Domain Specific Languages

Entwicklungsphasen für
Anwendungsbezogene Sprachen
(Domain Specific Language, DSL)

präsentiert von

Achim Schumacher

im Seminar

Software Language Engeneering

HTML: Anwendungsbezogene Sprache

```
<h1>Titel</h1>
<p>Lorem <i>ipsum</i>
dolor sit amet,
<u>consectetur</u>,
sadipisci</p>
```

Titel

Lorem *ipsum* dolor sit amet, consectetur, sadpisci

- Anweisungen für den Browser
- Anwendungsbereich („Domäne“): Layout und Hypertext (bspw. World Wide Web)
- Programmierung einfacher als in allgemein anwendbarer Programmiersprache

Weitere Beispiele für DSL

LaTeX	Typografie
SQL	Datenbankanfragen
VHDL	Hardwaredesign
Make	Softwareaktualisierung
Excel-Makro	Tabellenkalkulation
UML	Grafische Modellierung
Backus-Naur-Form	Spezifikation von Syntax

Inhalt

1. Nutzen und Eigenschaften von DSL
2. Phasenmodell für DSL-Entwicklung
3. Kurzvorstellung der Phasen
4. Fazit und Zusammenfassung

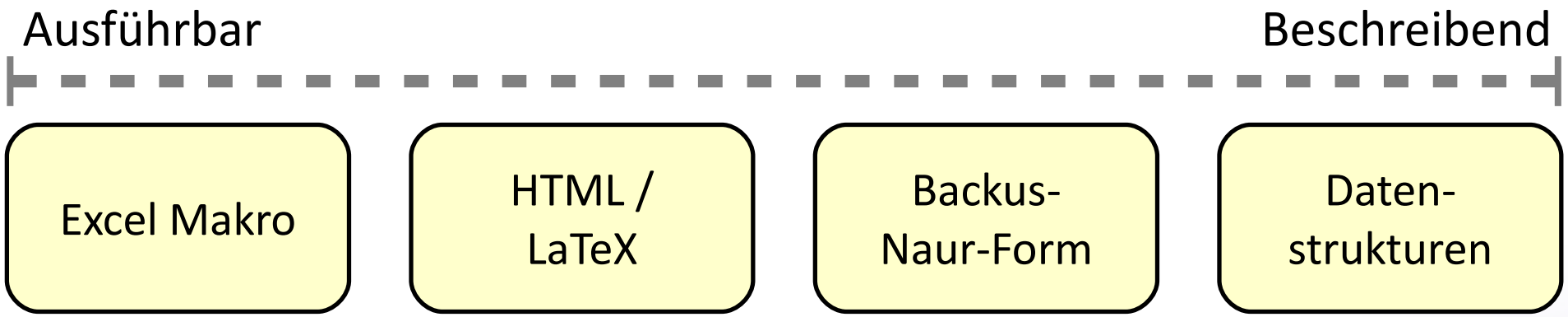
Nutzen von DSL

- Zugeschnitten auf Anwendungsbereich
  - Keine unnötigen Konstrukte ohne Domänenbezug
    → schneller erlernbar, effizienter nutzbar
- Kapselung von Domänenwissen
  - Ersetzung von (sich wiederholenden) Details durch aussagekräftige, kompakte Sprachkonstrukte
- Engere Sprache → Ausdrucksstärker
  - Beispiel: Kontextfreie Grammatiken:
    Konzepte auf hohem Niveau

Nutzen von DSL

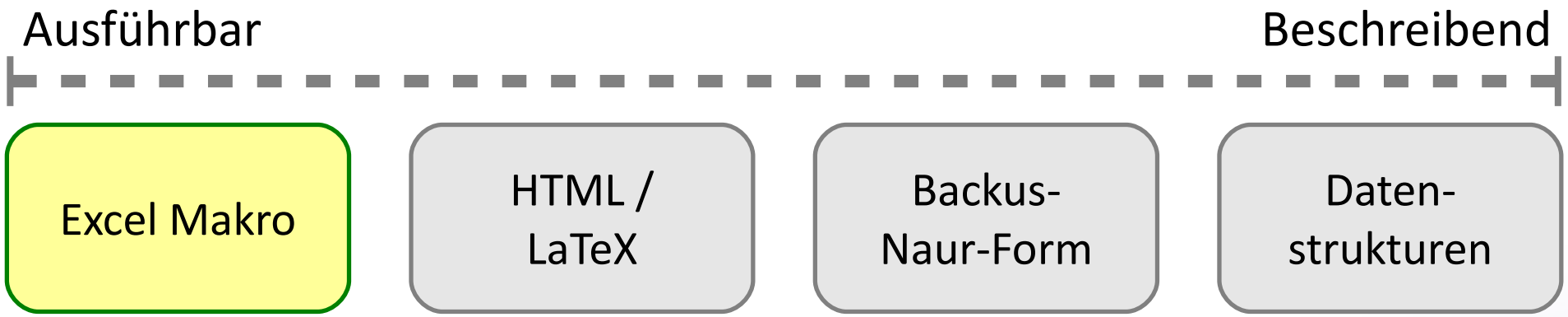
- Nutzung einfacher als bei allgemein anwendbarer Programmiersprache („general programming language“, GPL)
  - Vergrößerung der Zielgruppe für die Sprache
  - Einbezug von Benutzern ohne Programmiererfahrungen

Klassifizierung nach Ausführbarkeit



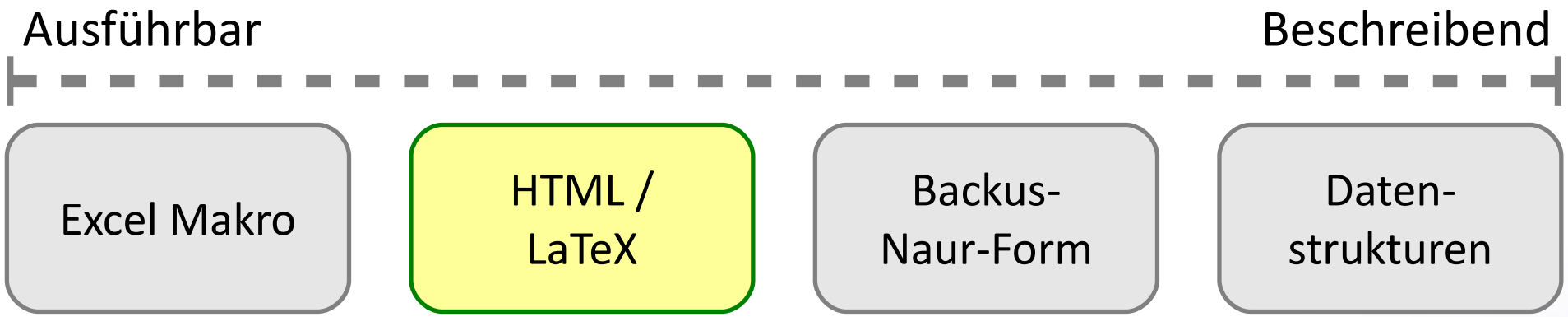
- DSL mit unterschiedlichem Grad an Ausführbarkeit und Beschreibung

Klassifizierung nach Ausführbarkeit



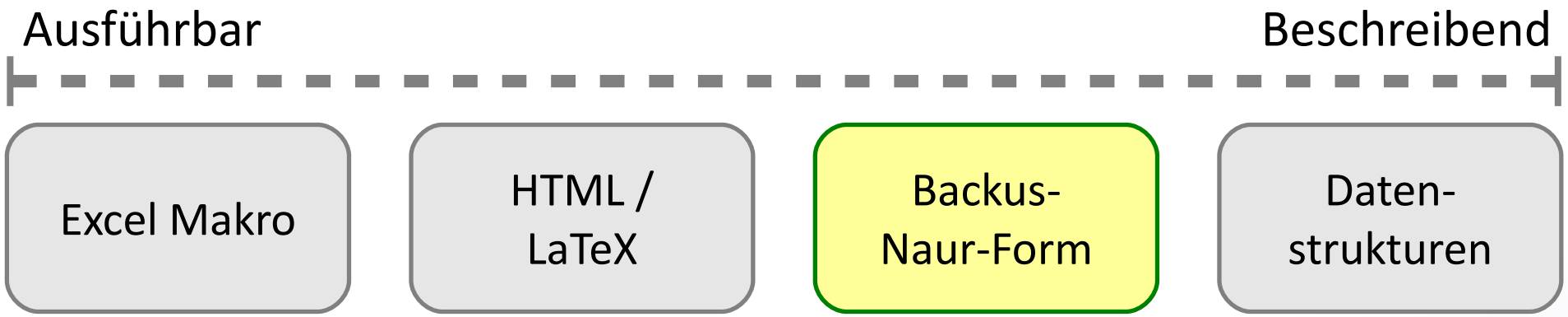
- Wohldefinierte Ausführungssemantik
  - Excel Makro: Ausführung von Operationen der Tabellenkalkulation
- Wenig oder gar nicht beschreibender Charakter

Klassifizierung nach Ausführbarkeit



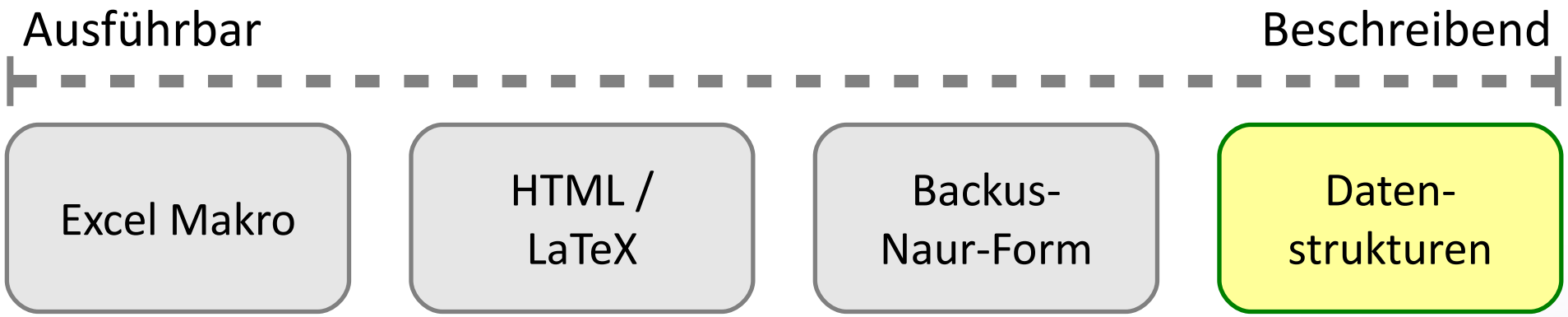
- Eingabesprachen über Übersetzer
  - LaTeX: Übersetzung in andere DSL wie PDF/PS
- Ausführbare Sprachen, aber auch beschreibend
  - HTML: Ausführbare Befehle für Browser und Beschreibung des Layouts

Klassifizierung nach Ausführbarkeit



- Primär beschreibend
  - Statische Strukturen
- Für Anwendungsgenerierung nutzbar
  - Backus-Naur-Form: Generierung von Parsern für die beschriebene Sprache

Klassifizierung nach Ausführbarkeit



- Nicht ausführbar
- Repräsentationen von Domänen-Spezifischen Datenstrukturen
- Analyse, Überprüfung von Konsistenz oder Visualisierung wie für ausführbare DSL möglich

Entwicklung in Phasen

- Aufteilung des Entwicklungsprozesses
  - Entscheidung → Analyse → Design → Implementierung → Einsatz
  - Modell, nicht zwangsweise sequenziell
- Muster für jede Phase: typische Fälle
  - Auswahl ohne Einfluss auf andere Phasen

Phase 1: Entscheidung

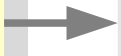
- Ökonomischer Vorteil durch Entwicklung
- Muster: Anwendungsszenarien für DSL

Notation	Notation für Domäne hinzufügen • textliche zusätzlich zu visueller Notation • benutzerfreundliche Notation für API
Aufgabenautomatisierung	Vermeidung wiederholender Aufgaben
Handhabung von Datenstrukturen	Vereinfachung von Beschreibungen und Durchlaufen von Daten
Anpassung von Benutzungsschnittstellen	• Interaktion programmierbar machen • Vereinfachung der GUI-Erstellung

Phase 1: Entscheidung. Beispiel

- Codegenerierung für Datenstrukturen

```
int{nummer}
String{name vorname}
```



```
private int nummer;
private String name, vorname;

public int getNummer()
{ return this.nummer; }

public void setNummer(int nummer)
{ this.nummer=nummer; }

public String getName()
{ return this.name; }

public void setName(String name)
{ this.name=name; }

public String getVorname() ...
```

- Kompaktere Schreibweise
- Mit verschiedenen Generatoren: verschiedene Zielsprachen

Phase 2: Analyse

- Identifizierung der Domäne
- Sammlung von Wissen
- Erstellung einer Ontologie
  - Sammlung von Begriffen, die in der Sprache umgesetzt werden sollen
- Quellen: Technische Dokumente, Experten, existierender GPL-Code
- Komplexer Sachverhalt

Phase 2: Analyse

- Ontologie: Wichtige Begriffe der Domäne
- Beispiel: HTML
  - Domäne: Layout und Hypertext
  - Begriffe: Absatz, Aufzählung, Bild, Fettschrift, Hyperlink, Kursivschrift, Tabelle, Überschrift, Zeilenumbruch, ...
- Umsetzung in zu entwickelnder Sprache

Phase 3: Design

- Festlegung der Notation der Sprache
- Beziehung zu existierenden Sprachen

Sprach-ausnutzung	(Teilweises) Nutzen einer existierenden GPL/DSL • „Huckepack“: Teilweise Nutzung • Spezialisierung: Begrenzung der existierenden Sprache • Erweiterung der existierenden Sprache
Sprach-erfindung	Kompletter Neuentwurf ohne Gemeinsamkeiten mit existierenden Sprachen

Phase 3: Design. Beispiel: HTML

- Auszeichnungssprache (Markup Language)
- Nutzung von Tags: Starttag und Endtag
  - Inhalt dazwischen
  - Auch Tags ohne Inhalt
- Schachtelung möglich
- Attribute im Starttag

```
<h1>Titel</h1>
<p>
  Lorem<br/>
  <i>ipsum
    <u>dolor</u>
    sit amet
  </i>
  <img src="img.gif"/>
</p>
```

Phase 3: Design. Beispiel: KFG

- Kontext-freie Grammatik: Definition konkreter Syntax
- Grammatik G=(T, N, P, S)
  - T: Menge Terminalsymbole
    N: Menge Nichtterminalsymb.
    S∈N: Startsymbol
    P: Produktionen
  - Produktionen+Startsymbol: weitere Mengen ergeben sich

```
T = {'a', 'b',
     'x', 'y'}
N = {A, B}
S = A
```

Menge der Produktionen

```
A ::= 'x' A 'y'
A ::= B
B ::= 'a'
B ::= 'b'
```

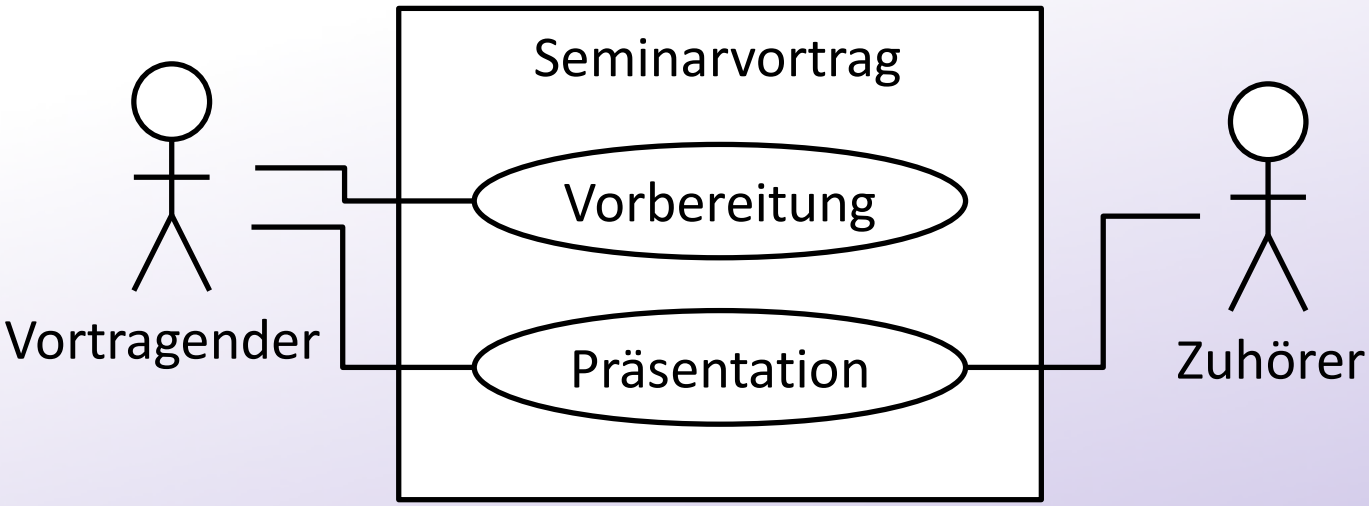
Erzeugbare Worte

```
a, b, xay, xby,
xxayy, xxbyy ...
```

Phase 3: Design. Beispiel: UML

- Grafische Notation zur Modellierung
- Klassen

- Use Cases



Phase 4: Implementierung

- Finden der geeigneten Implementierungstechnik

Interpretierer (Interpreter)	Zyklus: Holen, Dekodieren, Ausführen. → Flexibel und langsam
Übersetzer (Compiler)	Übersetzung in existierende Sprache Komplette statische Analyse des DSL-Codes
Präprozessor	Übersetzung in existierende Sprache Analyse des generierten Codes bei dessen Ausführung
Einbettung	Einfügen in bestehende Sprache Definition neuer abstrakter Datentypen, ... Einfachste Form: Anwendungsbibliothek

Phase 4: Implementierung

- Interpretierer für HTML: Webbrowser
  - Darstellung von Internetseiten oft bevor sie vollständig geladen sind → keine Übersetzung
  - Hohe Performance für HTML nicht vorrangig
- Übersetzer: komplette Analyse und Übersetzung
  - pdfLaTeX: LaTeX → Seitenbeschreibungssprache PDF
  - Generierung von Parsern für Sprachen, deren Syntax durch die Backus-Naur-Form definiert wurde

Phase 5: Einsatz

- Entwicklung nicht mit Produktion des Code abgeschlossen
- Erstellung von Trainingsmaterial
- Kommunikation mit Benutzern
- Keine Stagnation der Entwicklung
  - Anpassung der Sprache bei veränderten Anforderungen

Benötigte Expertise zur Entwicklung

- Domänen- und Sprachentwicklungswissen
  - Erfassung des Wissensgebiets: umfangreiche Aufgabe
  - Implementierung einer Sprache umsetzen können
- Erfahrung und Hilfe wichtig

Quellen

- Mernik, Heering und Sloane, 2005:
  When and how to develop domain-specific languages
  - DSL-Eigenschaften, Phasenmodell
- Kastens, 2009:
  Vorlesung Grundlagen der Programmiersprachen
  - Kontextfreie Grammatiken
- Kastens, 2008/09:
  Vorlesung Generating Software from Specifications
  - Domain-Specific Generator (DSL-Übersetzer)

Quellen

- Kelly und Pohjonen, 2009:
  Worst Practices for Domain-Specific Modeling
  - Ziel: Konkreter Ratgeber, Verdeutlichung von praktischen Problemen während der Entwicklung
  - Sehr eingeschränkte Datenbasis (76 Fälle bei 1 Firma mit 1 Werkzeug), unbelegte Thesen, vage Aussagen. Beispiel:
    - Problem (8%): Analyse bis die Sprache theoretisch komplett ist
      Begründung: Angst beim Entwickeln erster Sprache
    - Problem (4%): Ablehnung fremder Hilfe
      Begründung: Glauben, dass nur „Gurus“ Sprachen entwickeln

Zusammenfassung

- Domänen-spezifische Sprachen, zugeschnitten auf Anwendungsgebiet
- Entwicklung in Phasen

Entscheidung	Szenarien, in denen DSL sinnvoll sind
Analyse	Sammlung von Wissen, Erstellung einer Ontologie
Design	Beziehung zu anderen Sprachen Bestimmung der Notation (textlich/visuell)
Implementierung	Umsetzung als Interpretierer, Übersetzer, ...
Einsatz	Vermittlung an Benutzer, Weiterentwicklung

- Domänen- und Sprachentwicklungswissen