







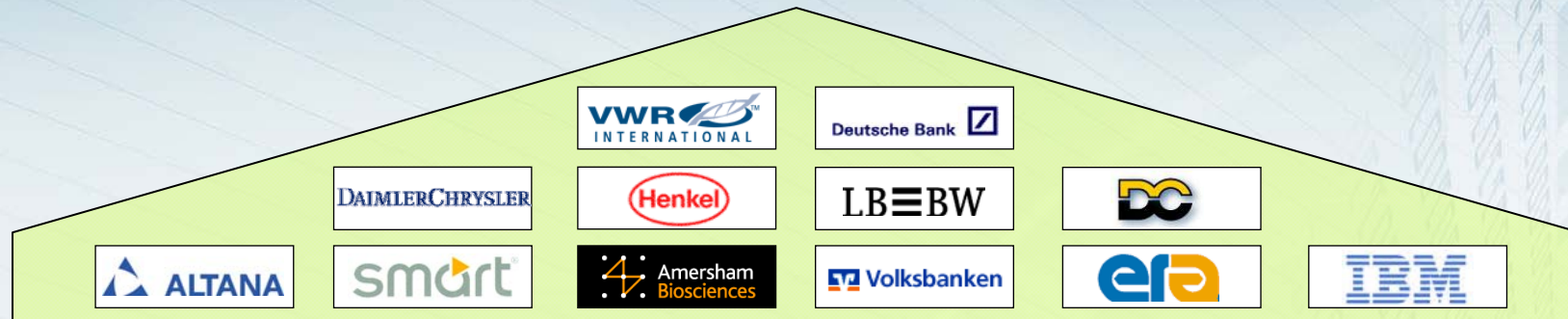
Transformation von Applikations-Logik über Plattformen hinweg via D² / XML / ALFRED

Dortmund, 22.02.2005

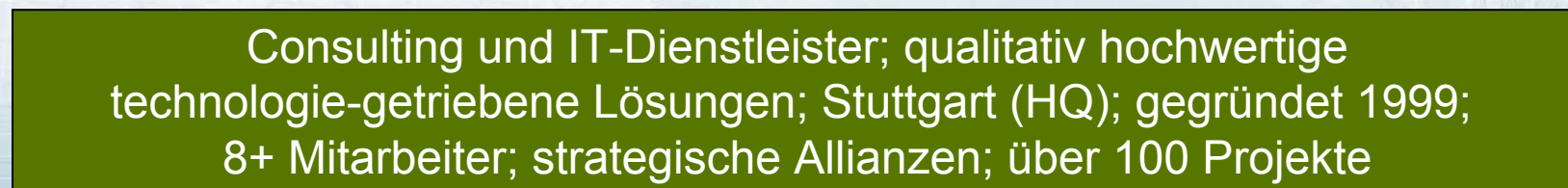
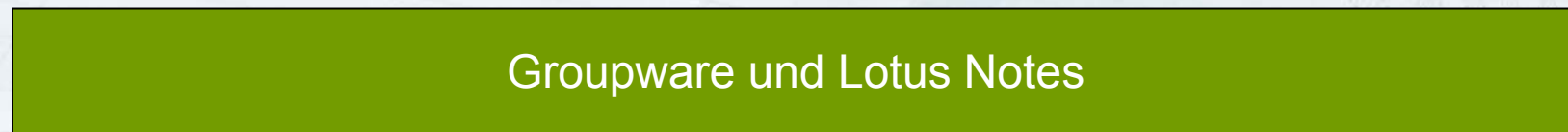
Veit Florian Lier (smartiX)


All trademarks are the
properties of their respective
owners

-  Prettiest girl on the dance-floor
-  Error-Handling
-  (D)XML – Darstellung von Anwendungen
-  Implikationen für Integration / Migration
-  Klassen-Modell auf Basis von D^2 (DD)
-  Produkte auf Basis von D^2 → ALFRED



Gründungsmitglied des deutschen Notes Expertenforums



 **10 Jahre Erfahrung mit Lotus (Domino)**

- ä Systemarchitektur, Entwicklung
- ä Lotus Workflow, Domino.Doc, LEI....
- ä Web-Anwendungen (ca. 70%), “reine” Lotus Notes-Anwendungen (ca. 30%)
- ä Anbindung an Backend-Daten-Systeme
- ä 8 x PCLP, 9 x CLP, ca. 14 x CLS

 **smartix consulting gmbh seit 1999**

- ä Mitglied der Penumbra Group
- ä Gründungsmitglied von B-KH

 **Aktuelle Tätigkeits-Schwerpunkte**

- ä COM-Schnittstellen
- ä XML / DXL
- ä 1 Woche Evaluierung von Optionen zur Integration LD / .Net in Redmond
- ä EntwicklerCamp 2005 (Dortmund)

 **Ist smartix verfügbar für Consulting/Entwicklung/Training?**

- ä Aber natürlich ☺

 **Email: Lier@smartix.de**

2 Fehlerarten:

- Geplante
- Ungeplante

Fehler-Handling in:

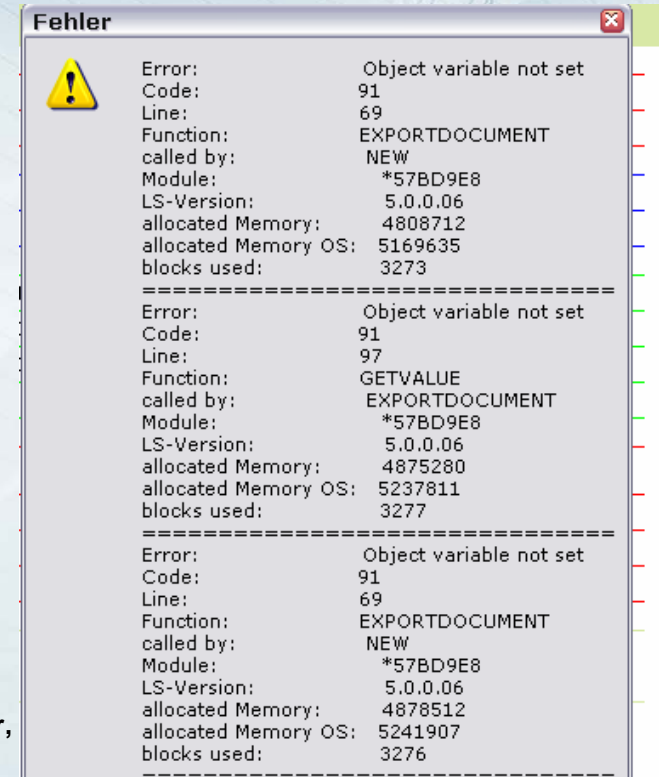
- @Formel-Sprache: (Beispiel 1)
 - `_CHECK:=@If(@IsError(_VALUE);"";_VALUE)`
- In LotusScript: (Beispiel 2)
 - On Error Goto ErrorHandler... Aber !!

```

str_Err      = str_Err & "Error:      " & Error & Chr(10)
'### Error-Code:
str_Err      = str_Err & "Code:      " & Cstr(Err())& Chr(10)
'### Error-Line:
str_Err      = str_Err & "Line:      " & Erl()   & Chr(10)
'### Current Function:
str_Err      = str_Err & "Function:   " & Lsi_info(2) & Chr(10)
'### Calling LS-Function:
str_Err      = str_Err & "called by:  " & Lsi_info(12) & Chr(10)
'### Current Module:
str_Err      = str_Err & "Module:    " & Lsi_info(3) & Chr(10)
'### LS-Version:
str_Err      = str_Err & "LS-Version:  " & Lsi_info(6) & Chr(10)
'### LS-Memory allocated:
str_Err      = str_Err & "allocated Memory: " & Lsi_info(50) & Chr(10)
'### LS-Memory allocated from OS:
str_Err      = str_Err & "allocated Memory OS: " & Lsi_info(51) & Chr(10)
'### LS-blocks used:
str_Err      = str_Err & "blocks used:  " & Lsi_info(52)      & Chr(10)
    
```

ErrorHandler:

Call Error_Set(Cstr(Erl()), Cstr(Lsi_info(2)), Cstr(Lsi_info(3)), Cstr(Lsi_info(12)), Error,
Resume Next



 Lotus Notes

(Beispiel 4)

 .Net

(Beispiel 5)

 MS Office

(Beispiel 6)

 SharePoint

 WebServices

(Beispiel 5)

<database>

➤ <form name=„(Config.ContactProfile)“...>

■ <body>

- <richtext>...
- </richtext>

■ </body>

➤ </form>

➤ <form name=„.....“>

<w:wordDocument>

➤ <w:body>

■ <wx:sect>

- <w:tbl>...
- </w:tbl >

■ </wx:sect >

➤ </w:body >


NotesDatabase

➤ NotesDocumentCollection

- NotesDocument
 - NotesItem
 - NotesItem

➤ NotesForm

- Forall Field in NotesForm.Fields
 -
- End Forall

 XML-Darstellung impliziert Hierarchien

 Hierarchien implizieren Klassenkonzept und Vererbung

 Class Firma

➤ Class Abteilung

▪ Class Mitarbeiter

- Class Private as Address
- Class Job as Address
- Class Delivery as Address

Probleme der Kombinatorik

Felder

- in Tabellen
 - in Tabellen
 - in Sektionen...

Etc...

 ➔ Sind Felder / Elemente in einem anderen Kontext unterschiedlich?

- Nein, da XML-Beschreibung immer identisch

 Feld: (Beispiel 7)

▼ field	
60	code
60	datetimeformat
60	keywords
60	numberformat
>>>	60 allowmultivalues
>>>	60 allownew
>>>	60 allowtabout
>>>	60 border style
>>>	60 choicesdialog
>>>	60 columns
>>>	60 computeaftervalidation
>>>	60 dataconnectionfield
>>>	60 date
>>>	60 defaultfocus
>>>	60 description
>>>	60 digits
>>>	60 event
>>>	60 fieldhint
>>>	60 format
>>>	60 fourdigityear
>>>	60 fourdigityearfor21stcentury
>>>	60 height
>>>	60 helperbutton
>>>	60 kind

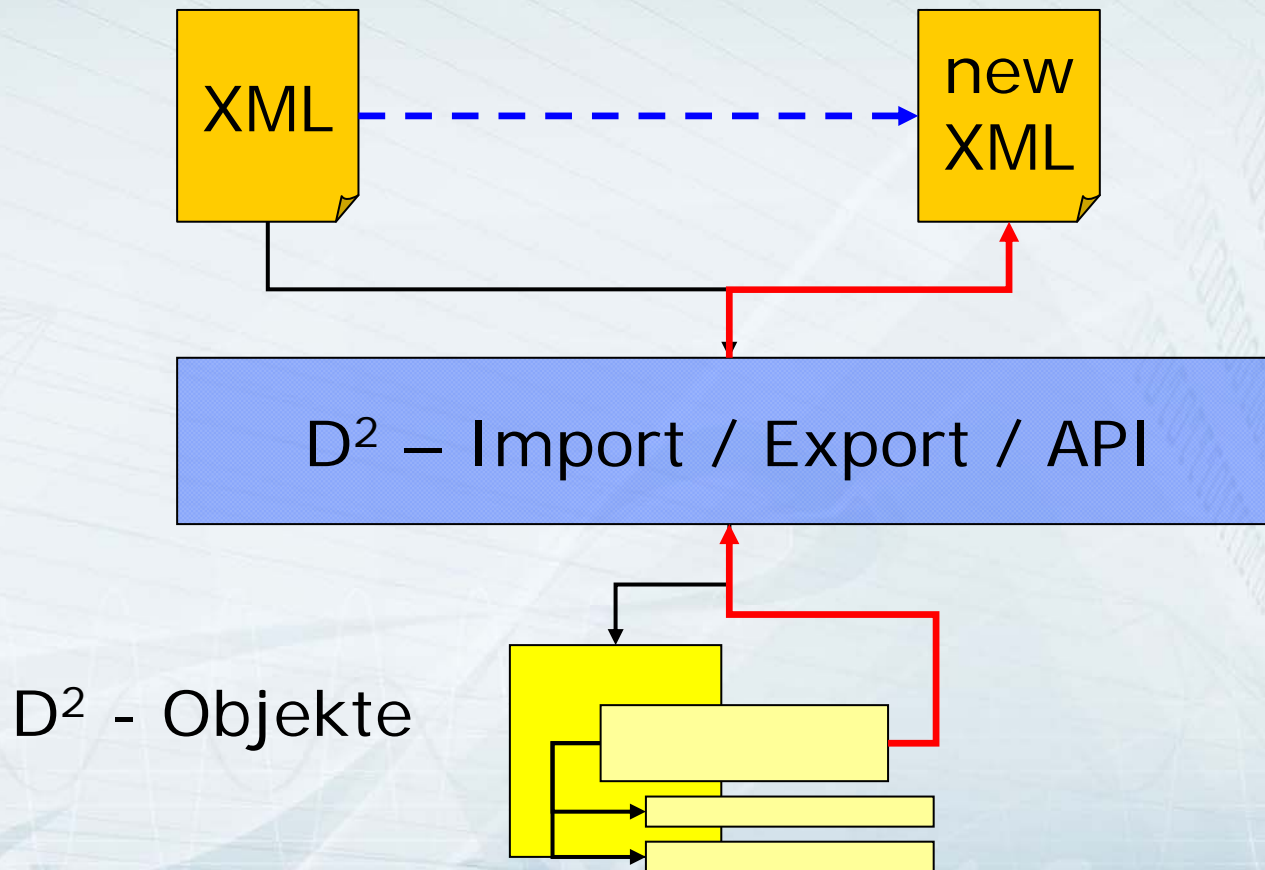
- 🐉 XML-Elemente sind immer „identisch“ aufgebaut (DDT)
- 🐉 Der Kontext und somit die Hierarchie sowie die „Eigenschaften“ können variieren und komplexer werden.
- 🐉 → Applikationen lassen sich durch sprachenspezifische Objekte und Attribute beschreiben und generieren!

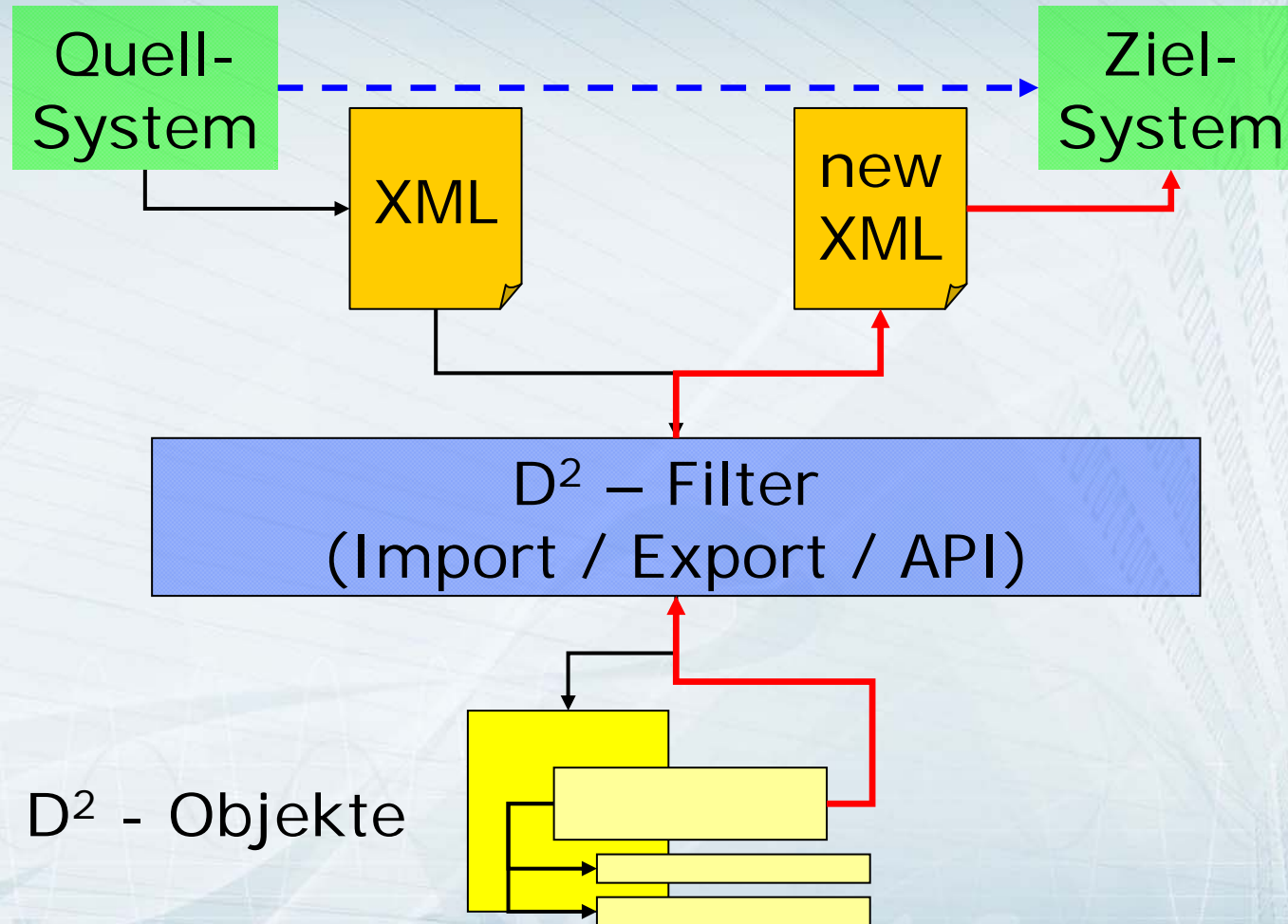
Wenn wir:

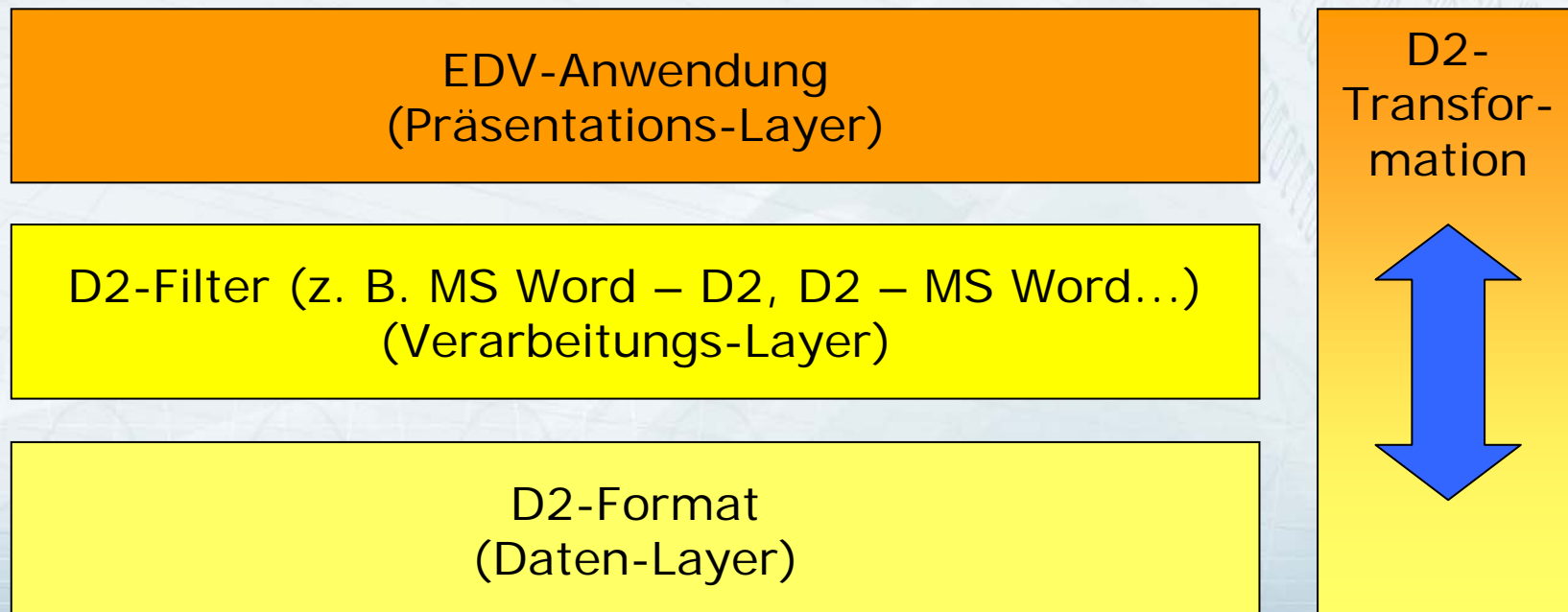
- Ein selbstlernendes,
- Sprachen-und
- Quellen-unabhängiges

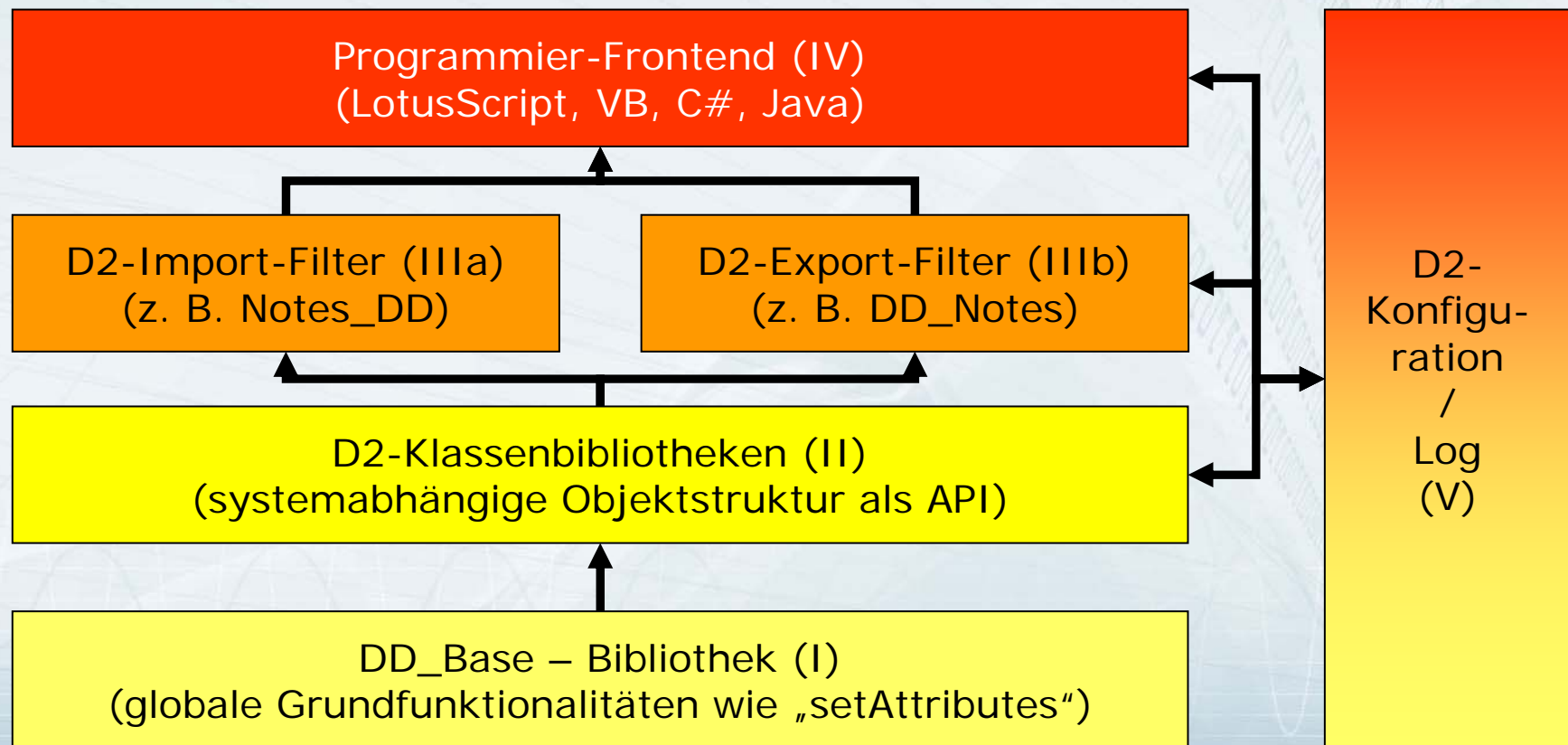
Beschreibungsmodell für Applikationen erschaffen und in eine Programmierplattform überführen können, sind wir in der Lage via definierter Filter:

- Elemente positiv / negativ zu identifizieren (Zeile QuellCode)
- Applikationslogik beliebig via Filter zu übersetzen











Code	2	3
Daten	1	4
	Lotus Notes	.Net

Von	Nach	Technologien	Bewertung	Begründung
1	4	WebServices / Konnektoren / XML	FH: + / ++ EA: + KA: ++ S: ++ GWG: a)	<p>Wird die Verarbeitungslogik in Lotus Domino belassen, muss diese „on the fly“ auf die Inhalte in anderen Datenspeichern zugreifen. Neben Produkten wie LEI bieten sich hier WebServices oder XML-Schnittstellen an.</p> <p>Dieser Schritt muss positiv bewertet werden, da eine Standardisierung von Datenaustauschformaten (WebServices) mehr Flexibilität für die Zukunft garantiert</p>
2	2	Neuentwicklung DXL	FH: - / + EA: + KA: +(+) S: + GWG: b), c)	<p>Sollen bestehende Applikationen auf ein neues Programmier-Paradigma (Styleguide) gehoben werden, kann dies entweder durch eine komplette Neuentwicklung oder eine tool-gestützte Überarbeitung durch die DXL-Klassen in Lotus Domino geschehen.</p> <p>Wird hierfür ein „Entwicklungs-Tool“ verwendet, fällt eine Kosten-Nutzen-Analyse sehr positiv aus. Wird alles in Handarbeit durchgeführt, stehen die anfallenden Aufwände in keinem Verhältnis zu dem möglichen Nutzen.</p>
3	3	Migrations-Assistenten	FH: -- EA: - / + KA: - / + S: (--) GWG: b), c)	<p>Für den Upgrade von bestehenden VB-Applikationen bietet Microsoft im Regelfalle Upgrade-Code im Rahmen ihrer Produkte an (siehe Access).</p> <p>Solche „Standard-Funktionen“ sollten ernsthafte Fragen hinsichtlich der Kompatibilität von Daten über den Zeitverlauf hinweg aufwerfen und sind daher negativ zu bewerten.</p>

Von	Nach	Technologien	Bewertung	Begründung
1+2	3+4	Komplex – Transformation auf Basis von Code: D ² – Daten: RDBS – übergangsweise Proposition N2N, InfoPath, SharePoint	FH: + EA: - / + KA: -- S: (--) GWG: b)	Ein Wechsel der verwendeten Plattform und Daten entspricht einem kompletten Redesign von allen existierenden Anwendungen / Ressourcen. Bei einfachen Anwendungen (Abwesenheitsmeldung, Besuchsberichte) ist dies einfach möglich, bei komplexeren prozessorientierten Anwendungen aufgrund von unterschiedlichen Sicherheitskonzepten sehr teuer bis technisch unmöglich
3+4	1+2	Komplex – Transformation auf Basis von Code: D ² – Daten: RDBS, XML, WebServices	FH: + EA: - / + KA: - S: (-) GWG: b)	Wie in dem oben beschriebenen Szenario ein kompletter Wechsel der gesamten „Umwelt“. Aufgrund der höher entwickelten Sicherheit (auf Feld-Ebene) in Lotus Domino aber eine sinnvolle Bewegung. Die im Regelfalle etwas einfacheren .Net-Applikationen machen den Wechsel in diese Richtung ein bisschen weniger schmerzvoll
2	3	DXL, XML, ADO, Proposition N2N, D ²	FH: + EA: ++ KA: - S: (+) GWG: b)	Ein Wechsel der Code-Basis von Lotus Domino hin zu .Net erfordert eine komplette Neu-Entwicklung. Alternativ kann der Code mittels einer Transformations-Software und entsprechenden .Net-Filtern sowie die Daten über die ADO-Schnittstelle „Proposition N2N“ übersetzt und überführt werden. Bei einfachen Anwendungen kann sehr viel automatisiert werden, bei komplexeren Anwendungen ist sehr viel manuelle Nacharbeit notwendig. Dies kann zu extrem hohen Kosten und Umsetzungsdauern führen
1+2	1+3	WebServices, ADO, Proposition N2N	FH: + EA: ++ KA: - S: (+) GWG: b), c)	Aufgrund der einfacheren Plattform für die Erstellung von WebServices unter .Net ist dieser Weg (unter Beibehalt der Daten in Lotus Domino) Stand Ende 2004 empfehlenswert. Allerdings hebt sich dieser Vorteil mit dem Einsatz von Lotus Domino R 7 zu weiten Teilen wieder auf, so dass nur kurzfristig Vorteile erzielt werden können.

- Lesen von XML / HTML
- Überführen in Strukturbaum
- Identifikation eindeutiger Elemente
- Generierung von Klassen-Code:
 - Set
 - Get
 - New
 - GenerateXML
 - ...
- Bereitstellung als:
 - LotusScript
 - Java
 - C+ / C++ / C#

Beispiel 8:

- Generierung neuer Struktur
- Parsen
- Modifizieren
- Zurückschreiben

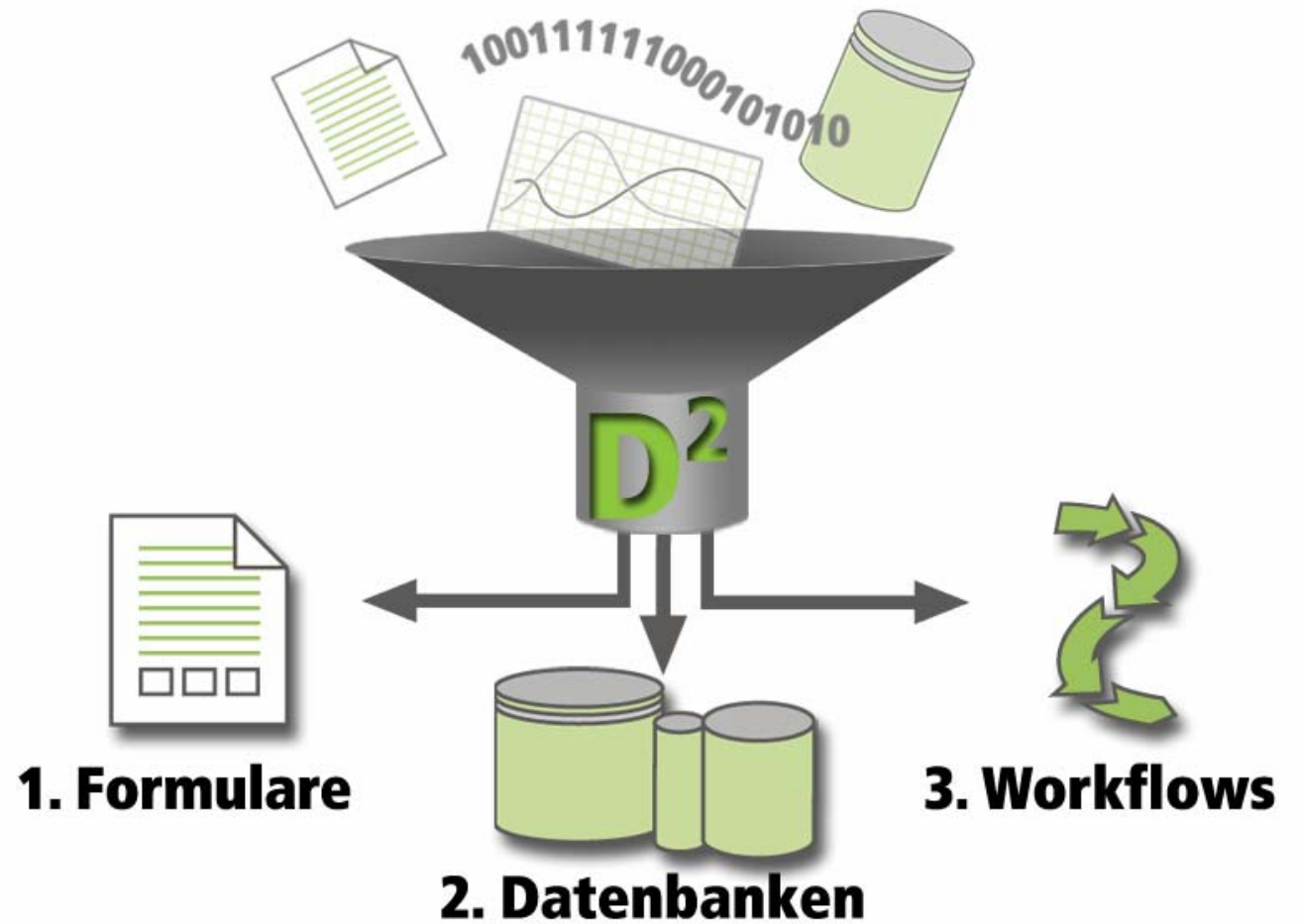
- 🐉 Identifikation von Code – Aufwandsabschätzung
- 🐉 Vorbereitung von Migrationen
- 🐉 Identifikation von Problemfeldern
- 🐉 Selbstlernendes System (**Rad neu erfinden**)
- 🐉 Plattformunabhängiges Shiften von Logik

Allgemeiner Formular Editor

- Erstellt einfache Lotus Notes Formulare, Datenbanken und Workflows ohne den Designer und Programmierkenntnisse.
- Intelligenter Einsatz von Formelsprache. Verwendet D², die zum Patent angemeldete Programmiersprache von smartix.
- Objektorientierter Programmaufbau.



Logik



Nutzen

- Reduziert Entwicklungszeit um bis zu 50%.
- Liefert qualitativ hochwertige Lösungen, einheitliche Logik und eine vollständige Dokumentation der Anwendung.
- Schnelle Umsetzung (time zu market) von Ideen, da der Power User selbst die „Entwicklung“ vornimmt.



Vorteile für Endkunden

- Verringert die Notwendigkeit externer Dienstleister.
- Verringert Abstimmungsaufwendungen mit Programmierern und Dienstleistern.
- Return on Investment bereits bei der zweiten erstellten Anwendung.
- Der Power User setzt seine eigenen Vorstellungen selbst um.
„You get what you want“



Vorteile für Softwarehäuser

Entwicklungsaufwand sinkt, dadurch:

- höhere Margen in Projekten.
- niedrigere Einstiegspreise bei Neukunden.
- Einfaches Prototyping – clickable Demo.



Preise

Endkunden, (MA):

1-50	€ 3.000
51 – 2.000	€ 6.000
2.001 – 10.000	€ 12.000
> 10.000	€ 25.000

Entwicklungslizenz: € 12.000

Maintenance: 30% p.a.

Schulung zzgl. Spesen: € 950



smartix
consulting gmbh

Kontakt

smartix consulting gmbh
Bismarckstr. 81
70197 Stuttgart / Germany

www.smartix.de

<http://www.myalfred.de>

myalfred@smartix.de



Fragen ??? und Antworten !!!

GMI KG · Die Gesellschaft für Migration und Integration



Vielen Dank für Ihre Aufmerksamkeit