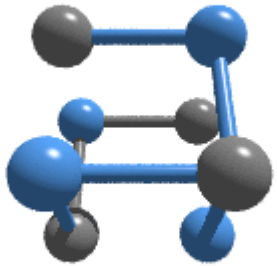


Präsenzveranstaltung zur E-Learning-Veranstaltung

Einführung in XML

Sommersemester 2008

Prof. Dr. Klaus-Peter Fähnrich
Dr. Maik Thränert

Summer School Model-Driven Software Engineering

22. bis 26. September 2008 als ganztägige Blockveranstaltung

- 2 Tage theoretische Einführung in die generative und modellgetriebene Software-Entwicklung
- 3 Tage praxisorientierter Power-Workshop mit erfahrenen Beratern der itemis AG (http://www.itemis.de/) am Beispiel des Generatorframeworks *openArchitectureWare* (http://www.openarchitectureware.org/ und http://www.eclipse.org/gmt/oaw/).
- Teilnehmerzahl: 20 Studenten
- Anmeldung per E-Mail an haucke@wifa.uni-leipzig.de **und** gebauer@informatik.uni-leipzig.de.

InfAI Institut für Angewandte Informatik

itemis Software | Consulting | Coaching

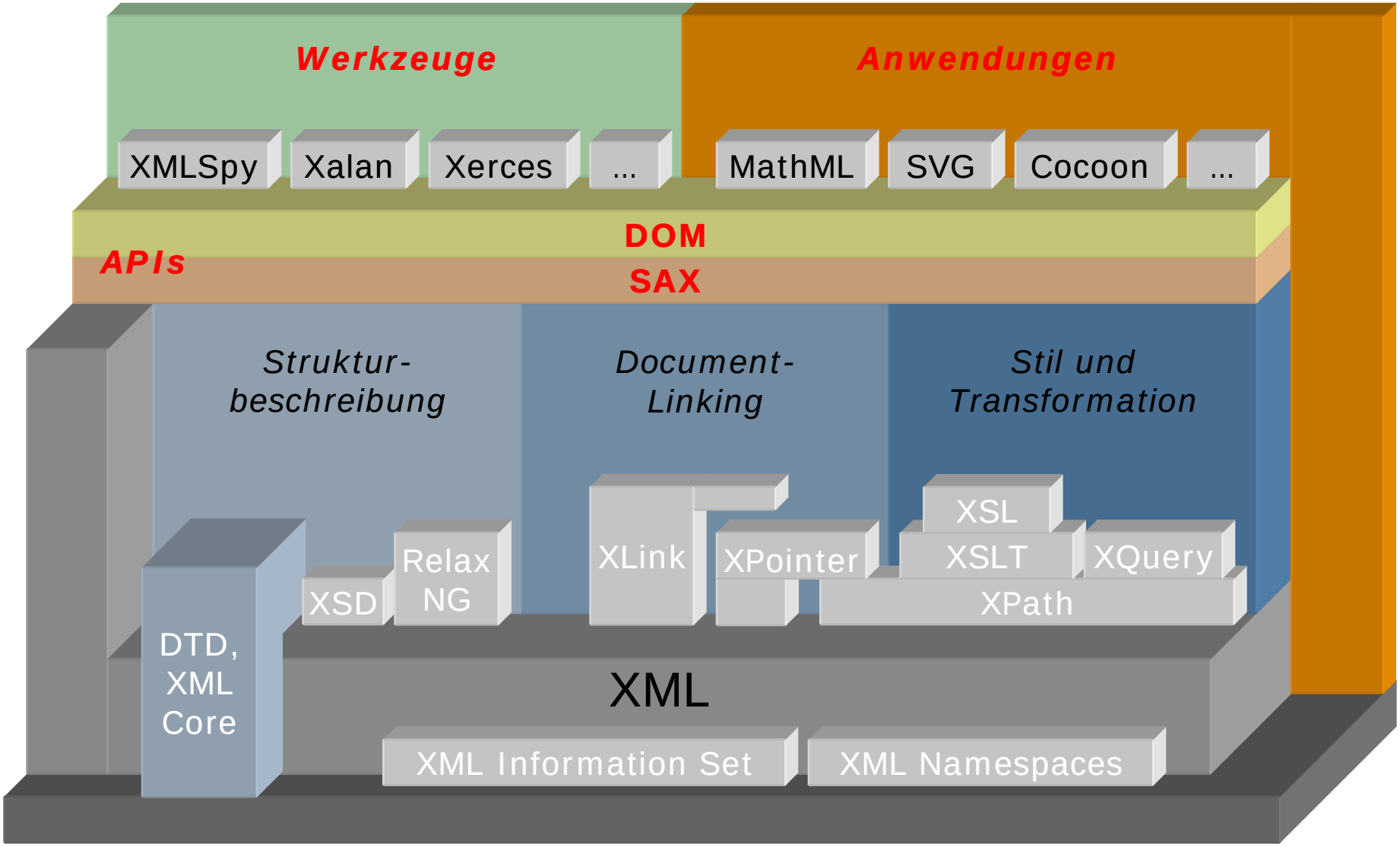
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Agenda

- **Zusammenfassung des WB Programmierschnittstellen und Werkzeuge**
- Fragen?
- Weiterer Ablauf

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Gliederung der Vorlesung



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Lernziele

- Zwei grundsätzliche Verarbeitungsmethoden von XML-Dokumenten kennen,
- Geschichte von SAX und DOM kennen,
- Einfache Programmieraufgaben lösen können,
- Werkzeugkategorien zur Bearbeitung von XML-Dokumenten kennen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

1. Programmierschnittstellen
    1. **Grundsätzliches,**
    2. SAX,
    3. DOM,
    4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
    1. Strukturbeschreibung,
    2. Dokumentenerstellung,
    3. Dokumentenbetrachtung,
    4. Datenhaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.1 Grundsätzliches

Bisher haben wir XML-Dokumente mit diversen Tools bearbeitet, ohne zu betrachten, wie genau Programme die XML-Dokumente verarbeiten, auswerten oder auch erzeugen können.

Hauptsächlich werden dafür 2 Techniken verwendet:

- DOM (Document Object Model; W3C-Empfehlung)
- SAX (Simple API for XML; De-facto-Standard)

Wichtig: Der programmiertechnische Zugriff soll leicht zu realisieren sein!

Daher erlauben beide Techniken die Arbeit auf einem logischen Modell des Dokumentes.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.1 Grundsätzliches

Ereignisgesteuerte Verarbeitung (bei SAX)

- Dokument wird sequentiell abgearbeitet (Daten "strömen vorbei")
  - Bei bestimmten Ereignissen (z. B. Finden eines Start-Tags) Aufruf entsprechender Methoden (Callback)
- Vorteil: Dokument kann in einem Durchgang zeit- und speichereffizient abgearbeitet werden
- Nachteil: keine Sprünge oder gezielte Bearbeitung ausgewählter Dokumentteile

1.1 Grundsätzliches

Baumbasierte Verarbeitung (bei DOM)

- Aufbau eines Dokumentbaumes im Speicher, die Knoten sind Objekte im OO-Sinne mit Methoden, Attributen usw.
- Ermöglicht freie Navigation im Baum und z. B. Einfügen von Knoten
- Nachteil: gesamter Dokumentbaum muss im Speicher gehalten werden

Es ist auch eine Kombination der Techniken möglich:

- z. B. Vorbereitung eines Dokumentes durch SAX für die weitere Verarbeitung mit DOM

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

1. Programmierschnittstellen
    1. Grundsätzliches,
    2. **SAX,**
    3. DOM,
    4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
    1. Strukturbeschreibung,
    2. Dokumentenerstellung,
    3. Dokumentenbetrachtung,
    4. Datenhaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX: Historie

- Dezember 1997: Peter Murray-Rust, Tim Bray und David Megginson beginnen in der XML-DEV-Mailingliste eine öffentliche Diskussion über eine ereignisgesteuerte API für XML
- Jon Bosak erlaubt die Nutzung seiner `xml.org`-Domain für das SAX-Paket: `org.xml.sax`
- Januar 1998: erstes Pre-Release
- Mai 1998: nach diversen weiteren Pre-Releases veröffentlicht die XML-DEV-Community SAX 1.0
- Januar 2002: SAX 2.0.1 wird veröffentlicht
- 27-April 2004: SAX 2.0.2

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX : Übersicht

- Open-Source-Programmierschnittstelle zum Zugriff auf XML- und HTML-Dokumente:
  - Ursprünglich nur als API für Java entwickelt

  ➔ keine allgemeine Beschreibung,

  - Mittlerweile auch Implementationen für andere Programmiersprachen (C++, Perl, Python, Visual Basic, ...)

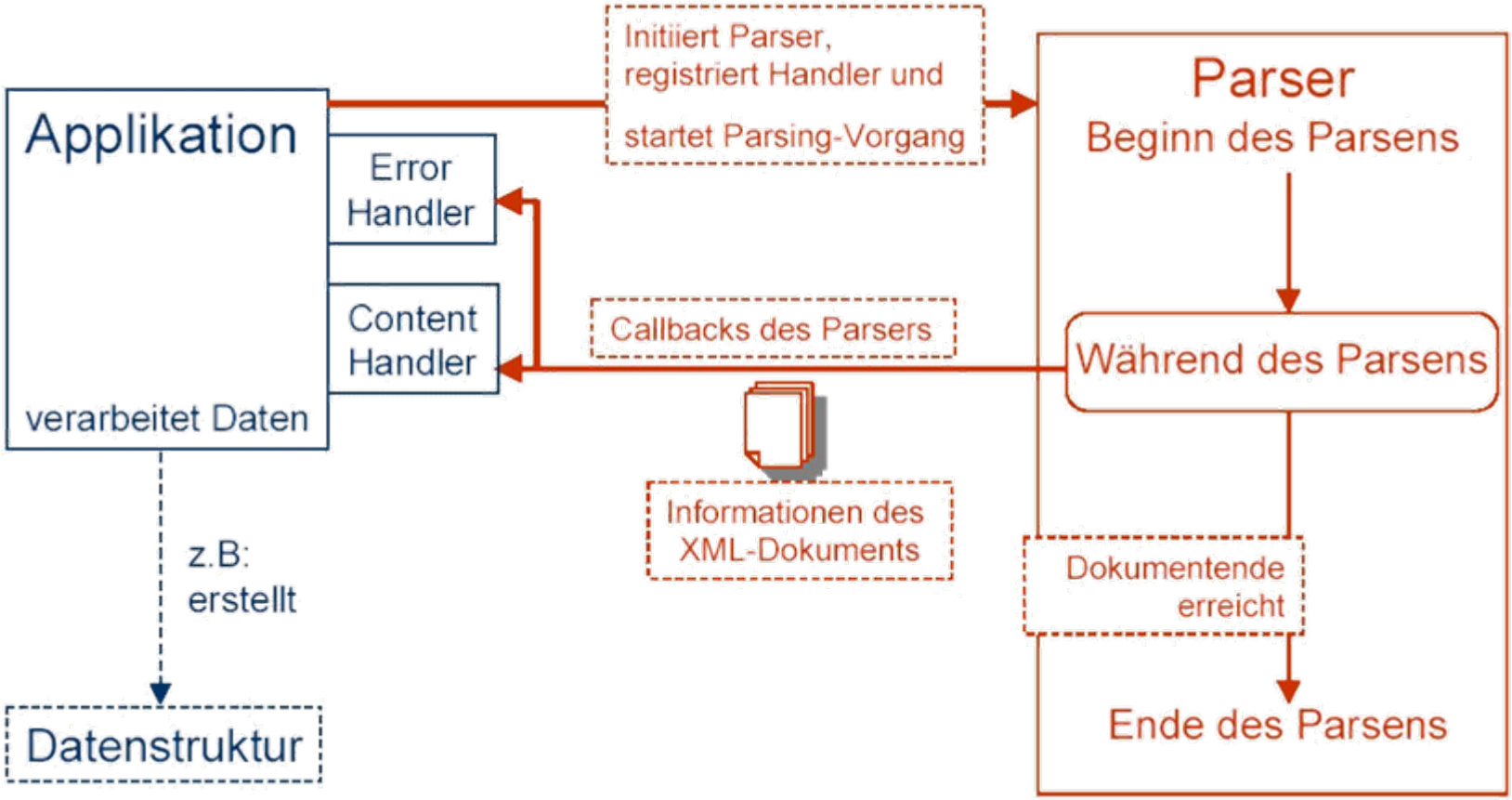
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX : Übersicht

- Prinzip:
  - Ereignisorientierter Zugriff auf XML-Dokumente

1. Anwendung "abonniert" Knoten eines bestimmten Typs
2. Parsen des Dokuments beginnt
3. trifft der Parser auf einen bestimmten "abonnierten" Knotentyp, so wird dies der Anwendung mittels Callback mitgeteilt und eine vorgegebene Funktion aufgerufen

1.2 SAX



rot: SAX; blau: Anwendung

Quelle: TU-Dresden, Informatik

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX – Beispiel Listing (1)

```
import java.io.FileReader;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public class MySAXApp extends DefaultHandler {
    public static void main (String args[]) throws Exception {
        XMLReader xr = XMLReaderFactory.createXMLReader ();
        MySAXApp handler = new MySAXApp();
        xr.setContentHandler(handler);
        xr.setErrorHandler(handler);
        for (int i = 0; i < args.length; i++) {// Parse each file
                        //provided on the command line.
                FileReader r = new FileReader(args[i]);
                xr.parse(new InputSource(r));
        }
    }

    public MySAXApp (){
        super();
    }
```

demo/sax/MySaxApp.bat

Quelle: www.saxproject.org

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX – Beispiel Listing (2)

```
public void startDocument () {
     System.out.println("Start document");
}

public void endDocument () {
     System.out.println("End document");
}

public void startElement (String uri, String name, String qName,
                                  Attributes atts) {
     if ("".equals (uri))
               System.out.println("Start element: " + qName);
     else
               System.out.println("Start element: {" + uri + "}" + name);
}

public void endElement (String uri, String name, String qName) {
     if ("".equals (uri))
               System.out.println("End element: " + qName);
     else
               System.out.println("End element:   {" + uri + "}" + name);
}
```

Quelle: www.saxproject.org

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX – Beispiel Listing (3)

```
    public void characters (char ch[], int start, int length) {
        System.out.print("Characters:    \"");
        for (int i = start; i < start + length; i++) {
                switch (ch[i]) {
                        case '\\': System.out.print("\\\\"); break;
                        case '"': System.out.print("\\\""); break;
                        case '\n': System.out.print("\\n"); break;
                        case '\r': System.out.print("\\r"); break;
                        case '\t': System.out.print("\\t"); break;
                        default: System.out.print(ch[i]); break;
                }
        }
        System.out.print("\"\n");
    }
}
```

Quelle: www.saxproject.org

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX – Beispiel (XML-Dokument)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AdressDB SYSTEM "beispiel1.dtd">
<AdressDB>
   <Adresse>
        <Person Anrede="Frau">
               <Vorname>Eva</Vorname>
               <Name>Mustermann</Name>
        </Person>
        <Strasse Nummer="4">Beispielstrasse</Strasse>
        <Ort PLZ="54783">Musterstadt</Ort>
   </Adresse>
   <Adresse>
        <Firma>Universitaet Leipzig</Firma>
        <Strasse Nummer="10-11">Augustusplatz</Strasse>
        <Ort PLZ="04109">Leipzig</Ort>
   </Adresse>
</AdressDB>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.2 SAX – Beispiel (Ausgabe)

```
Start document
Start element: AdressDB
Start element: Adresse
Start element: Person
Start element: Vorname
Characters:    "Eva"
End element: Vorname
Start element: Name
Characters:    "Mustermann"
End element: Name
End element: Person
Start element: Strasse
Characters:
   "Beispielstrasse"
End element: Strasse
Start element: Ort
Characters:    "Musterstadt"
End element: Ort
End element: Adresse
```

```
Start element: Adresse
Start element: Firma
Characters:    "Universitaet
   Leipzig"
End element: Firma
Start element: Strasse
Characters:    "Augustusplatz"
End element: Strasse
Start element: Ort
Characters:    "Leipzig"
End element: Ort
End element: Adresse
End element: AdressDB
End document
```

1.2 SAX

Vorteile:

- einfach, effizient bzgl. Speicher und Rechenzeit
- geeignet für große Datenmengen
- geringer Overhead bei Verarbeitung eines geringen Anteils der Daten

Nachteile:

- kein Mechanismus zum Schreiben von XML spezifiziert
- keine freie Navigation im Dokument

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

1. Programmierschnittstellen
   1. Grundsätzliches,
   2. SAX,
   3. **DOM,**
   4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
   1. Strukturbeschreibung,
   2. Dokumentenerstellung,
   3. Dokumentenbetrachtung,
   4. Datenhaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM - Historie

- 1995 führte Netscape die Sprache JavaScript ein,
- Diese wurde später von der ECMA als Industriestandard unter dem Namen "ECMAScript" festgelegt,
- Während des "Browserkrieges" zwischen Netscape und Microsoft immer neue Features, unter anderem immer neue JavaScript-Versionen,
- Um dem "Wildwuchs" bei den Sprachversionen Einhalt zu gebieten, wurde das W3C eingeschaltet,
- W3C erarbeitete jedoch keinen konkreten Standard für JavaScript, sondern nur ein Modell für Objekte eines Dokuments

➔ Document Object Model (DOM) in Version 1.0

Quelle: http://selfhtml.teamone.de/javascript/intro.htm

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM

- Programmierschnittstelle zum Zugriff auf XML- und HTML-Dokumente:
  - Vom World Wide Web Consortium spezifiziert,
  - Die Schnittstellen sind sprachneutral und plattformunabhängig in IDL (Interface Definition Language) beschrieben,
  - DOM definiert Standardmethoden zum Einfügen, Löschen, Ersetzen von Knoten und zur Navigation zwischen den Knoten,
  - Zum Einsatz kommen dann konkrete Implementierungen, die so genannten Parser

1.3 DOM

Auszug aus der Definition (IDL):

```
 /**
     *
     * method validate
     *
     * This method validates the given Document against its DTD. Returns true if
     * the Document is valid and throws a DOMException otherwise.
     * The following DOMException may be thrown
     *       IO_ERR :   Raised if any kind of IOException is thrown processing
     *                  the Document
     *       PARSE_ERR: Raised whenever a SAXException is thrown (e.g. Document
     *                  is invalid)
     *
*/
    boolean validate(in Document doc) raises (DOMException);
/**
     *
     * method writeDocument
     *   ... */
    void writeDocument(in Document doc, in DOMString url) raises (DOMException);
```

1.3 DOM

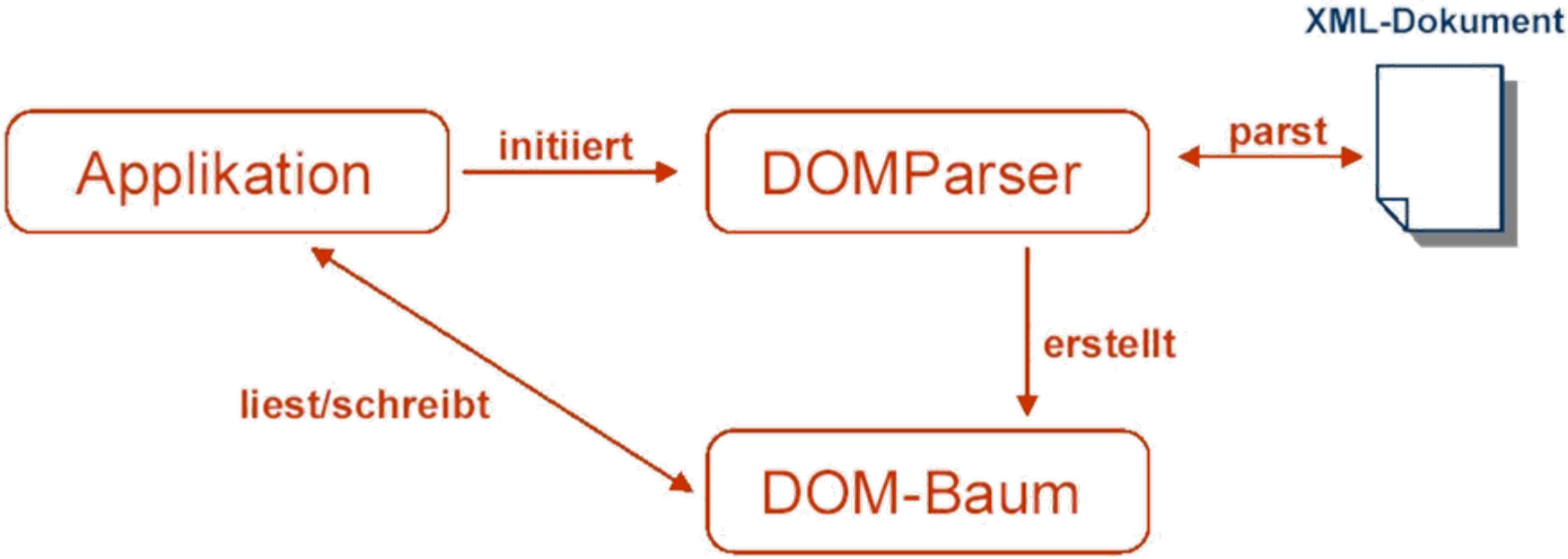
- Ansatz:
  - Erstellung eines baumförmigen Modells der Dokumentstruktur (XML oder HTML)
  - Auswertung und Manipulation dieser Struktur über Methoden der Objekte im Baum
  - Sammlung von Klassen ("Interfaces"), die im Baum verwendet werden
- Nachteile dieses Ansatzes:
  - Hoher Speicherbedarf, da das ganze Dokument eingelesen wird

1.3 DOM

Genereller Ablauf der Verarbeitung von XML-Dokumenten mit DOM:

- Kreieren einer Parser-Instanz
- Parsen des Dokumentes → Aufbau des Dokumentbaums
- Verarbeitung des DOM-Baumes

1.3 DOM



Quelle: TU-Dresden, Informatik

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM - Beispiel (1)

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.SAXException;
import java.io.*;

public class SimpleDOM {
  public static void main(String args[]) {
    try {   // Get the Document object
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder docBuilder = factory.newDocumentBuilder();
      Document doc = docBuilder.parse(args[0]);
      printDoc(doc, System.out);   // And print it to stdout
    }
    catch (ParserConfigurationException pce) {
      System.err.println("Error in DOM Parser System Properties: "
                         + pce.getMessage());
    }
    catch (SAXException se) {
      System.err.println("Parse error: " + se.getMessage());
    }
    catch (IOException ie) {
      System.err.println("I/O Exception: " + ie.getMessage());
    }
  }

  public static void printDoc(Document doc, PrintStream out) {
    printNode(doc, "", out);
  }

  private static void printAttributes(NamedNodeMap m, PrintStream out) {

    for (int i = 0; i < m.getLength(); i++) {
      out.print(" " + ((Attr) m.item(i)).getName() + "='"
                    + ((Attr) m.item(i)).getValue() + "'");
    }
  }
}
```

demo/dom/startMyDomApp.bat

Quelle: www.inf.uni-konstanz.de

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM - Beispiel (2)

```
private static void printNode(Node n, String indent, PrintStream out) {
    NodeList children;
    switch (n.getNodeType()) {
      case Node.CDATA_SECTION_NODE:
      case Node.TEXT_NODE: // print text
        String trimmedString = n.getNodeValue().trim();
        if (trimmedString.length() != 0)
          out.println(indent + trimmedString);
        break;
      case Node.DOCUMENT_NODE:
        out.println("<?xml version='1.0' encoding='iso-8859-1'?>");
        out.println();
        children = n.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
          printNode(children.item(i), indent, out);
        } break;
      case Node.ELEMENT_NODE: // print element
        out.print(indent + "<" + n.getNodeName());
        printAttributes(n.getAttributes(), out);
        out.println(">");
        children = n.getChildNodes();
        for (int i = 0; i < children.getLength(); i++) {
          printNode(children.item(i), indent + "  ", out);
        }
        out.println(indent + "</" + n.getNodeName() + ">");
        break;
      default: System.err.println("Warning: Unsupported Node Type!");
    }
  }
```

Quelle: www.inf.uni-konstanz.de

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM - Beispiel (3) – XML-Dokument

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AdressDB SYSTEM "beispiel1.dtd">
<AdressDB>
   <Adresse>
        <Person Anrede="Frau">
               <Vorname>Eva</Vorname>
               <Name>Mustermann</Name>
        </Person>
        <Strasse Nummer="4">Beispielstrasse</Strasse>
        <Ort PLZ="54783">Musterstadt</Ort>
   </Adresse>
   <Adresse>
        <Firma>Universitaet Leipzig</Firma>
        <Strasse Nummer="10-11">Augustusplatz</Strasse>
        <Ort PLZ="04109">Leipzig</Ort>
   </Adresse>
</AdressDB>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.3 DOM - Beispiel (4) - Ausgabe

```
<?xml version='1.0' encoding='iso-8859-1'?>

Warning: Unsupported Node Type!
<AdressDB>
  <Adresse>
    <Person Anrede='Frau'>
      <Vorname>
        Eva
      </Vorname>
      <Name>
        Mustermann
      </Name>
    </Person>
    <Strasse Nummer='4'>
      Beispielstrasse
    </Strasse>
    <Ort PLZ='54783'>
      Musterstadt
    </Ort>
  </Adresse>...
```

Vorteile:

- einheitlicher Zugriff in verschiedenen Sprachen,
- gezielter und schneller Zugriff,
- geringerer Programmieraufwand als bei SAX,
- besser geeignet zur Generierung dynamischen Inhalts

Nachteile:

- Schreiben von XML nicht im API selber enthalten,
  - Erweiterung: DOM Level 3 Load- and Save-Specification
- automatisches Durchschreiten des Baumes nicht enthalten,
- konsequente Baumorientierung umständlich,
- zunächst hoher Overhead für Baumerstellung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

1. Programmierschnittstellen
    1. Grundsätzliches,
    2. SAX,
    3. DOM,
    4. **Vergleich SAX – DOM**

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
    1. Strukturbeschreibung,
    2. Dokumentenerstellung,
    3. Dokumentenbetrachtung,
    4. Datenhaltung

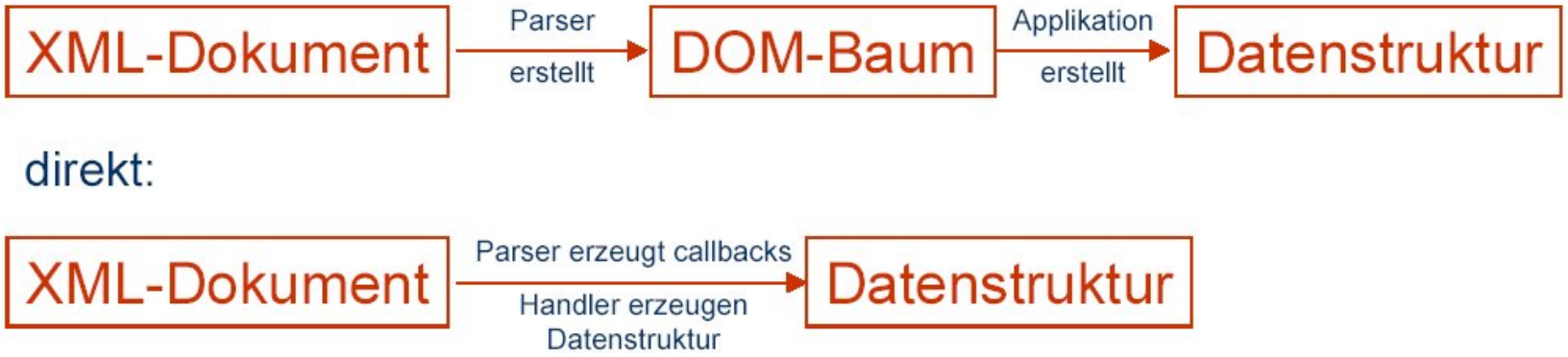
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.4 Vergleich SAX-DOM

	SAX	DOM
Pro	• zeit- und speichereffizient • Parsen jederzeit abbrechbar • gezielte Extraktion von Informationen	• einfacher • gezielter Zugriff auf Elemente • einfache Suche • flexiblere Verarbeitung
Contra	• keine Nachbearbeitungs-möglichkeit • schwierige Navigation • wenig Funktionalität	• Zeit- und Speicherverbrauch • Probleme bei sehr großen Dokumenten

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

1.4 Vergleich SAX-DOM



Fazit:

DOM und SAX sind nicht direkt miteinander konkurrierende Standards,

Sie ergänzen sich beide, z. B. verwenden die meisten DOM-Implementierungen intern einen SAX-Parser.

Quelle: TU-Dresden, Informatik

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

1. Programmierschnittstellen
    1. Grundsätzliches,
    2. SAX,
    3. DOM,
    4. Vergleich SAX – DOM

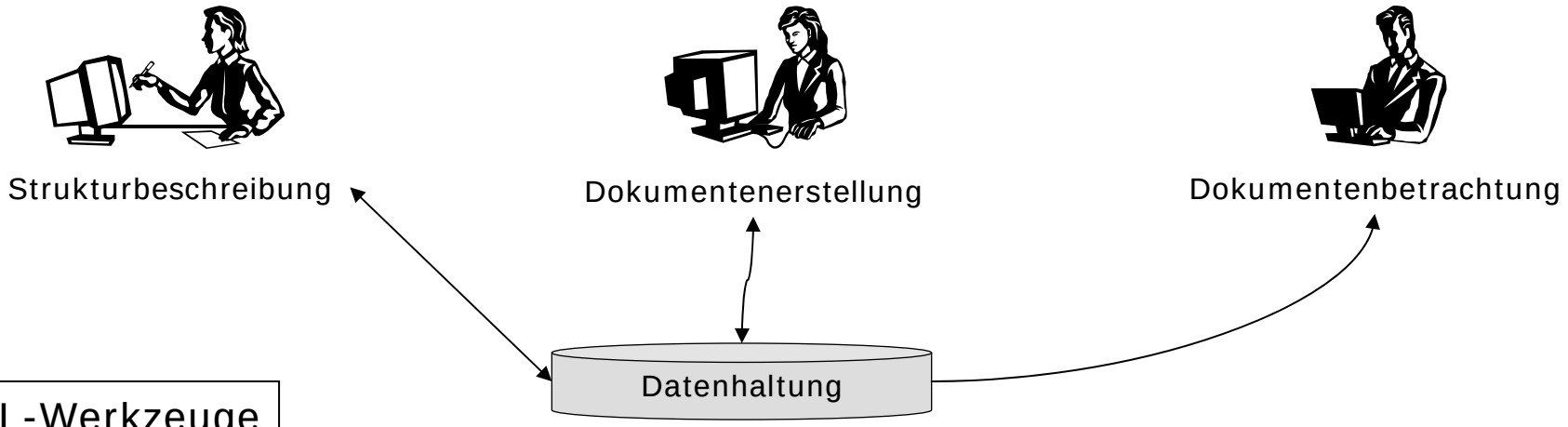
2. Werkzeuge zur Bearbeitung von XML-Dokumenten

1. Strukturbeschreibung,
2. Dokumentenerstellung,
3. Dokumentenbetrachtung,
4. Datenhaltung

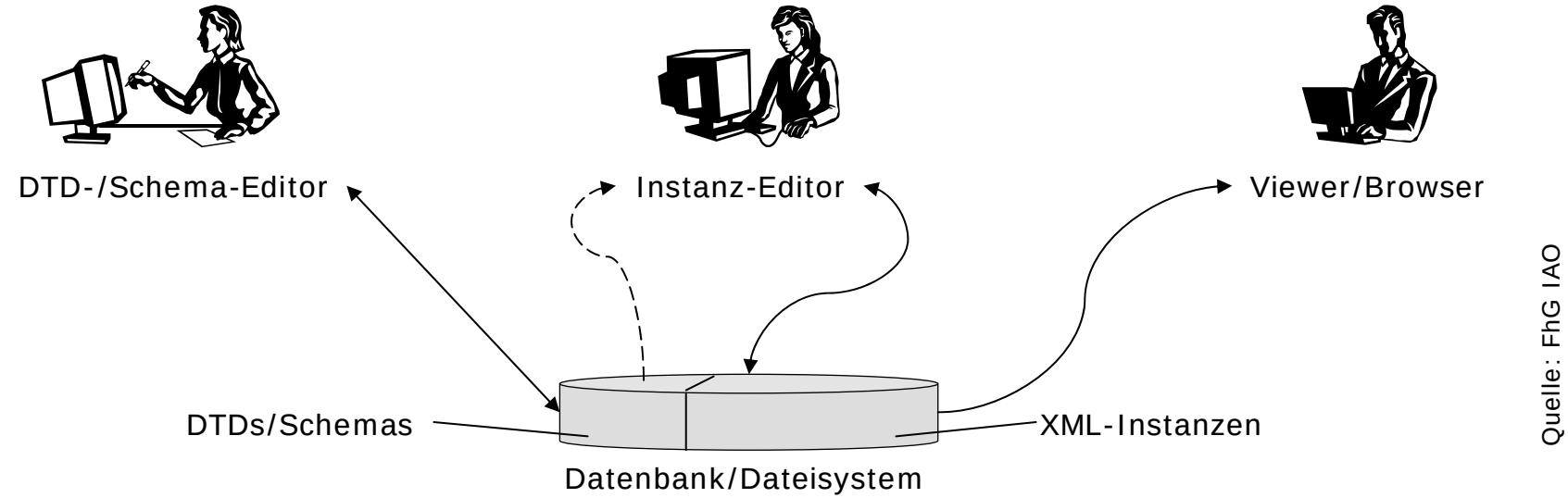
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Werkzeugunterstützung / Werkzeugkategorien

Werkzeugkategorien zur Dokumentbearbeitung



XML-Werkzeuge



Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

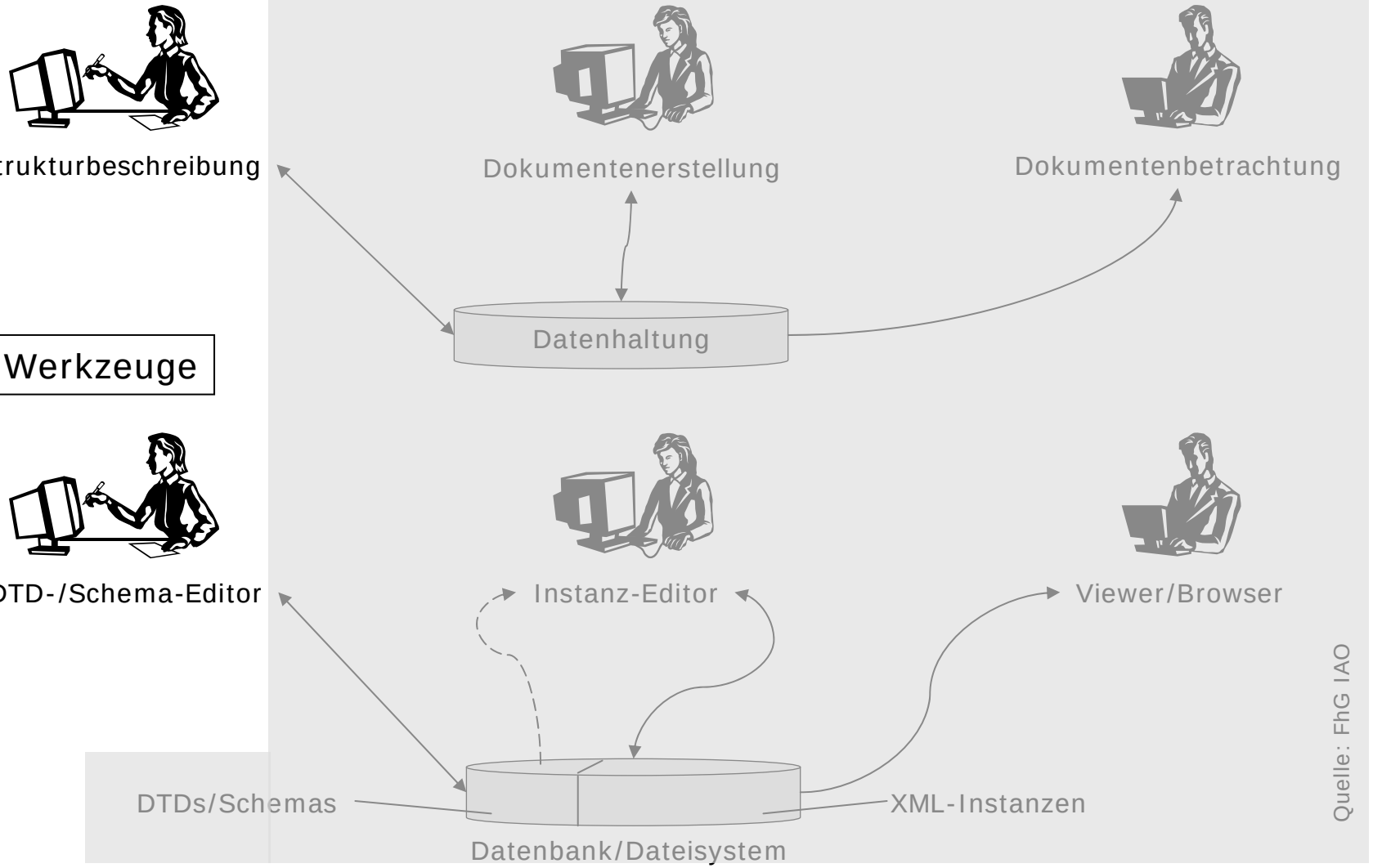
1. Programmierschnittstellen
   1. Grundsätzliches,
   2. SAX,
   3. DOM,
   4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
   1. **Strukturbeschreibung,**
   2. Dokumentenerstellung,
   3. Dokumentenbetrachtung,
   4. Datenhaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Werkzeugunterstützung / Werkzeugkategorien

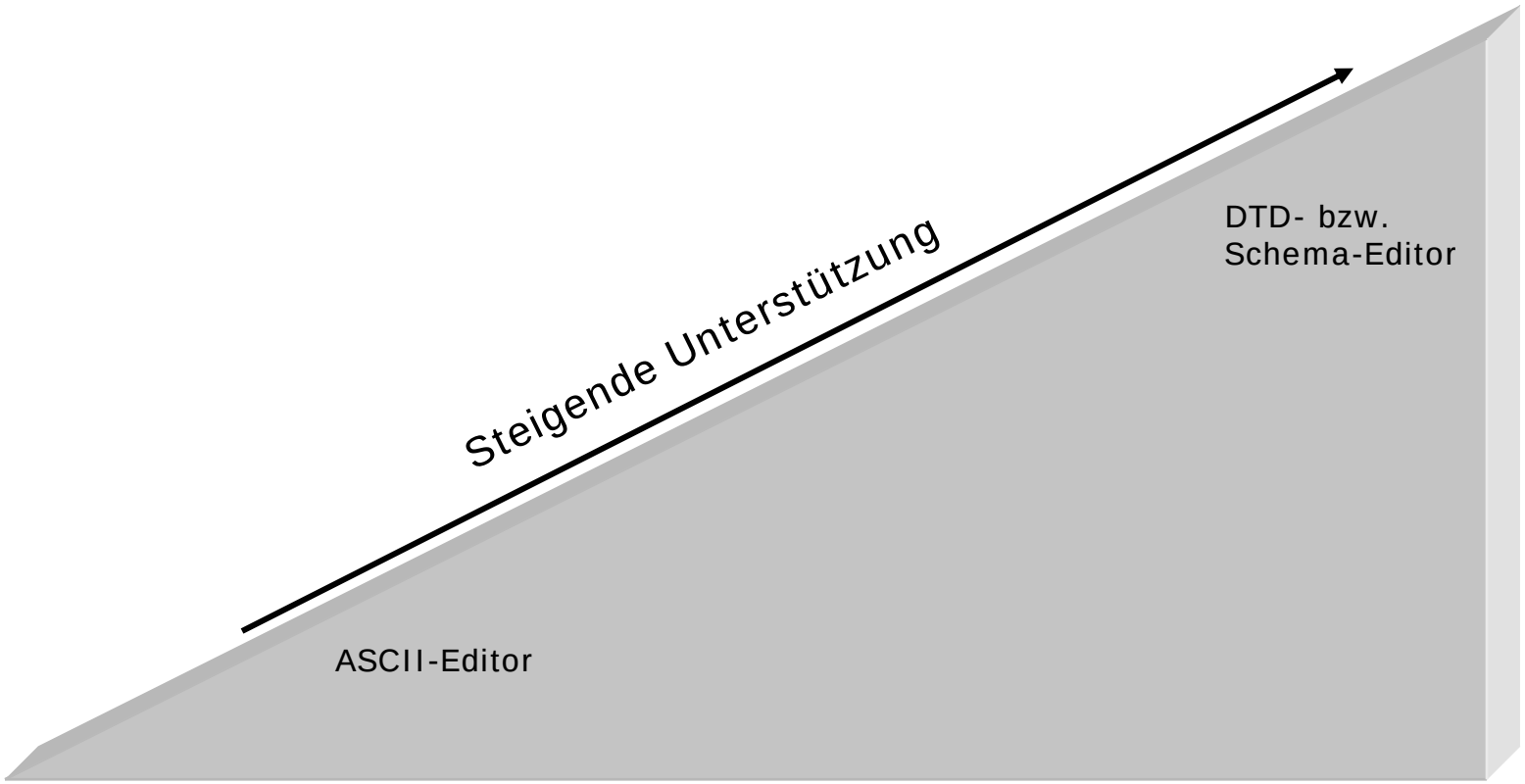
Werkzeugkategorien zur Dokumentbearbeitung



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.1 Strukturbeschreibungswerkzeuge

- DTD-/Schema-Editoren



Quelle: FhG IAO

2.1 Strukturbeschreibungswerkzeuge

- ASCII-Editor:
  - Vorteile:
    - Relativ einfach zu bedienen,
    - Verhältnismäßig preiswert,
    - Schnelle Ergebnisse bei kleinen DTDs,
  - Nachteile:
    - Keine eingebaute Erkennung von Syntaxfehlern,
    - Erstellung sehr großer DTDs und Schemas sehr fehleranfällig

- DTD-/Schema-Editoren:
  - Vorteile:
    - Angepasste grafische Benutzeroberfläche,
    - Teilweise Entwicklung durch "drag and drop",
    - Entwickler kann am Quelltext oder grafisch arbeiten,
    - Gute Strukturübersicht
  - Nachteile:
    - Gegenüber ASCII-Editoren sehr teuer

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.1 Strukturbeschreibungswerkzeuge

- ASCII-Editor:

library.dtd - Editor

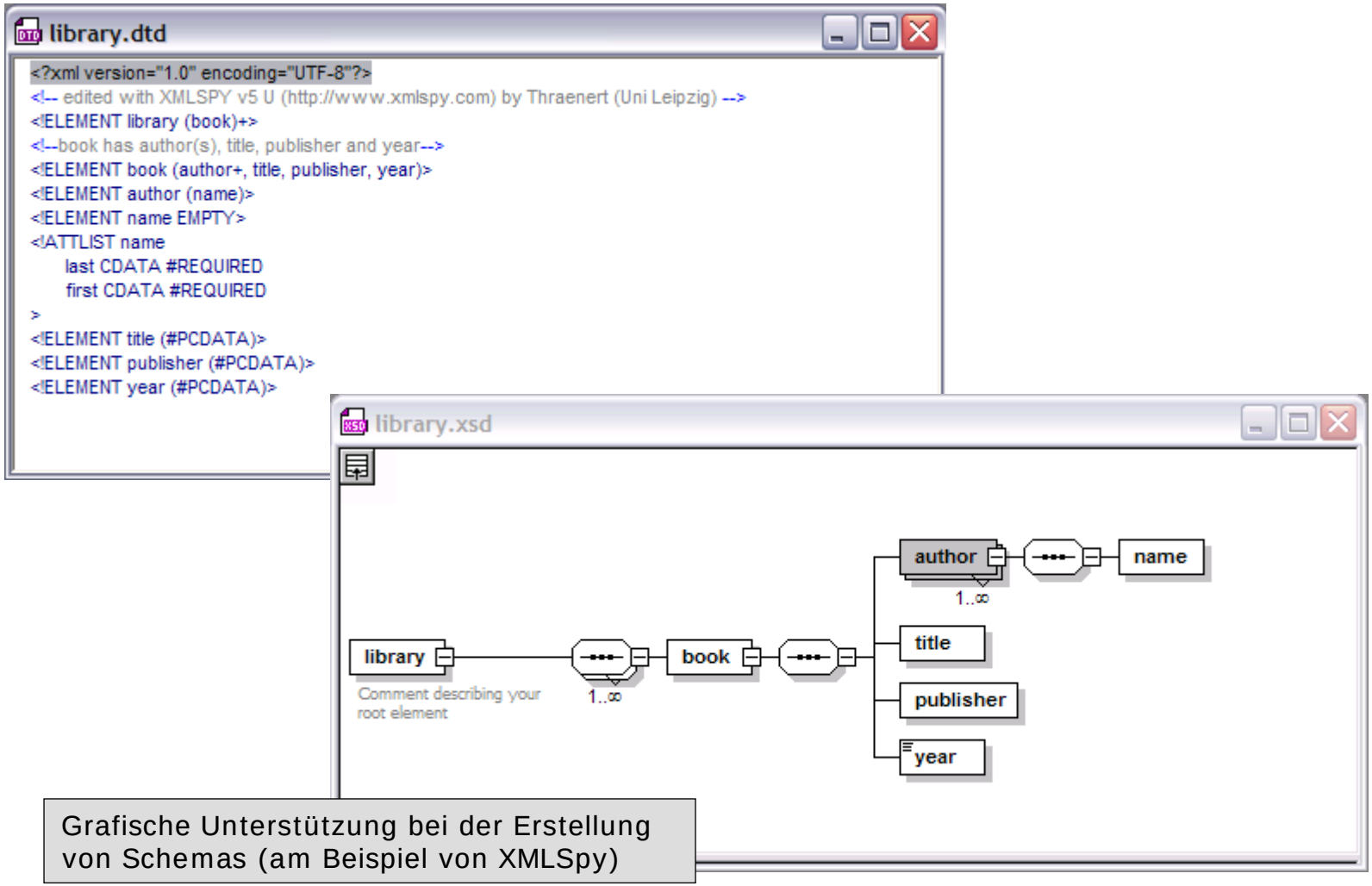
Datei Bearbeiten Format Ansicht ?

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Thraenert (Uni Leipzig) -->
<!ELEMENT library (book)+>
<!--book has author(s), title, publisher and year-->
<!ELEMENT book (author+, title, publisher, year)>
<!ELEMENT author (name)>
<!ELEMENT name EMPTY>
<!ATTLIST name
        last CDATA #REQUIRED
        first CDATA #REQUIRED
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT year (#PCDATA)>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.1 Strukturbeschreibungswerkzeuge

- DTD-/Schema-Editoren:



Grafische Unterstützung bei der Erstellung von Schemas (am Beispiel von XMLSpy)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

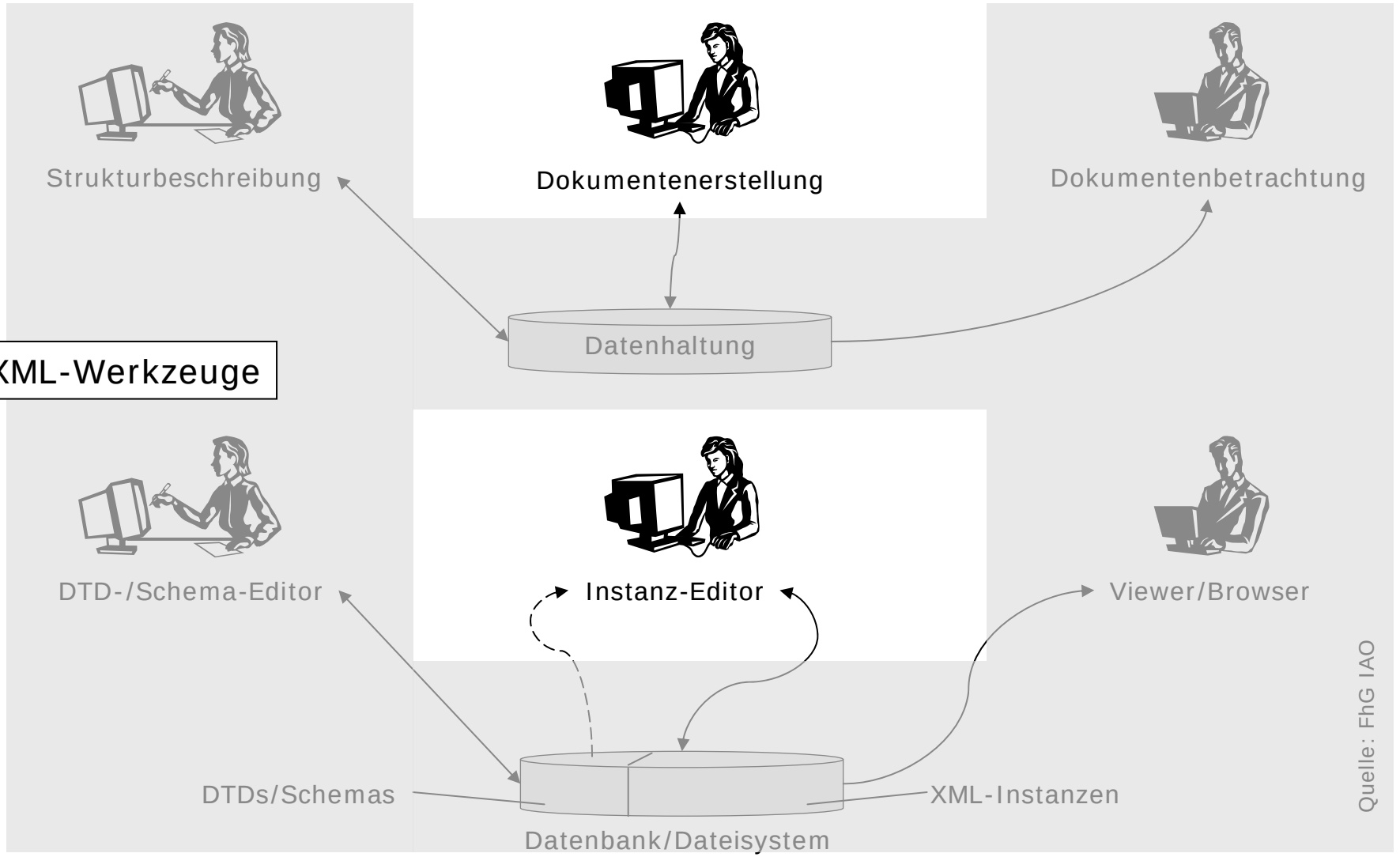
Übersicht

1. Programmierschnittstellen
   1. Grundsätzliches,
   2. SAX,
   3. DOM,
   4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
   1. Strukturbeschreibung,
   2. **Dokumentenerstellung,**
   3. Dokumentenbetrachtung,
   4. Datenhaltung

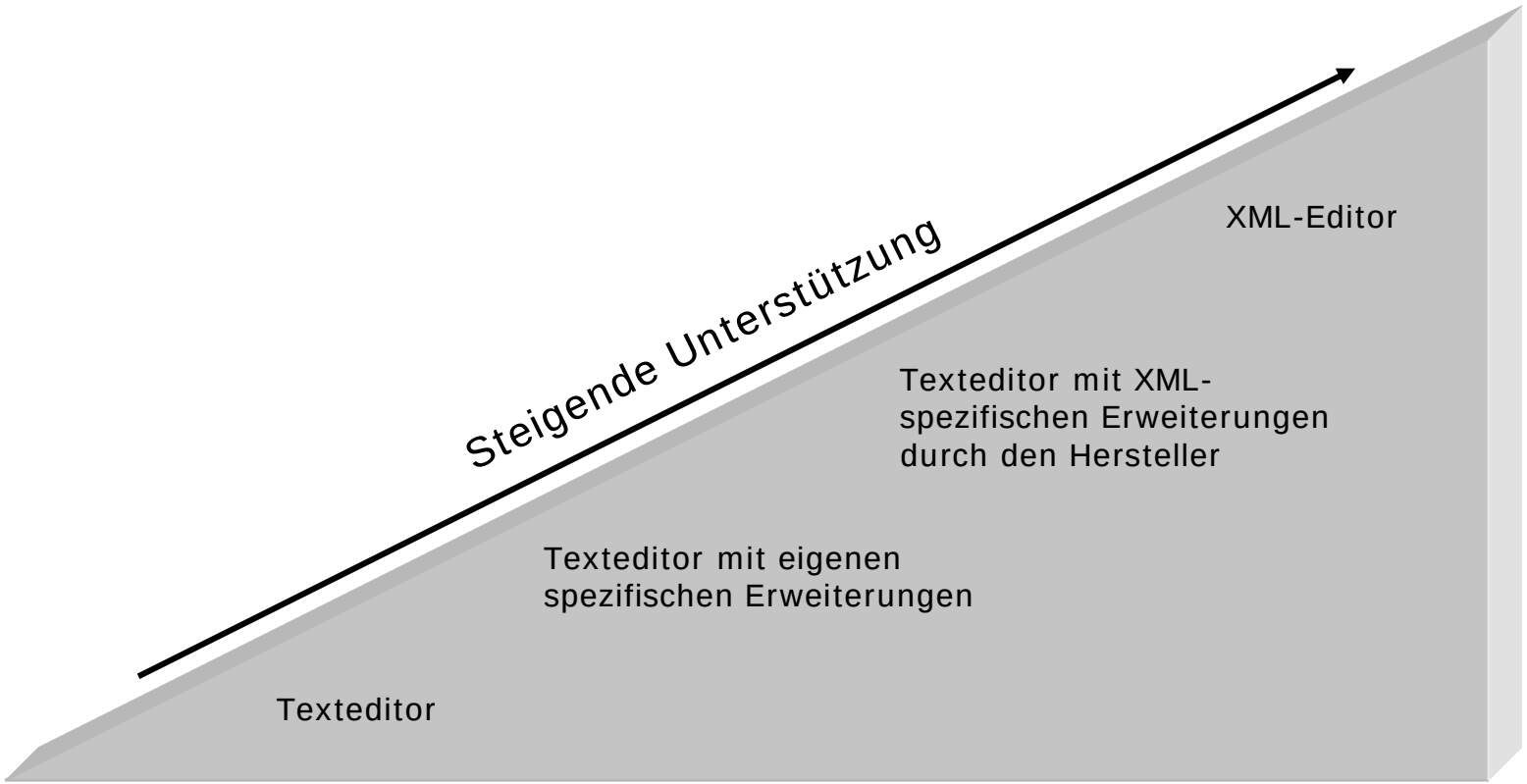
Werkzeugunterstützung / Werkzeugkategorien

Werkzeugkategorien zur Dokumentbearbeitung



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.2 Dokumentenerstellung



Quelle: FhG IAO

2.2 Dokumentenerstellung

- Texteditor:
  - Vorteile:
    - Anwender ist die Bedienung i. A. schon gewöhnt,
    - Verbreitete Editoren bieten dem Nutzer mehr oder weniger großen Komfort und viele Freiheiten,
    - Bei Instanzen zu kleinen DTDs gute Ergebnisse erzielbar
  - Nachteile:
    - Keine Abbildung von XML-Spezifika,
    - Strukturfehler werden nicht erkannt
      - Reihenfolge von Elementen,
      - Kardinalität von Elementen,
      - Verwendung vorgeschriebener Elemente,
    - Instanzerstellung zu sehr komplexen DTDs und Schemas ist sehr fehleranfällig,
    - Keine Erkennung von Syntaxfehlern

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.2 Dokumentenerstellung

- Texteditor:

books.xml - Editor

Datei Bearbeiten Format Ansicht ?

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Thraenert (Uni Leipzig) -->
<!DOCTYPE library SYSTEM "C:\Dokumente und Einstellungen\MT\Desktop\eb1\samples\library.dtd">
<library>
        <book>
                <author>
                        <name last="Behme" first="Henning"/>
                </author>
                <author>
                        <name last="Mintert" first="Stefan"/>
                </author>
                <title>XML in der Praxis</title>
                <publisher>Addison-Wesley</publisher>
                <year>2000</year>
        </book>
        <book>
                <author>
                        <name last="Hansen" first="Hans"/>
                </author>
                <author>
                        <name last="Fritzen" first="Fritz"/>
                </author>
                <title>Der verflixte Hans</title>
                <publisher>Hans-Verlag</publisher>
                <year>1977</year>
        </book>
</library>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.2 Dokumentenerstellung

- Texteditor mit eigenen spezifischen Erweiterungen:
  - Vorteile:
    - Anwender kann mit gewohntem Werkzeug arbeiten,
    - Einhaltung gewisser Regeln wird vom Werkzeug geprüft,
    - Nachbearbeitungsaufwand wird reduziert
  - Nachteile:
    - Keine echte XML-Unterstützung,
    - Eingeschränkte Funktionalität,
    - Nicht alle Strukturfehler werden erkannt,
    - Erstellung komplexer Dokumente weiterhin sehr fehleranfällig,
    - Erweiterungen meist proprietär
  - Beispiele:
    - Word & entsprechende Formatvorlagen,
    - Word & Makros

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

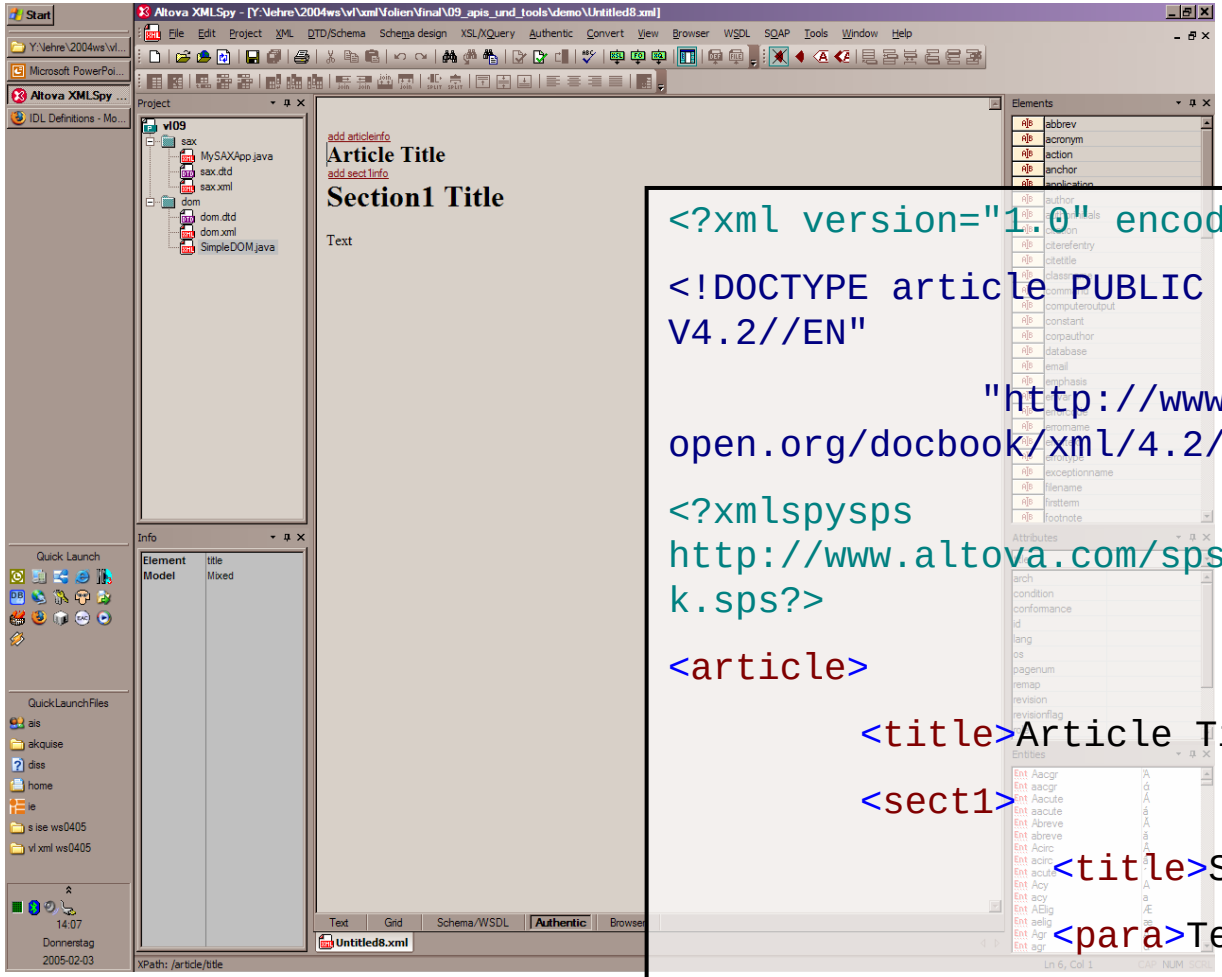
2.2 Dokumentenerstellung

- Texteditor mit XML-spezifischer Erweiterung durch den Hersteller:
  - Vorteile:
    - Unterstützung des Autors bei der Eingabe,
    - Bieten Hilfestellung bzgl. der Dokument-Struktur,
    - Verbergen die reine XML-Syntax vor dem Autor,
    - Integration von Publishing-Werkzeugen und XML-Funktionalität
  - Nachteile:
    - Relativ teuer,
    - Umstieg nicht immer machbar
  - Beispiele:
    - XML-Publishing mit Adobe FrameMaker
    - Authentic Views in XML Spy

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.2 Dokumentenerstellung




```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML
V4.2//EN"

                "http://www.oasis-
open.org/docbook/xml/4.2/docbookx.dtd">

<?xmlspysps
http://www.altova.com/sps/Template/Publishing/docboo
k.sps?>

<article>

        <title>Article Title</title>

        <sect1>

                <title>Section1 Title</title>

                <para>Text</para>

        </sect1>

</article>
```

2.2 Dokumentenerstellung

- XML-Editor:
  - Vorteile:
    - Unterstützung des Autors bei der Eingabe,
    - Bieten insbesondere Hilfestellung bzgl. der Dokument-Struktur,
    - Verbergen die reine XML-Syntax vor dem Autor,
    - Prüfen die XML-Instanz auf Korrektheit ("wohlgeformt" und "valide")
    - Besitzen oft "WYSIWYG"-Darstellung, dazu häufig Stylesheets nötig,
    - Erlauben dem Anwender auch im Quelltext zu arbeiten
  - Nachteile:
    - Relativ teuer,
    - Gewöhnungsbedürftige Oberfläche
  - Beispiele:
    - XMetal, XML Pro, XML Spy, ...

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.2 Dokumentenerstellung

- XML-Editor:

books.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Thraenert (Uni Leipzig) -->
<!DOCTYPE library SYSTEM "library.dtd">
<library>
    <book>
        <author>
            <name last="Behme" first="Henning"/>
        </author>
        <author>
            <name last="Mintert" first="Stefan"/>
        </author>
        <title>XML in der Praxis</title>
        <publisher>Addison-Wesley</publisher>
        <year>2000</year>
    </book>
    <book>
        <author>
            <name last="Hansen" first="Hans"/>
        </author>
        <author>
            <name last="Fritzen" first="Fritz"/>
        </author>
        <title>Der verflixte Hans</title>
        <publisher>Hans-Verlag</publisher>
        <year>1977</year>
    </book>
</library>
```

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

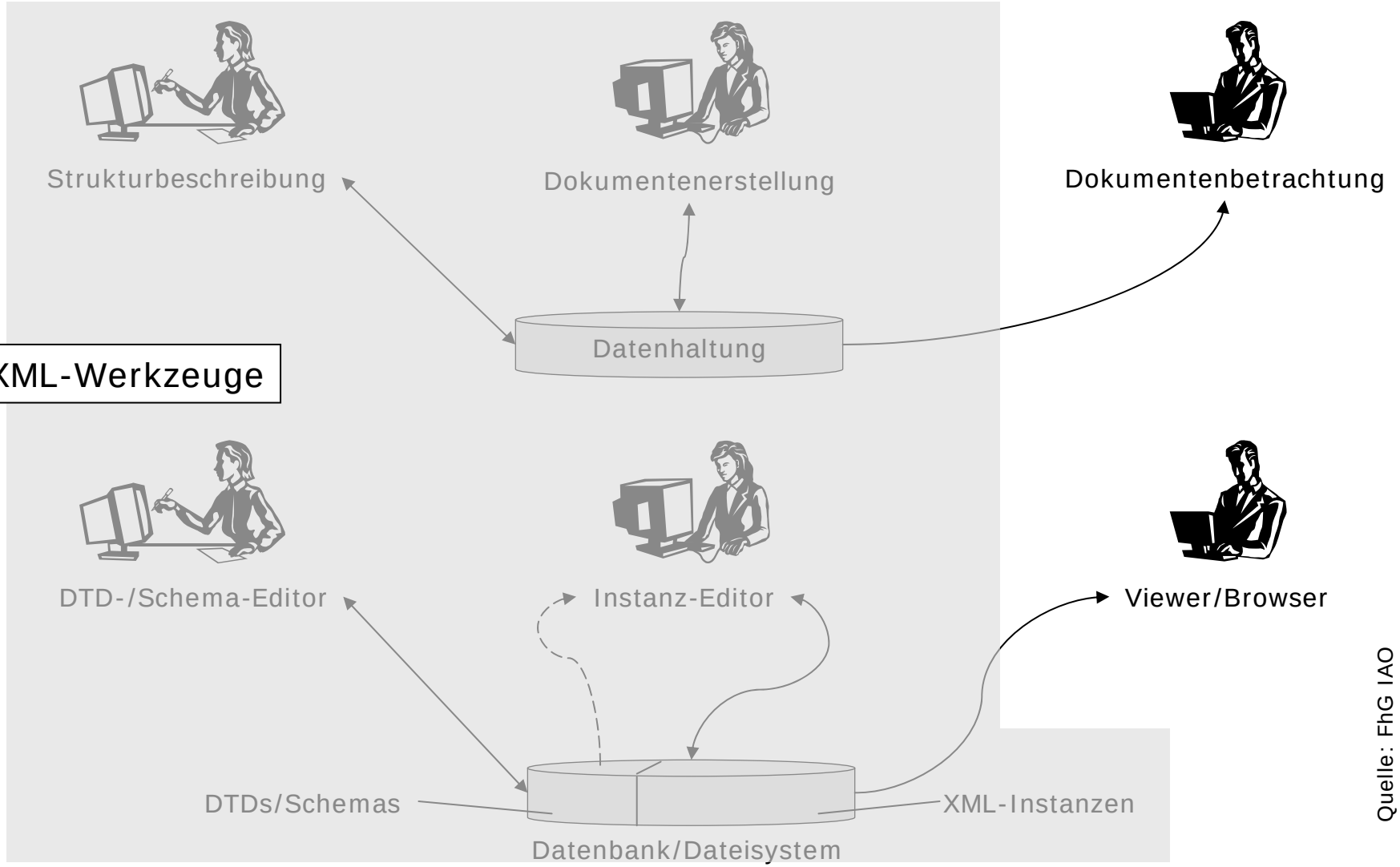
1. Programmierschnittstellen
   1. Grundsätzliches,
   2. SAX,
   3. DOM,
   4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
   1. Strukturbeschreibung,
   2. Dokumentenerstellung,
   3. **Dokumentenbetrachtung,**
   4. Datenhaltung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Werkzeugunterstützung / Werkzeugkategorien

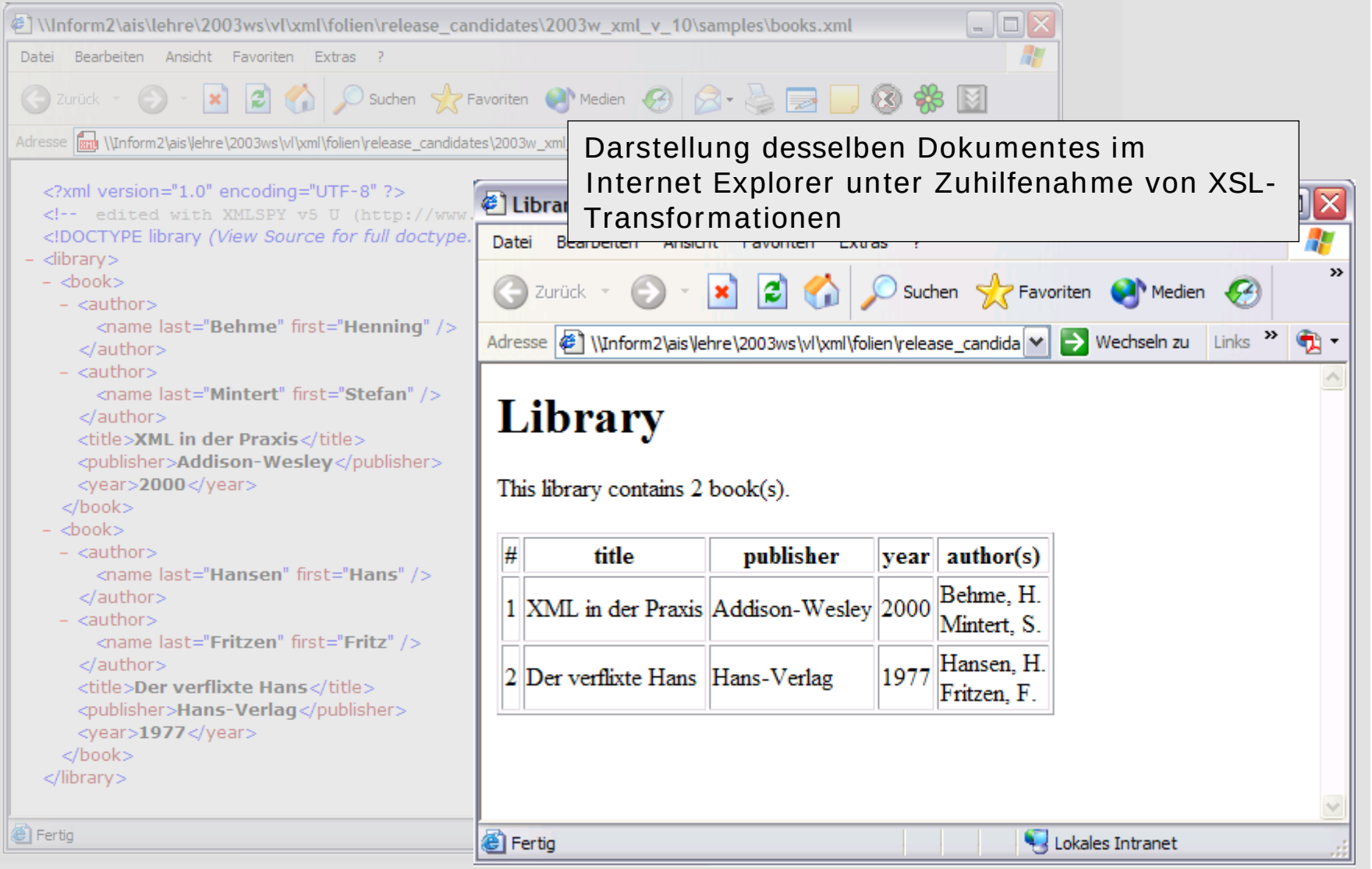
Werkzeugkategorien zur Dokumentbearbeitung



Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.3 Darstellung von Dokumenten



Darstellung desselben Dokumentes im Internet Explorer unter Zuhilfenahme von XSL-Transformationen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.3 Darstellung von Dokumenten

- Lesender Zugriff sollte für einen großen Personenkreis möglich sein:
  - Direkte Anzeige von XML-Dokumenten ➔ Struktur-Sicht
  - Formatierte Anzeige ➔ Verwendung von Stylesheets,
  - Anzeige von serverseitig bereits in HTML konvertierten XML-Dokumenten ➔ serverseitige Publishing-Systeme, z. B.: Cocoon

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

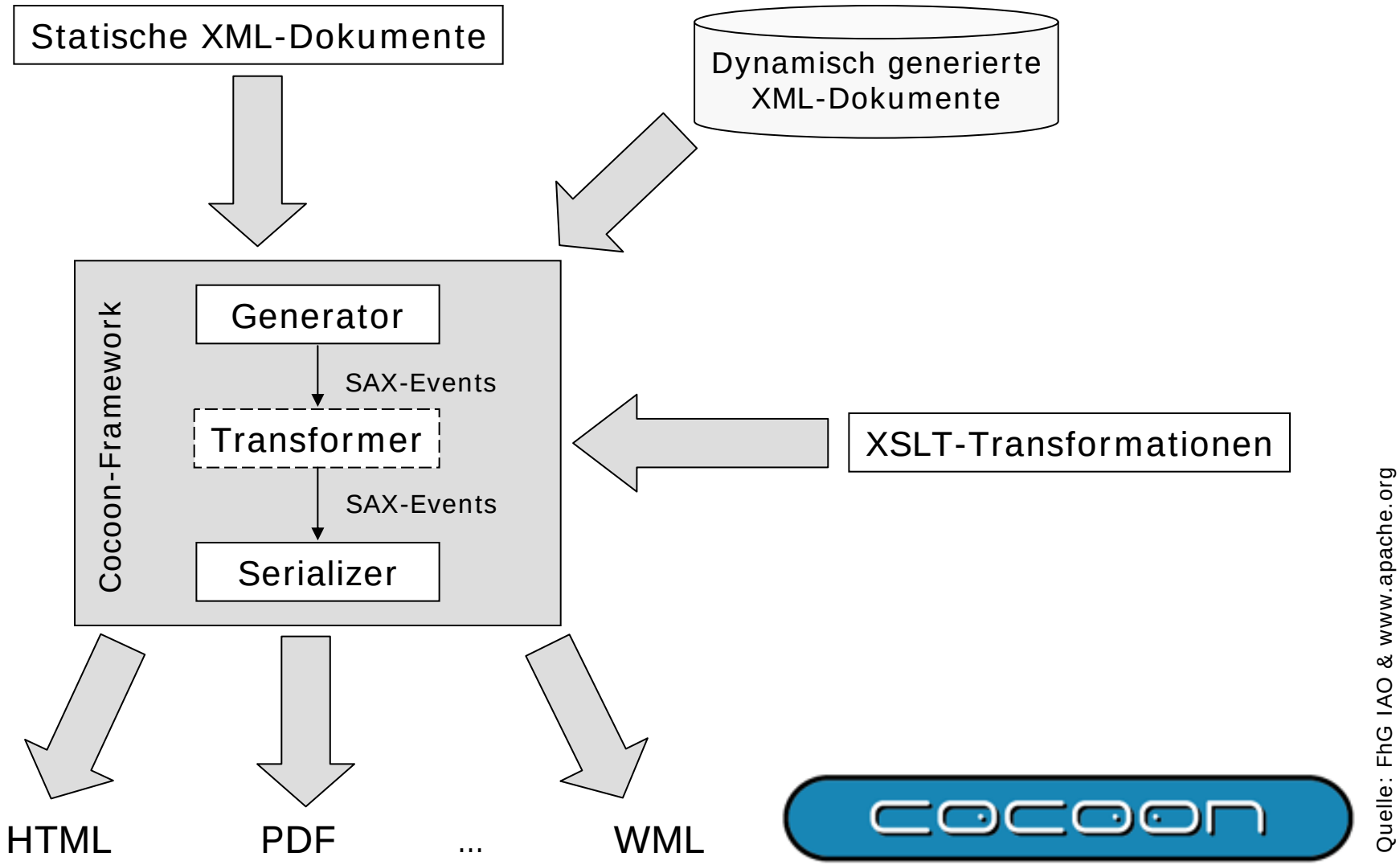
2.3 Serverseitiges Publishing: Cocoon

- Funktionsweise:
  - Statisch oder dynamisch erzeugte XML-Dateien werden auf Serverseite in HTML oder andere Formate transformiert,
  - Zur Steuerung der Umwandlung wird XSLT verwendet.

- Vorteile:
  - Aus einer Datenquelle können mehrere verschiedene Ausgabeformate erzeugt werden,
  - Auf Clientseite wird nur ein HTML-fähiger Browser benötigt,
  - Implementierung ist Open-Source, sie verwendet nur Komponenten des Apache-Projekts:
    - Apache Web-Server,
    - Tomcat Servlet-Engine,
    - Xerces XML-Parser,
    - Xalan XSLT-Prozessor,
    - FOP XSL Formatting Object

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.3 Serverseitiges Publishing: Cocoon



Quelle: FhG IAO & www.apache.org

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Übersicht

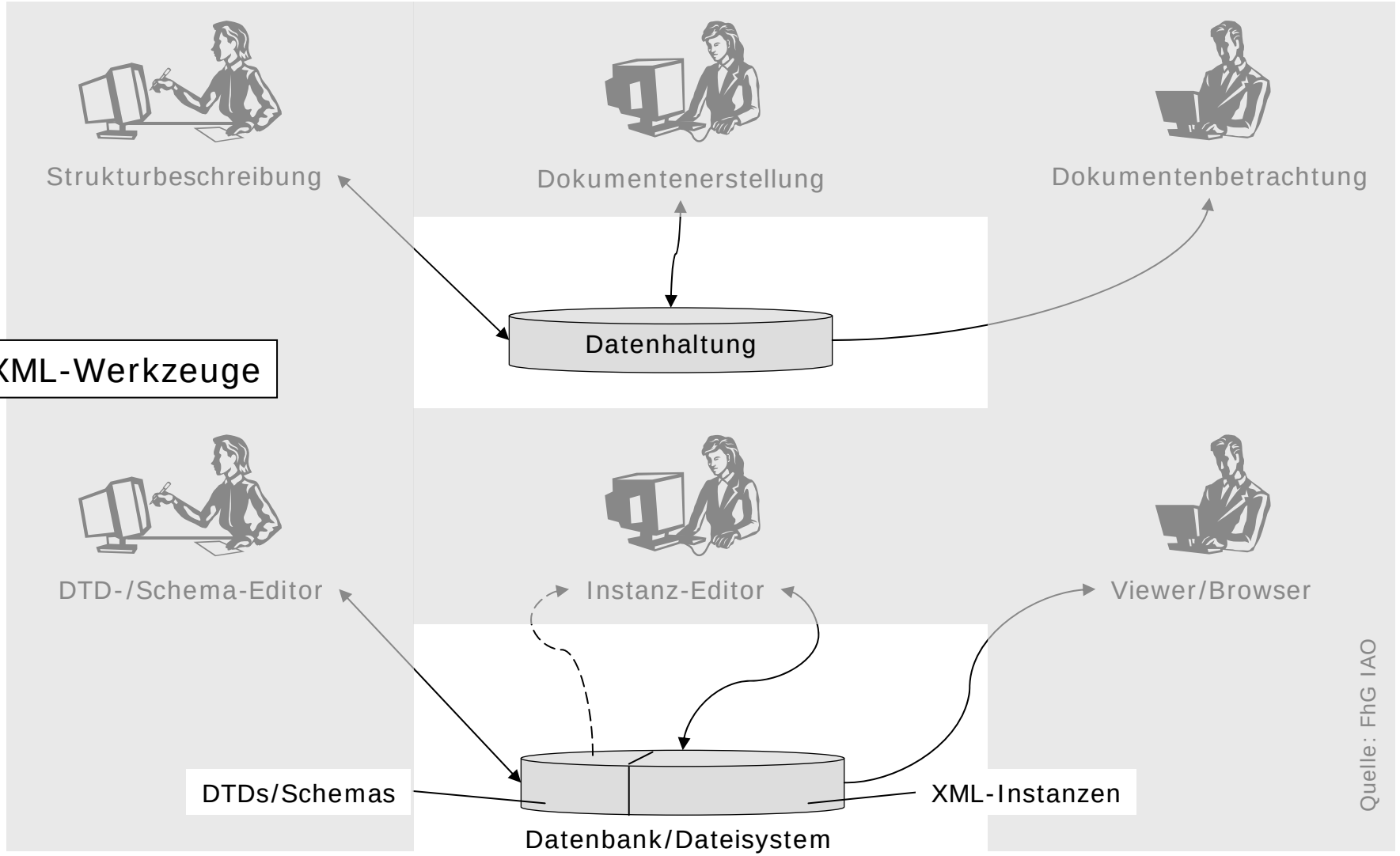
1. Programmierschnittstellen
    1. Grundsätzliches,
    2. SAX,
    3. DOM,
    4. Vergleich SAX – DOM

2. Werkzeuge zur Bearbeitung von XML-Dokumenten
    1. Strukturbeschreibung,
    2. Dokumentenerstellung,
    3. Dokumentenbetrachtung,
    4. **Datenhaltung**

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

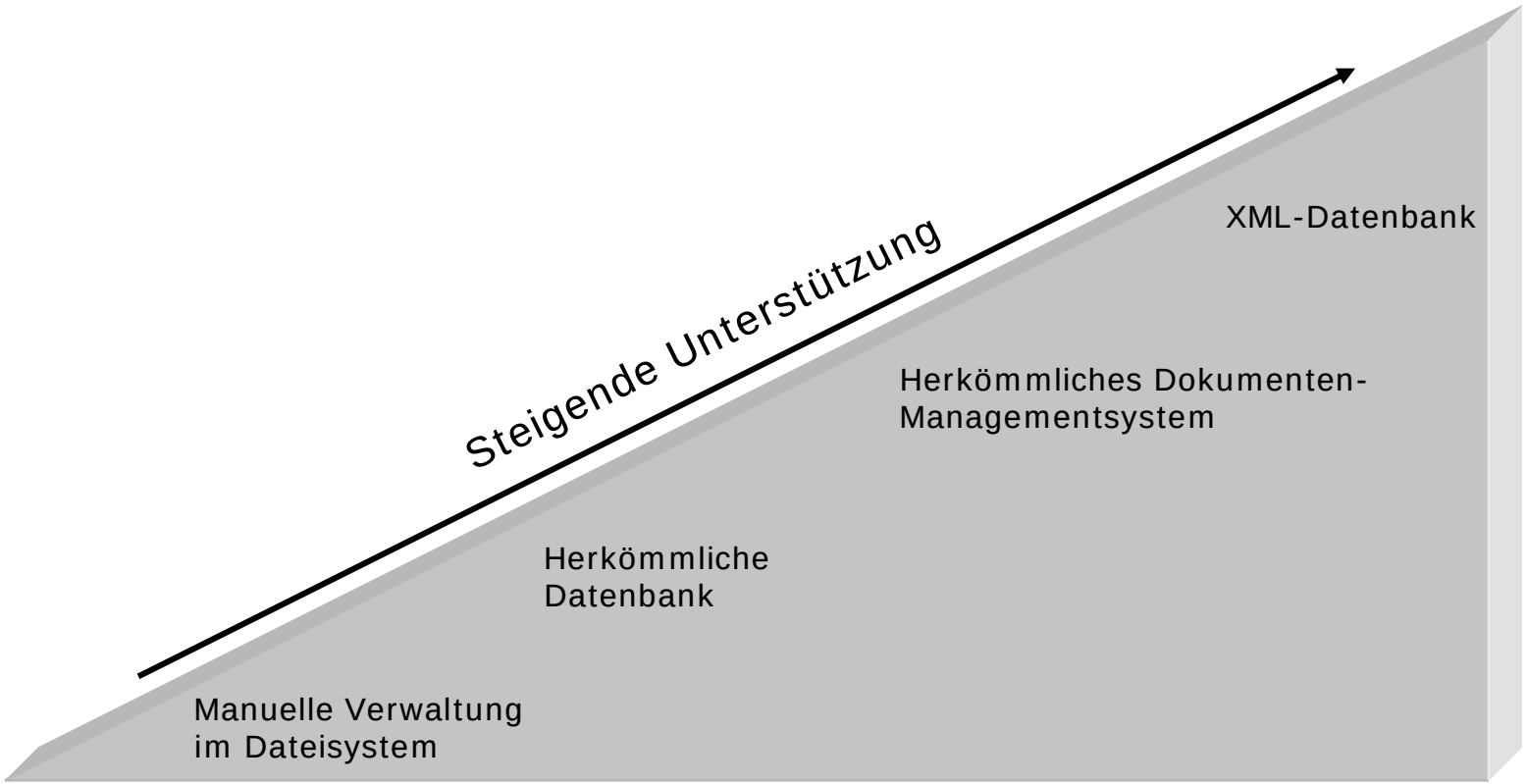
Werkzeugunterstützung / Werkzeugkategorien

Werkzeugkategorien zur Dokumentbearbeitung



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung



Quelle: FhG IAO

2.4 Datenhaltung

- Manuelle Verwaltung im Dateisystem:
  - Vorteile:
    - Anwender ist an die Bedienung des Dateisystems gewöhnt,
    - Dateisystem verursacht keine Extrakosten (Anschaffung),
    - Zukünftige Betriebssysteme erweitern die Funktionalität des Dateisystems:
      - Versionsverwaltung,
      - Indexierung über Schlagworte
  - Nachteile:
    - Keine XML-spezifische Unterstützung,
    - Manuelle Verwaltung bei großen Datenmengen sehr aufwändig,
    - Momentan noch keine Unterstützung für Versionsverwaltung usw.

Quelle: FhG IAO

2.4 Datenhaltung

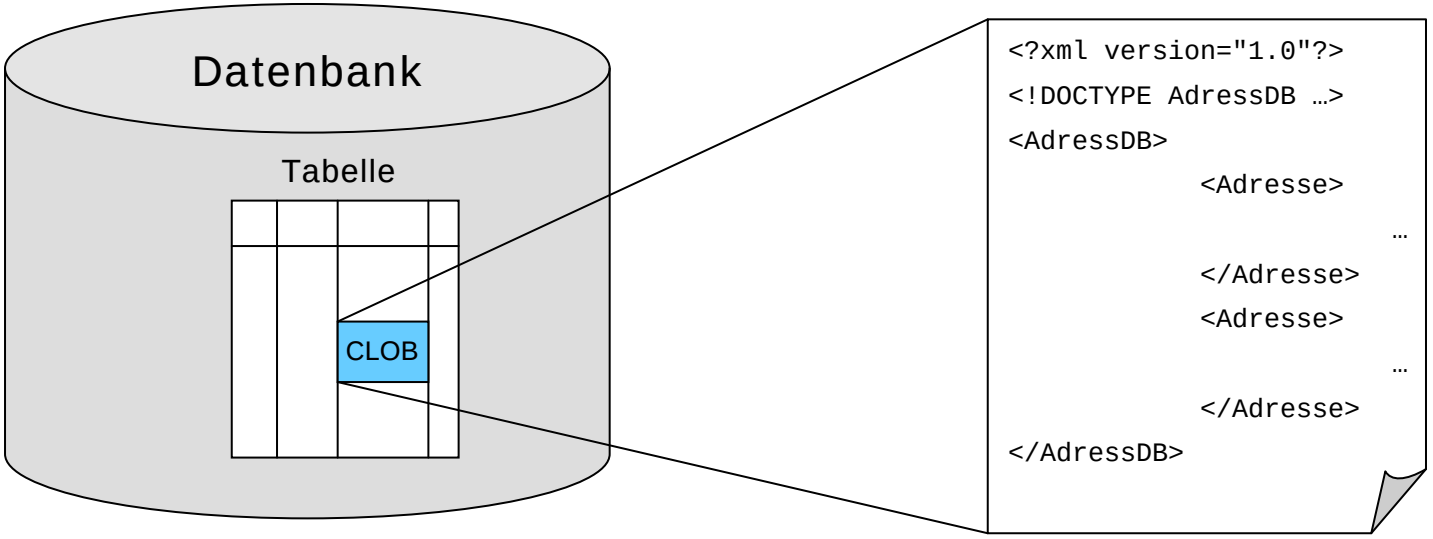
- Herkömmliche Datenbank:
  - Vorteile:
    - Meist schon im Unternehmen im Einsatz (RDBMS),
    - Administratoren benötigen keine weitere Schulung,
    - Individuelle Anpassungen sind unter Umständen einfach möglich,
    - Viele Hersteller bieten XML-spezifische Erweiterungen an
  - Nachteile:
    - Keine native XML-Unterstützung (Zugriff auf Elemente, …),
    - Anpassung/Erweiterung um XML-Spezifika teuer und aufwändig,
    - Versionsverwaltung muss individuell geregelt werden,
  - Alternative:
    - Speicherung der XML-Daten als BLOB oder als Daten in einem relationalen/objektrelationalen Schema,
    - Problem: Mangelnde Flexibilität des Datenbank-Schemas
  - Beispiele:
    - MS SQL Server, Oracle, IBM DB2, …

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung

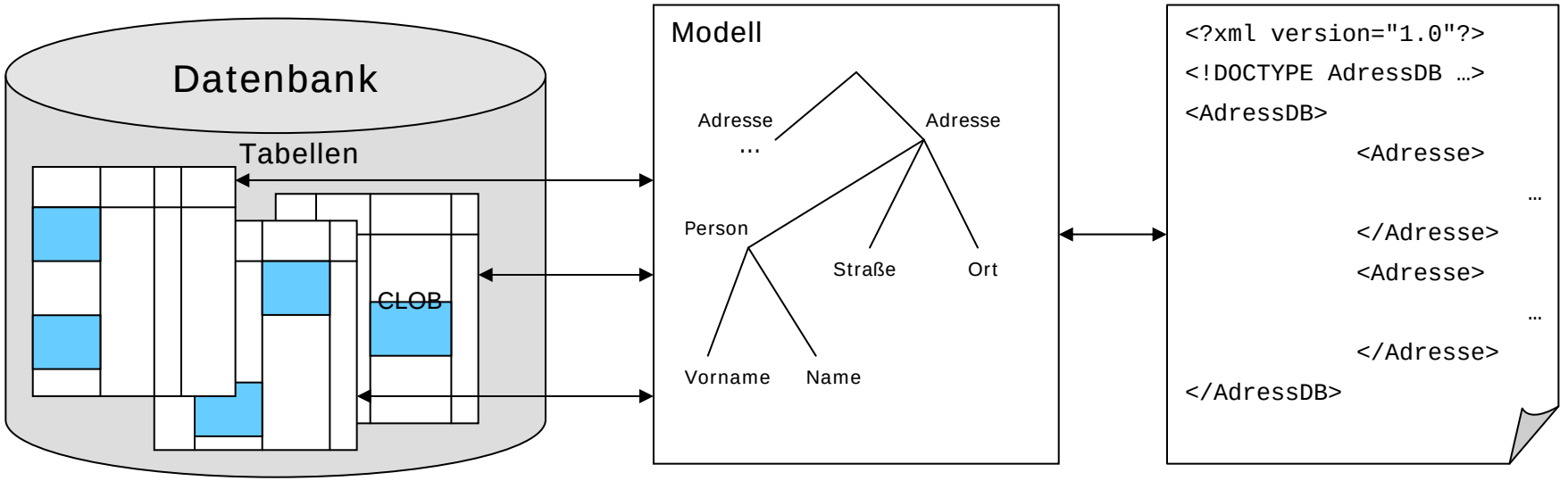
- Möglichkeiten zur Speicherung von XML in RDBMS:
  - Speicherung als „flache Zeichenkette“
    - Speicherung erfolgt entweder in einem CLOB oder einer Textdatei
    - XML-Dokument wird originalgetreu gespeichert und wiederhergestellt,
    - hohe Performanz bei der Rückgabe des gesamten Dokuments,
    - ineffektiv bei der Suche im XML-Dokument, da nicht die DBMS-Funktionen zur Suche genutzt werden können



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung

- Möglichkeiten zur Speicherung von XML in RDBMS:
  - Modellbasierte Speicherverfahren:
    - XML-Dokument wird durch strukturelle Analyse auf Tabellen abgebildet,
    - Speicherung in einer oder mehreren relationalen Tabellen,
    - SQL-Anfragen greifen auf Strukturinformationen des XML-Dokuments zu,
    - schnelle Antworten bei Anfragen zu Teilen des Dokuments,
    - langsame Verarbeitung, wenn das gesamte Dokument angefordert wird



UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung

- Modellbasierte Schemaabhängige Verfahren:
  - Basieren auf einem Relationenschema, welches nur diesem Schema entsprechende XML-Dokumente speichern kann,
  - Schema kann durch DTD oder XML-Schema gegeben sein,
  - Vorteil: Daten können ohne aufwändige Navigation durch die XML-Hierarchie direkt angefragt und geändert werden,
  - Nachteil: sehr inflexibel, bei Änderungen am Dokumentenschema muss das DB-Schema geändert werden

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung

- Modellbasierte Schemaunabhängige Verfahren:
  - Nutzen ein Relationenschema, welches beliebige XML-Dokumente speichern kann,
  - Vorteil: hohe Flexibilität bei der Speicherung,
  - Nachteil: hoher Aufwand bei der Navigation durch die gespeicherte Baumstruktur, um Daten an den Blättern zu finden

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

2.4 Datenhaltung

- XML-Datenbank:
  - Vorteile:
    - Bieten spezielle XML-Unterstützung:
      - Parsing beim Check-In,
      - Check-Out von Dokumententeilen,
      - Herunterbrechen der Speicher-/Sperrgranulate auf ein definiertes Elementniveau,
      - Suche nach Elementinhalten/Attributwerten mit XQuery/XQL
    - Unterstützt die Wiederverwendung von Dokumentteilen,
    - Unterstützt die Parallelbearbeitung von Dokumenten
  - Nachteile:
    - Noch relativ neue Produkte
  - Beispiele:
    - Tamino (Software AG),
    - eXcelon (eXcelon Corp.),
    - X-Hive/DB (X-Hive Corp.)
    - Xindice (Apache XML Project)

Quelle: FhG IAO

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Agenda

- Zusammenfassung des WB Programmierschnittstellen und Werkzeuge
- **Fragen?**
- Weiterer Ablauf

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Agenda

- Zusammenfassung des WB Programmierschnittstellen und Werkzeuge
- Fragen?
- **Weiterer Ablauf**

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Weiterer Ablauf

- Selbststudium des Materials
  - Beispiele eigenständig nachvollziehen
  - Unklarheiten sind die Basis für die Diskussion in den Präsenzveranstaltungen
- Präsenzveranstaltungen (Termine vorläufig, montags, 09:15–10:45 Uhr, Johannisgasse 26, R 1-22)
  1. 2008-05-19: Einführung/Strukturbeschreibung (SK)
  2. 2008-06-09: Adressierung, Abfrage und Speicherung (MT)
  3. 2008-06-23: Document-Linking/Transformation und Präsentation (MT)
  4. 2008-07-14: APIs und Werkzeuge, Semantic Web (MG, TR)
  5. 2008-07-31f: Modulprüfung BIS
- Aktuelle Informationen siehe http://bis.informatik.uni-leipzig.de/

Literatur

- Merz, M.: *E-Commerce und E-Business*, dpunkt, 2002
- Behme/Mintert: *XML in der Praxis*, Addison Wesley, 2000
- Behme: XML & Co, Die Spezifikationen für Dokumenten- und Datenarchitektur, Addison-Wesley, 2002
- Dbooks (digitale Publikationen) (http://www.dbooks.de)
- Tutorial Fraunhofer IAO (Stuttgarter E-Business-Tage 2001)
- Rahm, E.: *Skript zur Vorlesung DBS2*
- http://www.w3.org/TR/xquery/

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Literatur

- H. Vonhoegen: *Einstieg in XML*, Galileo, 2002
  - Leseprobe verfügbar unter: http://www.galileocomputing.de/download/dateien/283/galileocomputing_xml.pdf

- http://selfhtml.teamone.de/javascript/intro.htm
- http://www.saxproject.org
- http://www.w3.org
- http://www.jdom.net