

Software Engineering Aspekte von Sprachen für die Netzwerkdarstellung

Software Engineering Aspekte - Ziele

- die alternativen Vorgehensweise *code first* und *communication first* erklären können und deren Vor- und Nachteile nennen können
- die Rolle und Bedeutung von Schemasprachen bei Entwicklung verteilter Systeme erklären können
- wesentliche Eigenschaften und Sprachkonstrukte von XML Schemasprachen erklären können
- Alternative Netzwerkdarstellungen einordnen können

Vorgehensalternative: Lokale Modelle exportieren (*Code first*)

- Quelle sind Programmiersprachen oder (relationale) Datenbanksysteme
- **Vorteil:** hohe Automatisierung
- **Nachteil:** Restriktionen und Spezifika des lokalen Systems werden allgemeingültig für alle beteiligten Komponenten vorgeschrieben.

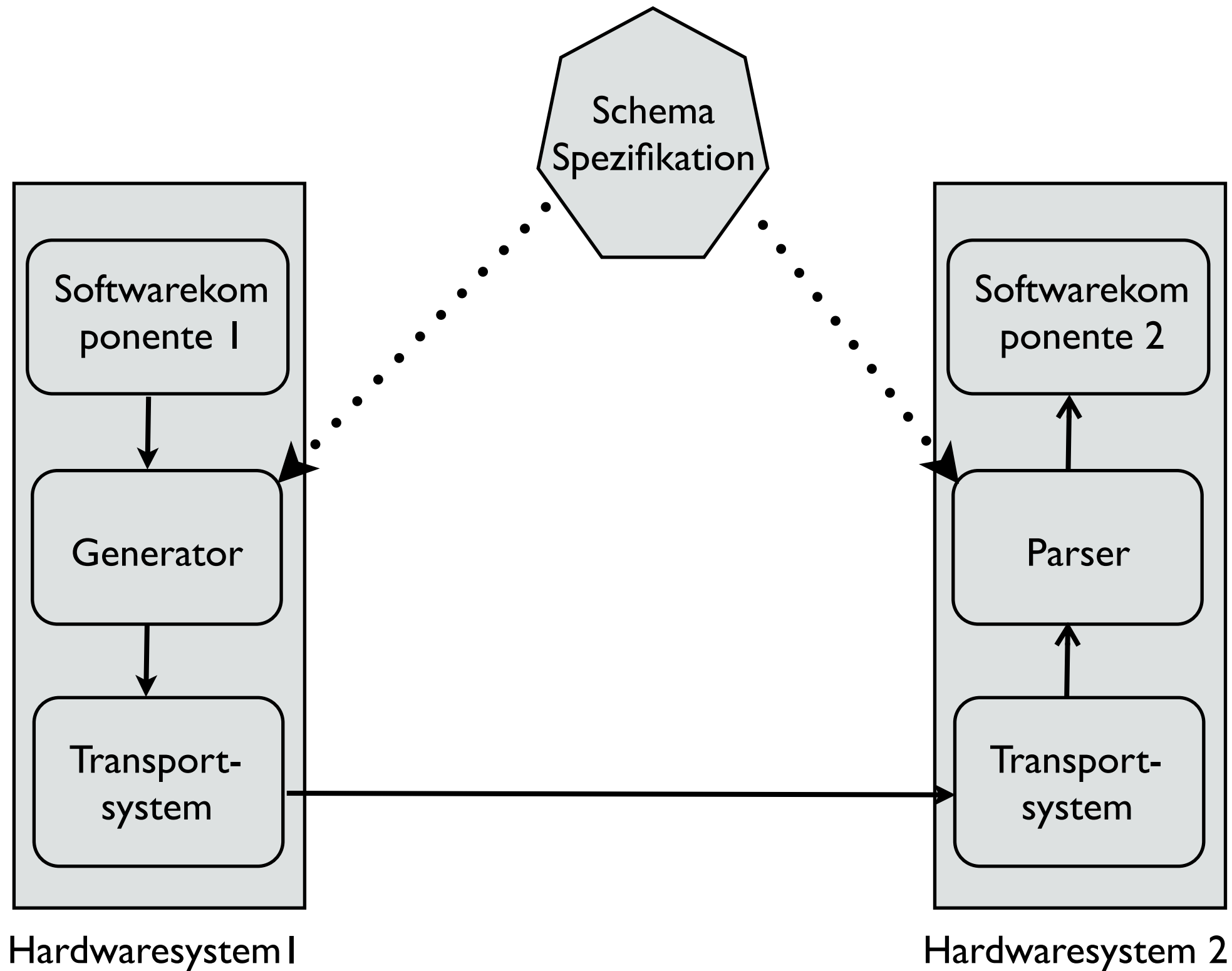
Vorgehensalternative: Netzwerkdarstellung importieren (*Communication first*)

- Struktur und Format werden global definiert und an die jeweiligen lokalen Programmiersprachen und Datenbanksysteme adaptiert.
- **Vorteil:** Format und Struktur können optimal an die Kommunikationsaufgabe angepasst werden und enthalten keine Spezifika der Implementierung.
- **Nachteil:** Anpassung an die lokalen Systeme kann aufwändig sein.

(XML) Schemasprachen

- Durch Sprachen wie XML oder JSON ist eine allgemeine Syntax zur Spezifikation von Datenstrukturen gegeben.
- Gerade für den Informationsaustausch ist es wichtig, dass das empfangende System sicherstellt (**validiert**), dass die **Struktur der empfangenen Daten korrekt** und somit **Bedeutung tragend** ist.
- Erforderlich ist ein Werkzeug das für **jede mögliche eine XML Struktur** feststellt, ob ein konkretes **XML Dokument** dieser Struktur genügt.

Rolle von Schemasprachen



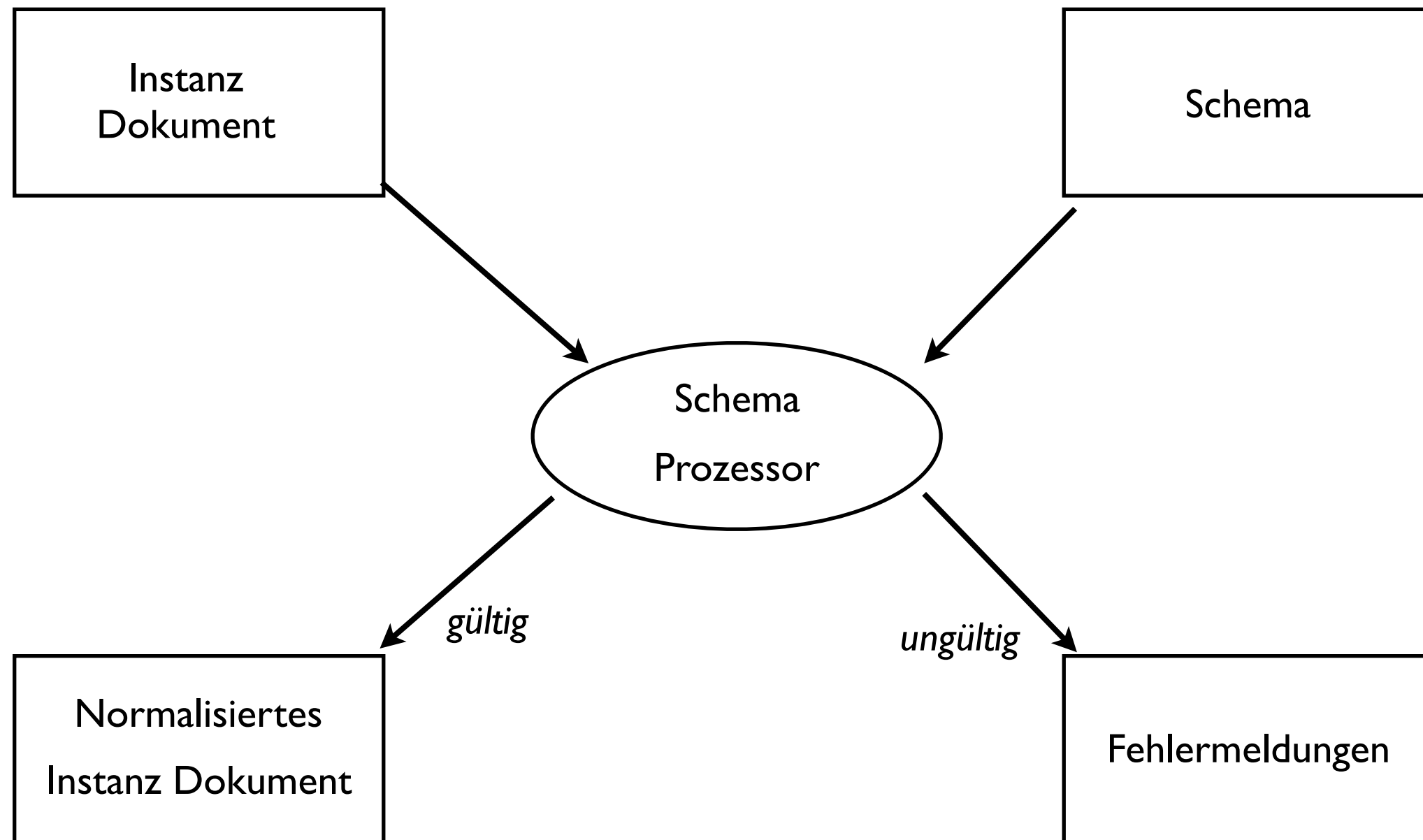
.....> ≙ Generierung zur Übersetzungszeit
oder Interpretation zur Laufzeit

→ ≙ Datenkommunikation

XML Schemasprachen

Ebene der Abstraktion	Erklärung
XML Sprache	Eine Menge von XML Dokumenten mit gemeinsamen Vokabular und Struktur
XML Schema	Eine Formale Definition der Syntax einer XML Sprache
Schemasprache	Eine Sprache zur Definition von XML Schemata

Validierung



Document Type Definition

- DTD Konzept geht auf SGML zurück
- Spezifikation ist Teil von XML 1.0
- Für datenorientierte Anwendungen zu schwach:
 - fehlendes Datentypkonzept, keine Namespaces, in XML 1.0 integriert
- Weitergehende Sprachen: XML Schema, Relax NG, ...

Bindung von DTDs an Dokumente

- `<?xml version="1.1"?>`

`<!DOCTYPE collection SYSTEM "http://www.brics.dk/ixwt/recipes.dtd">`

`<collection>`

`...`

`</collection>`

- `<!DOCTYPE html`

`PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"`

`"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
  <description>Recipes suggested by Jane Dow</description>
  <recipe id="r117">
    <title>Rhubarb Cobbler</title>
    <date>Wed, 14 Jun 95</date>
    <ingredient name="diced rhubarb" amount="2.5" unit="cup"/>
    <ingredient name="sugar" amount="2" unit="tablespoon"/>
    <ingredient name="fairly ripe banana" amount="2"/>
    <ingredient name="cinnamon" amount="0.25" unit="teaspoon"/>
    <ingredient name="nutmeg" amount="1" unit="dash"/>
    <preparation>
      <step>
        Combine all and use as cobbler, pie, or crisp.</step>
    </preparation>
    <comment>
      Rhubarb Cobbler made with bananas as the main sweetener.
      It was delicious.
    </comment>
    <nutrition calories="170" fat="28%"
      carbohydrates="58%" protein="14%"/>
    <related ref="42">Garden Quiche is also yummy</related>
  </recipe>
</collection>
```

Beispiel Rezept DTD 1/2

```
<!ELEMENT collection (description,recipe*)>
```

```
<!ELEMENT description (#PCDATA)>
```

```
<!ELEMENT recipe (title,date,ingredient*,preparation,comment?,  
    nutrition,related*)>
```

```
<!ATTLIST recipe id ID #IMPLIED>
```

```
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT date (#PCDATA)>
```

```
<!ELEMENT ingredient (ingredient*,preparation)?>
```

```
<!ATTLIST ingredient name CDATA #REQUIRED
```

```
    amount CDATA #IMPLIED
```

```
    unit CDATA #IMPLIED>
```

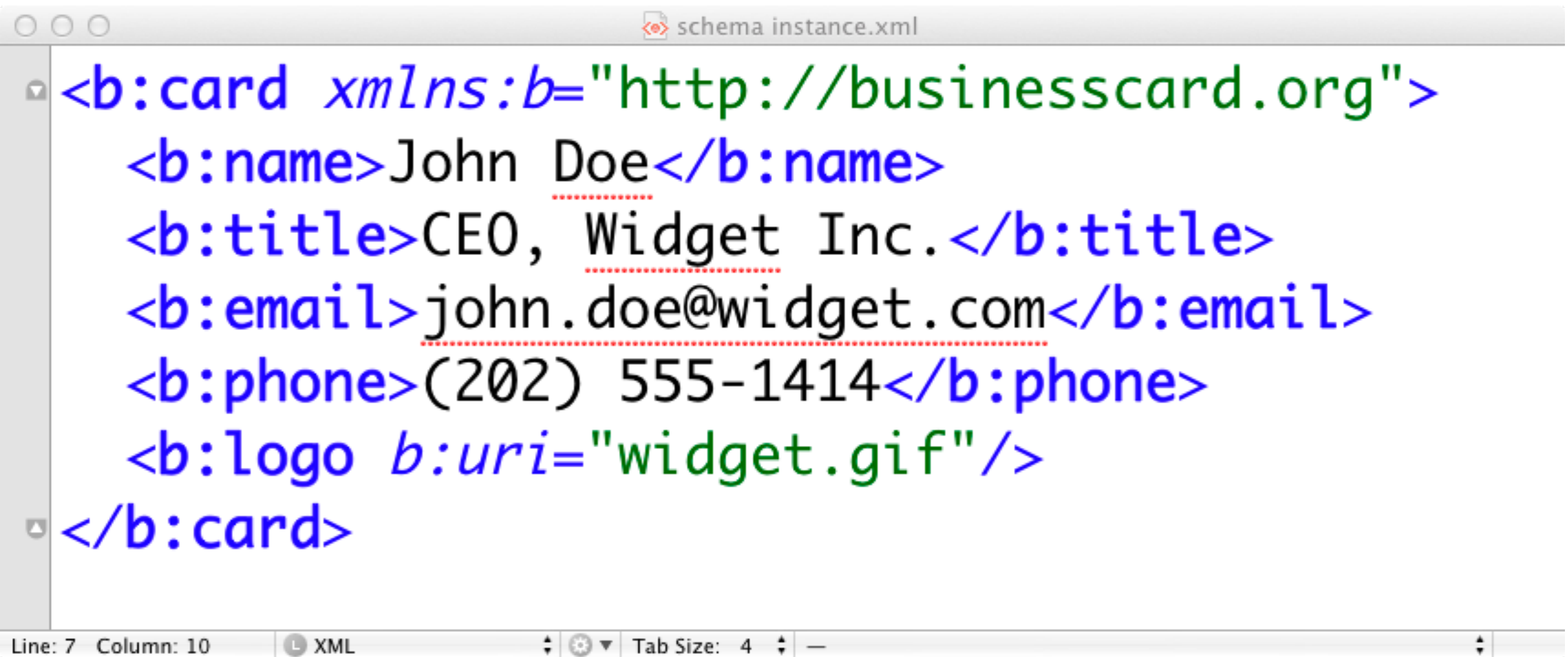
Beispiel Rezept DTD 2/2

```
<!ELEMENT preparation (step*)>
<!ELEMENT step (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT nutrition EMPTY>
<!ATTLIST nutrition calories CDATA #REQUIRED
                carbohydrates CDATA #REQUIRED
                fat CDATA #REQUIRED
                protein CDATA #REQUIRED
                alcohol CDATA #IMPLIED>
<!ELEMENT related EMPTY>
<!ATTLIST related ref IDREF #REQUIRED>
```

XML Schema: Typen und Deklarationen

Typ	Zweck
Simple Type Definition	Definiert eine Menge von Unicode Zeichenreihen
Complex Type Definition	Definiert ein Datenmodell für eine Aggregation von Simple Types
Element Deklaration	Assoziiert einen Element Namen mit einem Simple oder Complex Type
Attribut Deklaration	Assoziiert einen Attribut Namen mit einem Simple Type

Beispiel: XML Schema (Instanzdokument)



```
<b:card xmlns:b="http://businesscard.org">
  <b:name>John Doe</b:name>
  <b:title>CEO, Widget Inc.</b:title>
  <b:email>john.doe@widget.com</b:email>
  <b:phone>(202) 555-1414</b:phone>
  <b:logo b:uri="widget.gif"/>
</b:card>
```

Line: 7 Column: 10 XML Tab Size: 4

Beispiel: XML Schema (Schemadokument)

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:b="http://businesscard.org"
  targetNamespace="http://businesscard.org">
  <element name="card" type="b:card_type"/>
  <element name="name" type="string"/>
  <element name="title" type="string"/>
  <element name="email" type="string"/>
  <element name="phone" type="string"/>
  <element name="logo" type="b:logo_type"/>
  <attribute name="uri" type="anyURI"/>
  <complexType name="card_type">
    <sequence>
      <element ref="b:name"/>
      <element ref="b:title"/>
      <element ref="b:email"/>
      <element ref="b:phone" minOccurs="0"/>
      <element ref="b:logo" minOccurs="0"/>
    </sequence>
  </complexType>
  <complexType name="logo_type">
    <attribute ref="b:uri" use="required"/>
  </complexType>
</schema>
```


Verbindung von Schema und Instanz



```
<b:card xmlns:b="http://businesscard.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://businesscard.org
    business_card.xsd">
  <b:name>John Doe</b:name>
  <b:title>CEO, Widget Inc.</b:title>
  <b:email>john.doe@widget.com</b:email>
  <b:phone>(202) 555-1414</b:phone>
  <b:logo b:uri="widget.gif"/>
</b:card>
```

Line: 10 Column: 10 XML Tab Size: 4

Abstract Syntax Notation 1 (ASN.1)

- Eine Schemasprache zur Repräsentation von Daten
- U. a. in Telekommunikationsanwendungen
(Videokonferenzen, Netzmanagement, UMTS, ...)
vielfältig benutzt.
- Standardisiert in der ITU X.680 Serie
- Eine Stärke ist die Unterscheidung in die Schemasprache selbst und sogenannte *Encoding Rules*
- Wird auch verwendet um das binäre XML Format *Fast Infoset (ISO/IEC 24824-1)* zu definieren.

ASN.1 Beispiel

```
FooProtocol DEFINITIONS ::= BEGIN
```

```
    FooQuestion ::= SEQUENCE {  
        trackingNumber INTEGER,  
        question        IA5String  
    }
```

```
    FooAnswer ::= SEQUENCE {  
        questionNumber INTEGER,  
        answer          BOOLEAN  
    }
```

```
END
```

Quelle: https://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One (Abruf 16.5.2017)

Apache Thrift

- Sprache und Rahmenwerk um Code zum Austausch von Informationen zwischen Systemen zu generieren
- Geht auf Input von Facebook zurück und ist jetzt ein Open Source Projekt der Apache Foundation
- Unterscheidet wie ASN.1 zwischen Schemasprache und Encoding

Thrift Beispiel

```
enum PhoneType {  
    "HOME",  
    "WORK",  
    "MOBILE"  
    "OTHER"  
}
```

```
struct Phone {  
    1: i32      id,  
    2: string  number,  
    3: PhoneType type  
}
```

```
struct Person {  
    1: i32      id,  
    2: string  firstName,  
    3: string  lastName,  
    4: string  email,  
    5: list<Phone> phones  
}
```

```
struct Course {  
    1: i32      id,  
    2: string  number,  
    3: string  name,  
    4: Person  instructor,  
    5: string  roomNumber,  
    6: list<Person> students
```

Quelle: <http://jnb.ociweb.com/jnb/jnbJun2009.html> (Abruf 16.5.2017)

```
{
  "id": 1,
  "name": "A green door",
  "price": 12.50,
  "tags": [ "home", "green" ]
}
```

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",
  "properties": {
    "id": {
      "description": "The unique identifier for a product",
      "type": "integer"
    },
    "name": {
      "description": "Name of the product",
      "type": "string"
    },
    "price": {
      "type": "number",
      "minimum": 0,
      "exclusiveMinimum": true
    },
    "tags": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "minItems": 1,
      "uniqueItems": true
    }
  },
  "required": [ "id", "name", "price" ]
}
```

Quelle: <http://json-schema.org/example1.html>,
letzter Abruf 17.5.17

Software Engineering Aspekte von Sprachen für die Netzwerkdarstellung - Zusammenfassung

- alternativen Vorgehensweise *code first* und *communication first*
- Rolle und Bedeutung von Schemasprachen bei Entwicklung verteilter Systeme
- wesentliche Eigenschaften und Sprachkonstrukte von XML Schemasprachen
- Alternative Netzwerkdarstellungen