

Filtertext eingeben

- Accounts
- Allgemein
- Ant
- Arbeitsumgebungen
- Ausführen/Debug
- Cachespeicher
- Domino Designer
 - Java-Editor
 - LotusScript-Editor
 - Codeschablonen
 - Kommentarschablonen**
 - Schriftarten und Farben
- Palette
- XPages
- Editor für modulare Anwendungen
- Feeds
- Fenster und Motive
- Fensterobjekte
- Home-Portalaccount
- Installation/Update
- Java
- JavaScript

Kommentarschablonen

Kommentare automatisch in neue Codeelemente aufnehmen

Codeelement:

- Design Element
- Class**
- Type
- Function
- Sub
- Property Get
- Property Set

Kommentarschablone:

```
%REM
  ${element_type} ${element_name}
  Description: Comments for ${element_type}
%END REM
```

Objekte Referenz

- MyClass
- ExploreDesigner
 - (Options)
 - (Declarations)
 - Initialize
 - Terminate
 - MyClass

```
1 %REM
2   Class MyClass
3   Description: Comments for Class
4 %END REM
5 Class MyClass
6
7 End Class
```

Code und Kommentarschablonen

Benutzervorgaben

Filtertext eingeben

- Java-Editor
 - LotusScript-Editor
 - Codeschablonen**
 - Kommentarschablonen
 - Schriftarten und Farben
 - Palette
 - XPages
- Editor für modulare Anwendungen
- Feeds
- Fenster und Motive
- Fensterobjekte
- Home-Portalaccount
- Installation/Update
- Java
- JavaScript
- Kalender und Aufgaben
- Kontakte
- Livertext
- Ländereinstellungen
- Mail
- Notes Client-Basiskonfiguration
- Notes-Ports
- Plug-in-Entwicklung
- Protokolleinstellungen
- Prüfung
- Rechtschreibprüfung
- Replikation und Synchronisation

Codeschablonen

Schablone automatisch in neue Codeelemente aufnehmen

Codeelement:

- Design Element
- Class**
- Type
- Sub
- Function
- Property Get
- Property Set

Codeschablone:

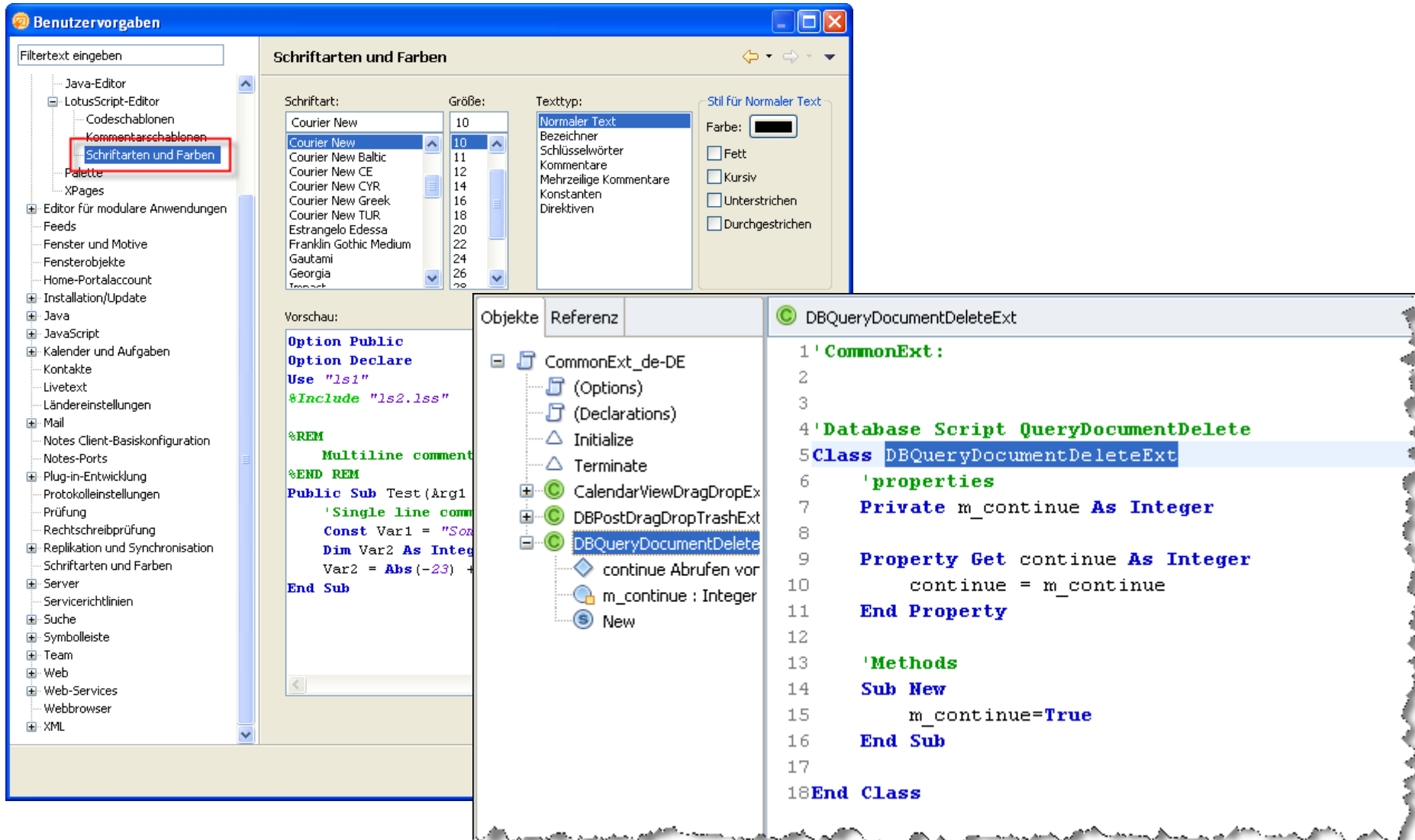
```

Sub New
End Sub

Sub Delete
End Sub
  
```

Objekte	Referenz
<ul style="list-style-type: none"> ExploreDesigner <ul style="list-style-type: none"> (Options) (Declarations) Initialize Terminate MyClass 	<pre> MyClass 1 %REM 2 Class MyClass 3 Description: Comments for Class 4 %END REM 5 Class MyClass 6 Sub New 7 8 End Sub 9 10 11 Sub Delete 12 13 End Sub 14 15 End Class </pre>

Syntax-Highlighting



Benutzervorgaben

Filtertext eingeben

- Java-Editor
 - LotusScript-Editor
 - Codeschablonen
 - Kommentarschablonen
 - Schriftarten und Farben**
 - Palette
 - XPages
- Editor für modulare Anwendungen
 - Feeds
 - Fenster und Motive
 - Fensterobjekte
 - Home-Portalaccount
- Installation/Update
- Java
- JavaScript
- Kalender und Aufgaben
- Kontakte
- Livertext
- Ländereinstellungen
- Mail
 - Notes Client-Basiskonfiguration
 - Notes-Ports
- Plug-in-Entwicklung
 - Protokolleinstellungen
 - Prüfung
 - Rechtschreibprüfung
- Replikation und Synchronisation
 - Schriftarten und Farben
- Server
 - Servicerichtlinien
- Suche
- Symboleiste
- Team
- Web
 - Web-Services
 - Webbrowser
- XML

Schriftarten und Farben

Schriftart: Courier New Größe: 10 Texttyp: Normaler Text

Stil für Normalen Text

Farbe:

Fett

Kursiv

Unterstrichen

Durchgestrichen

Vorschau:

```
Option Public
Option Declare
Use "ls1"
#include "ls2.lss"

%REM
Multiline comment
%END REM
Public Sub Test (Arg1
'Single line comment
Const Var1 = "Soz
Dim Var2 As Integer
Var2 = Abs(-23) +
End Sub
```

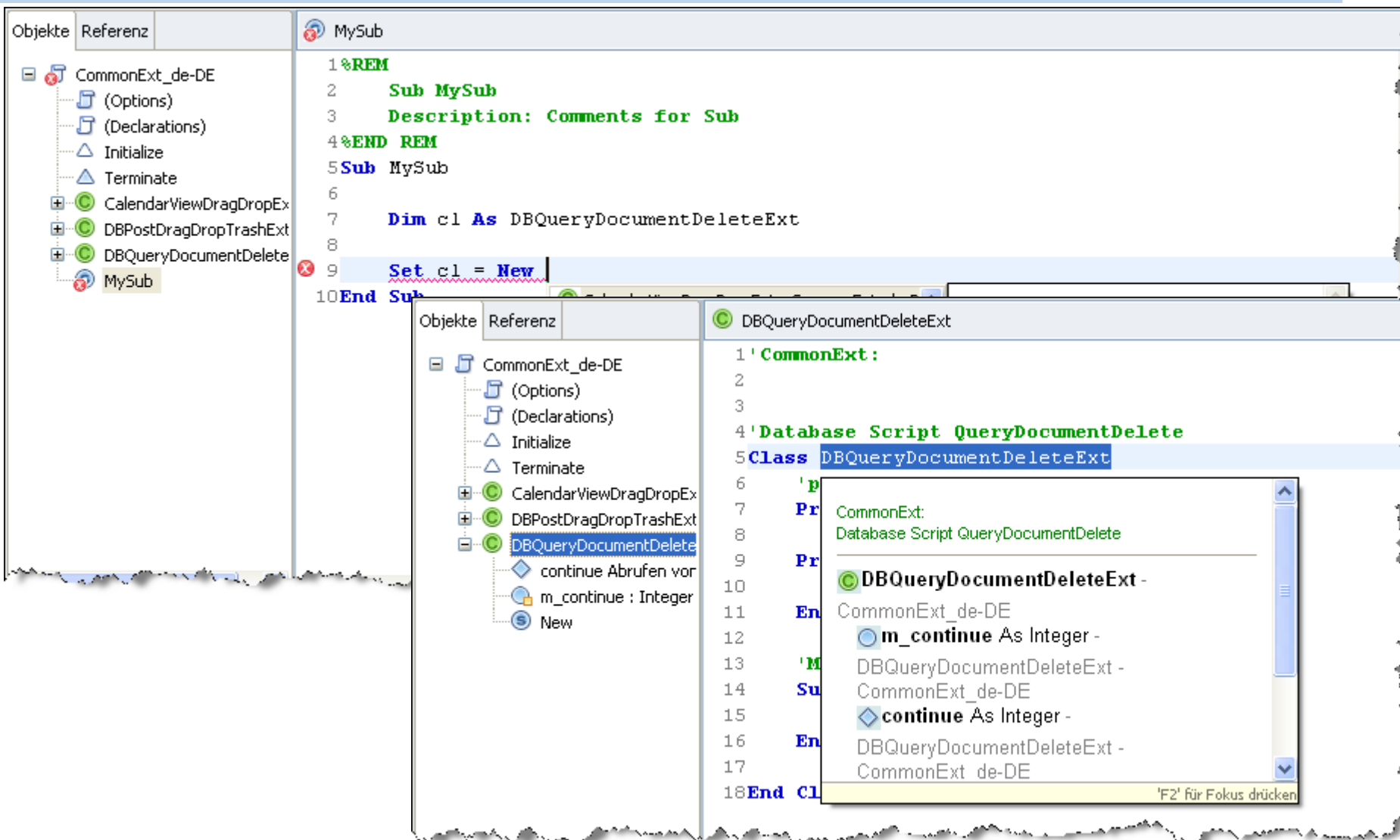
Objekte Referenz

- CommonExt_de-DE
 - (Options)
 - (Declarations)
 - Initialize
 - Terminate
 - CalendarViewDragDropExt
 - DBPostDragDropTrashExt
 - DBQueryDocumentDelete**
 - continue Abrufen vor
 - m_continue : Integer
 - New

DBQueryDocumentDeleteExt

```
1 'CommonExt:
2
3
4 'Database Script QueryDocumentDelete
5 Class DBQueryDocumentDeleteExt
6 'properties
7 Private m_continue As Integer
8
9 Property Get continue As Integer
10 continue = m_continue
11 End Property
12
13 'Methods
14 Sub New
15 m_continue=True
16 End Sub
17
18 End Class
```

Auto-Vervollständigung sowie Hover-Help und -Links



The screenshot illustrates the Visual Studio IDE with the following components:

- Top Window (MySub):** Shows VBA code with line numbers 1-10. Line 9, `Set c1 = New`, is highlighted. A red 'x' icon is visible next to the line number.
- Left Window (Object Explorer):** Shows a project tree for 'CommonExt_de-DE' with sub-items: (Options), (Declarations), Initialize, Terminate, CalendarViewDragDropExt, DBPostDragDropTrashExt, DBQueryDocumentDelete, and MySub.
- Bottom Window (DBQueryDocumentDeleteExt):** Shows the class definition for 'DBQueryDocumentDeleteExt'. Line 5, `Class DBQueryDocumentDeleteExt`, is highlighted. A tooltip is displayed over this line, showing:
 - Project: CommonExt_de-DE
 - Member: m_continue As Integer
 - Member: continue As Integer

Skripte mit Fehlern speichern

Objekte	Referenz	MyClass - New
ExploreDesigner		1 %REM
(Options)		2 Class MyClass
(Declarations)		3 Description: Comments for Class
Initialize		4 %END REM
Terminate		5 Class MyClass
MyClass		6 Sub New
Delete		7
New		8 Dim
		9
		10 End Sub

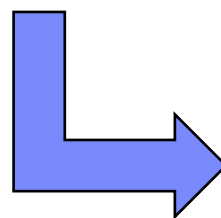
Speichern mit Fehlern

Das Script enthält Fehler. Möchten Sie es trotzdem speichern?

Ja Nein

Anwendungen

- Doctor Notes
 - \Hub\SVR\WWW\Corp\mail\dr_notes.nsf
- Masken
- Ansichten
- Ordner
- XPages
- Benutzerdefinierte Steuerelemente
- Rahmengruppen
- Seiten
- Gemeinsame Elemente
- Code
- Agenten
- Gemeinsame Aktionen
- Scriptorchestrator
- A ExploreDesigner
- ACLManagement_de-DE
- AddressBookSync_de-DE
- AddressParser_de-DE



..und trotzdem ist erkennbar, wo noch ein Fehler ist

TESTVERFAHREN FÜR OBJEKTORIENTIERTE ANWENDUNGEN



Umfangreiche Unit-Tests sichern Projekterfolg

- Mehrstufige Tests erforderlich
 - ▶ Unit-Test Entwickler
 - ▶ Integrationstests QS
 - ▶ Abnahmetests Fachabteilung / Auftraggeber

- Unit-Tests möglichst automatisieren
 - ▶ Wiederholbarkeit
 - ▶ Dokumentation
 - ▶ Debugging

- Integrationstests
 - ▶ Nicht zu früh

- Abnahmetests
 - ▶ Auf der Basis der erfassten und dokumentierten Anforderungen



Unit-Tests

- Separates Werkzeug, um Klassen in isolierter (einfacher Umgebung) zu testen
 - ▶ Whitebox-Tests
 - ▶ Blackbox-Tests
- Aufruf (möglichst) aller Methoden, um eine 100% Testabdeckung zu erhalten (im integrierten Anwendungssystem nicht möglich)
- Tests müssen auch Randbedingungen (ungültige oder nicht vorhandene Daten) simulieren können, um Aussagen für das spätere Systemverhalten zu erhalten
- Bei Fehlern bildet diese Umgebung die Basis für isolierte Tests und Weiterentwicklung von Basisklassen



Klassentester

- Manuelle Prüfungen
 - ▶ UI Elemente (Maske mit Button oder Agent) um Testcases aufzurufen
 - ▶ Manuelle Dateneingabe und Ergebnisprüfung

- Automatische Prüfungen
 - ▶ UI Elemente (Maske mit Button oder Agent) um Testcases aufzurufen
 - ▶ Programmgesteuertes Testverfahren auf der Basis eines OpenSource Testframeworks („Domino Unit Framework“ Projekt von Tony Palmer bei OpenNTF)
 - ▶ Testergebnisse werden dokumentiert als
 - Fehler (Laufzeitfehler)
 - Fehlverhalten (Funktionsergebnis nicht erfüllt)

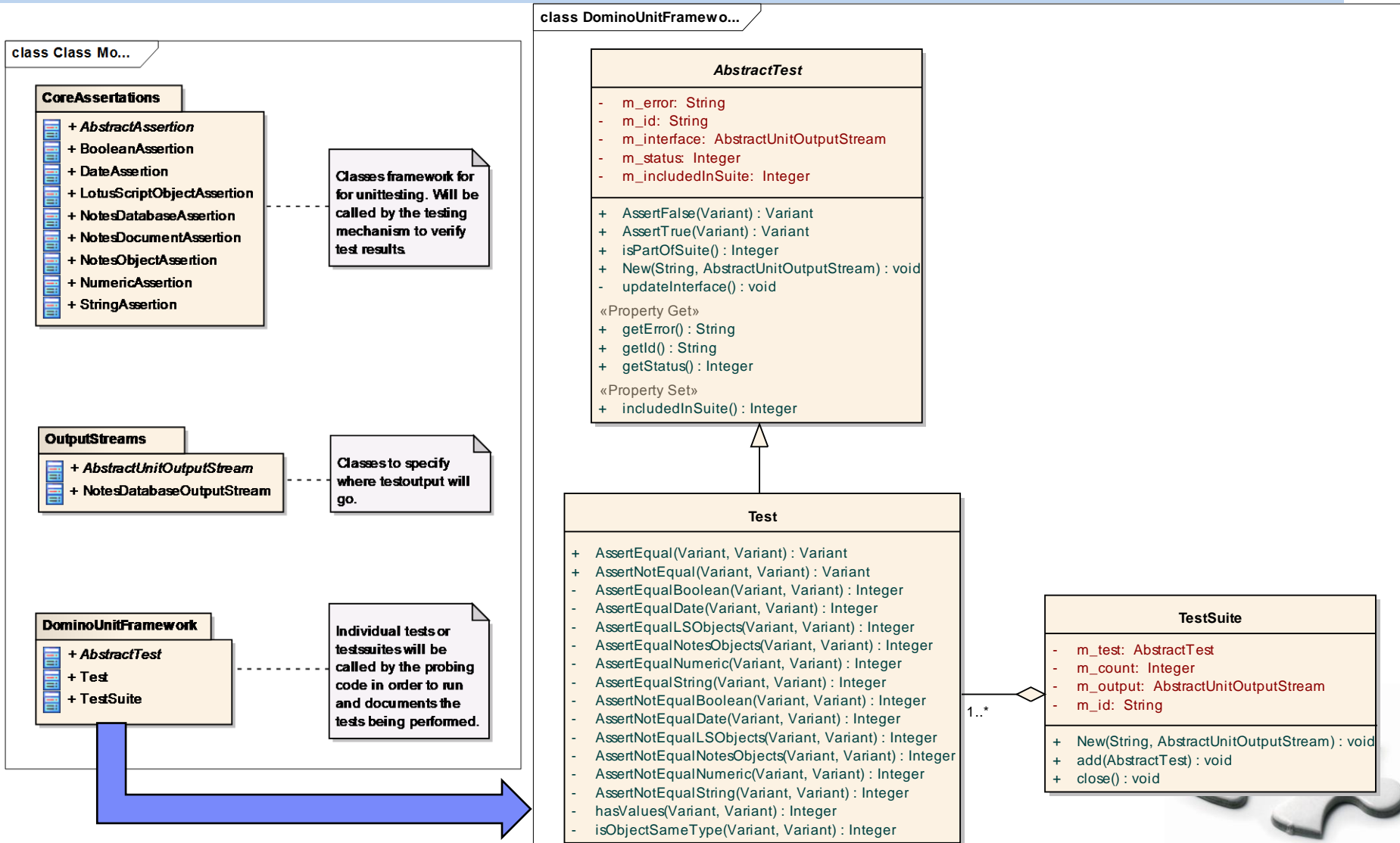


Domino Unit Framework

- Angelehnt an jUnit (<http://www.junit.org> und <http://de.wikipedia.org/wiki/JUnit>)
- Besteht aus Skriptbibliotheken, die abstrakte und konkrete Methoden für Testabläufe und Testcases beinhalten
- Verwendet „assert<Vergleich>“ Methoden um Prüfungen durchzuführen:
 - ▶ assertEquals
 - ▶ assertNotEqual
 - ▶ assertTrue
 - ▶ assertFalse
- Unterstütze Datentypen
 - ▶ Numeric
 - Integer
 - Long
 - Double
 - currency
 - ▶ LS Date
 - ▶ Boolean (Integer)
 - ▶ String
 - ▶ NotesObjects
 - NotesDocument (based on UNID)
 - NotesDatabase (based on Replica Id)
- Unterstützung eigener Klassen
 - ▶ Klasse muss die assert-Methode(n) implementieren und spezifische Prüfung durchzuführen



Domino Unit Framework in Enterprise Architect



DUF Beispiel

```

'-- Testergebnisse
Dim OutputStream As NotesDatabaseOutputStream      'lege Ergebnisse in Datenbank ab
Set OutputStream = New NotesDatabaseOutputStream(session, "", "DominoUnitResults.nsf")

'-- Test allgemeiner Datentypen
Dim testObject As New Test("Test Objects", OutputStream)
Call testObject.AssertEqual(session.currentdatabase, Null)
Call testObject.AssertNotEqual(session.currentdatabase, Null)

Dim testDiff As New Test("Test Different", OutputStream)
Call testDiff.AssertEqual("Test", 1)

Dim testTrue As New Test("Test True", OutputStream)
Call testTrue.AssertTrue(True)

Dim testFalse As New Test("Test False", OutputStream)
Call testFalse.AssertFalse(False)

Dim testTrue2 As New Test("Test True but False", OutputStream)
Call testTrue2.AssertTrue(False)

'-- Test eigener Klassen
Dim testPerson As New Test("Test eigener Personenklasse", OutputStream)
Dim objPerson1 As New Person
Dim objPerson2 As New Person

Call objPerson1.init("Manfred", "", "Meise")
Call objPerson2.init("Manfred", "", "Meise")

Call testPerson.AssertEqual(objPerson1, objPerson2)

'- Snippet der Klassendefinition
Class Person
...
Function isEqual(people As Person) As Integer
' add in your own code that determines equality or not.
End Function
..
End Class

```

Testwerkzeuge



Live-Demonstration



INTEGRATION DER WERKZEUGE



Klassentwurf

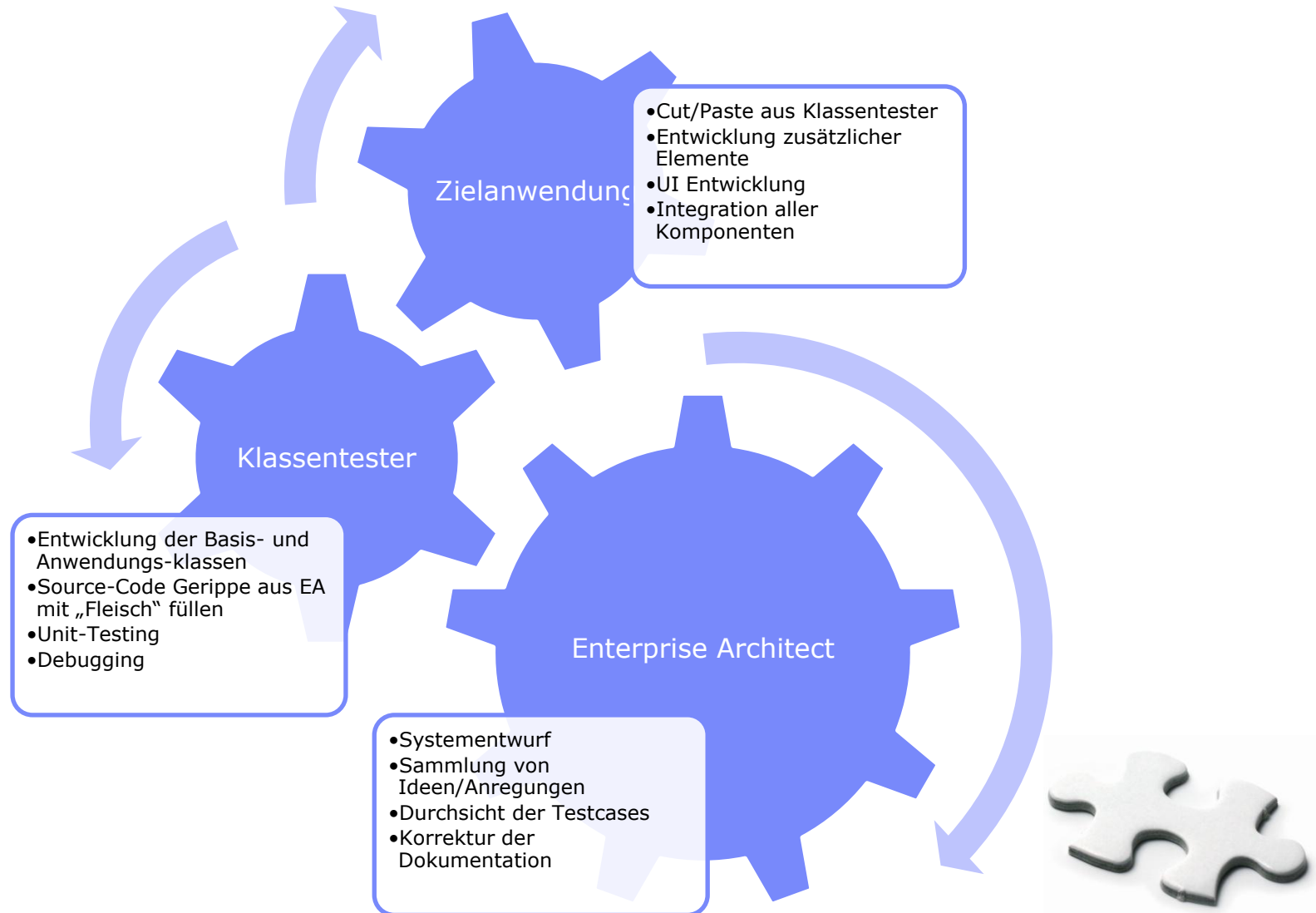
- Systementwurf mit Enterprise Architect
 - ▶ mit verschiedenen Diagrammen
 - ▶ Klassendiagramm mit Kommentaren für Klassen, Methoden, Eigenschaften und Variablen
 - ▶ Angestrebte Paketierung als Dokumentation der LotusScript Bibliotheken
 - ▶ Codegenerierung ins Filesystem

- Source-Code-Gerippe mittels Import oder Cut/Paste in den Domino Designer übernehmen

- Da von EA kein Code-Round-Trip für selbst definierte Sprachschablonen unterstützt wird, sind nachträgliche Änderungen/Erweiterungen an der Klassenstruktur leider mit „Fummelei“ verbunden



Implementierung / Test



Integrations- und Abnahmetests

- Basis für die Tests bildet die abgestimmte Anforderungsdefinition
 - ▶ Requirements Specification aus Enterprise Architect
 - ▶ Erfasste Testcases aus Enterprise Architect
- Tests durch Dritte Person (QA) bzw. Fachabteilung
- Fehler in ein Bugtracking Tool übernehmen



RESUMEE UND AUSBLICK



Mein Fazit der letzten 5 Jahre

- Objektorientierung und OOA / OOD / MDG sind eigentlich ein alter Hut
- Auch Domino Entwickler erkennen seit einigen Jahren Vorteile für eigene Anwendungen
- Bordmittel bislang noch zu spärlich: Erst Zusatzwerkzeuge erlauben effizientes Arbeiten
- Wenn man schon mit Werkzeugen arbeitet, dann gleich für große Teile des Projektes (den gesamten Projektzyklus)
 - ▶ Zusätzliche Tools erlauben meist mehr, als man gerade benötigt
 - ▶ So rutscht man automatisch in methodische Arbeitsweisen
- Vorgehensweise bis zum ersten vorzeigbaren Arbeitsergebnis aufwändiger als klassisches Rapid Prototyping, Wartungsaufwand erheblich geringer
- Ausstehende Tests und Arbeiten
 - ▶ EA Codetemplate korrigieren und vervollständigen
 - SourceForge Projekt: <http://eactls.sourceforge.net/>
 - ▶ EA Codegenierung in das virtuelle Dateisystem des Domino Designers
 - ▶ Round-Trip für LotusScript??
 - ▶ Testtools generalisieren und optimieren



Noch Fragen offen geblieben?



<http://www.mmi-consult.de>
<http://www.mmi-consult.de/faq>
<mailto:manfred.meise@mmi-consult.de>

