VSR | EDU

XML

XML

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und Selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Chapter 2

INTRODUCTION

Idea: Markup

- **Markup** – Text added to data in a document carrying information about this data
  - http://www.w3.org/MarkUp/
  - Idea originates from the publishing industry (document typeset instructions)

- Example
  - *Markup:* *Hello*
    - '*' is interpreted as Markup in certain editors → **Hello**

Idea: Markup Language

- **Markup language** – Defines…
  - Markup elements (strings called Tags)
    - The term Tag comes from Stanley Rice, a New Yorker book designer, his "editorial structure tags" idea published in the late sixties
  - As well as rules to element application and document setup
    - Thereby, document setup structures are partially fixed, but requirements for document extension are defined.
- Example
  - *Markup language:* Hypertext Markup Language (HTML)
    - Markup provides information for display and interaction of data in an HTML document

Markup Example: HTML

```
<html>
  <body>
    <table border="1">
    <tr>
        <td>Water</td>
        <td>1 EUR</td>
     </tr>
     <tr>
        <td>Beer</td>
        <td>1.5 EUR</td>
     </tr>
     <tr>
        <td>Wine</td>
        <td>1.5 EUR</td>
     </tr>
    </table>
  </body>
</html>
```

Markup Example: HTML

```
<html>
 <body>
  <table border="1">
   <tr>
     <td>Water</td>
     <td>Beer</td>
     <td>Wine</td>
   </tr>
   <tr>
     <td>1 EUR</td>
     <td>1.5 EUR</td>
     <td>1.5 EUR</td>
   </tr>
  </table>
 </body>
</html>
```

Problem Area HTML

- HTML documents as information carriers
  - **Advantages:**
    - Platform- and vendor-independent
    - Simple, readable if needed
    - Markup encoding in ASCII – simple to read and simple to create
    - Easily extensible in the browser environment (see Browser War)
  - **Disadvantages:**
    - Processing is difficult, since focus lies on presentation, not on semantics
    - Unclear semantics (see extensible)
    - Predefined structure, user (author of the document) can't customize it
    - Information extraction and reuse is complex

Problem: Flexibility

- **Requirement**:
  - *Simple language* for describing Markup languages
  - Simple approach of creating *structured data* (such as tables, see DBMS)
  - Simple approach of creating *flexible structures – semi-structured data models*
- **Approaches from the publishing industry:**
  - Originates from IBM 1969 with the Generalized Markup Language (GML, by Goldfarb, Mosher and Lorie)
  - SGML - Standard Generalized Markup Language, ISO-Standard 8879:1986
  - Example: HTML is an **application of SGML**
  - ***Problem:*** SGML is complex

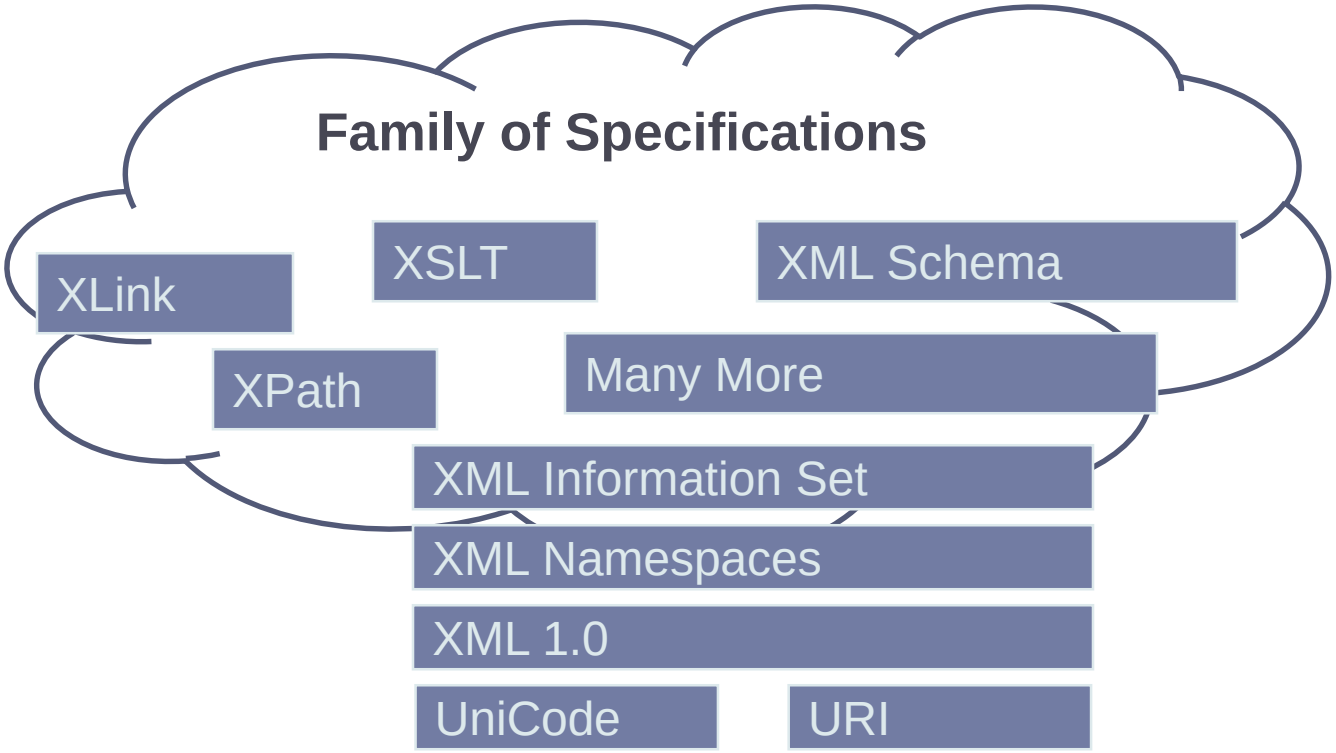
Solution: XML

- **eXtensible Markup Language (XML)**
  - W3C Recommendation – Universal format for structured documents and data on the Web: http://www.w3.org/XML/
  - XML is a simple meta-language for Markup language definition
    - Enables a semi-structured data model → Documents of one type can be structured differently
    - Enables self-description of data, i.e. XML documents can contain data and structure of that data (no separation of data-schema and specification as in databases)
  - Doesn't focus on automatic processing of a particular XML document (no semantic description)
  - Example: RSS, XHTML, XML/EDI, SVG are applications of XML

XML Retrospect

- 1996 Beginning of development
  - and introduction at the SGML Conference in November
- 1997 Public Drafts
  - The "red booklet" is distributed at the WWW6 Conference, Santa Clara, CA, USA
  - First XML Conference conducted by Tim Bray takes place in March
- 1998 W3C REC
  - W3C adopts XML as a recommendation, focus lies on simplification of SGML (based on experience in dealing with the Web)
  - XML = 80% of SGML's possibilities, but only 20% of SGML's complexity
  - XML documents can (like HTML) be written in a simple way (ASCII) and transported equally simply (HTTP)

XML – Family of Specifications

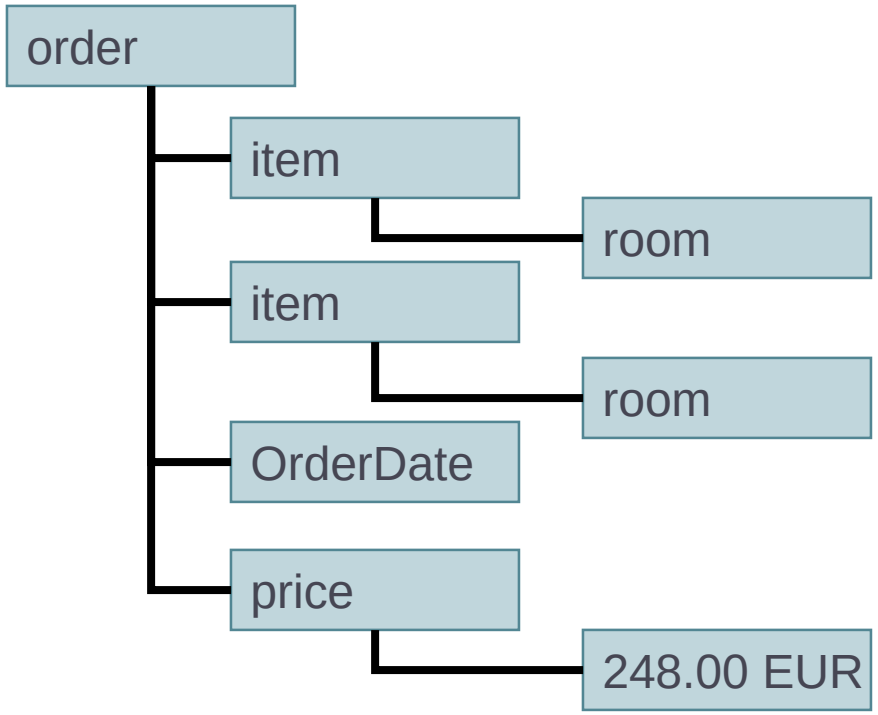


...and grows and grows and grows!

XML Example

- XML document as an element tree
  - XML documents consist of elements and attributes

```
<?xml version="1.0"?>
<order OrderID="10643">
    <item>
        <room id="Room10"/>
    </item>
    <item>
        <room id="Room11"/>
    </item>
    <OrderDate
        ts="2004-05-17T00:00:00"/>
    <price>248.00 EUR</price>
</order>
```



XML and Distributed Applications

- Rules of XML application are simple
- XML document is
  - **Well-Formed** if it complies with all the XML rules
    - All elements are closed, for example, <tag>Data</tag>
    - Empty elements are closed with " / ", such as <emptyelem/>
    - Attribute values in quotes: <element attribute="123"/>
  - **Valid** if
    - it is well-formed and
    - document rules adhere to Document Type Definition or a Schema
- Data (as application basis) in form of XML documents
  - Can be read, written, *reused*, *transported* and *exchanged* in a simple fashion
  - Validity can be easily checked
  - XML documents are the foundation for distributed applications on the Web

Concluding Remarks

OUTLOOK

Content Theme - Examples

- **XML Basics**
  - Validity: XML-DTD, Schemas
  - Standard attributes: XML Base, xml:id
  - Transformation: XSLT
  - Addressing: XPath
  - Linking: XLink, etc.
  - Selection: XQuery
  - Security: Signature und encryption
- **Languages: Data**
  - Texts: docBook
  - Formulae: MathML
  - Syndication: RSS
  - Graphics: SVG
- **Languages: Semantics**
  - Semantic Web: RDF, OWL, FOAF
  - Software development: DSLs, XMI
- **Tools**
  - Parsers
  - Editors
  - Validators
  - Databases

UX Theme - Examples

- **Presentation:**
  - XSL-FO
- **Dialog:**
  - XForms
- **Navigation:**
  - SiteMap
- **Languages:**
  - XHTML
  - VoiceXML
  - XAML
  - XUL
- **Tools:**
  - Editors
  - Browsers
  - Validators

(UX: User Interface Experience)

SOA Theme - Examples

- **Transport:**
  - SOAP
  - REST
- **Endpoints:**
  - WSDL
  - WADL
- **Wiring:**
  - BPEL4WS
  - XPDL
  - BPML
- **Services:**
  - Amazon AWS
  - Blogger
  - del.icio.us
  - UDDI
- **Tools:**
  - Editors
  - Generators
  - Development tools
  - WS-Composition Systems
  - Workflow-Engines

Chapter 3

XML-DOCUMENTS BASICS

XML 1.0 Specification

- Extensible Markup Language (XML) 1.0
- Since February 10 1998: **W3C Recommendation**
  - Latest: http://www.w3.org/TR/REC-xml/
  - "The function of the markup in an XML document is to describe its **storage and logical structure** and to **associate attribute-value pairs with its logical structures**. XML provides a mechanism, the **document type declaration**, to define constraints on the logical structure and to support the use of predefined storage units."

Markup in XML Documents

K03-01.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<News>
  <date>10.04.2007</date>
  <description headline="Neues zu XML">
    XML macht Spaß.
  </description>
</News>
```

- **Elements** – Define the logical structure
- **Attributes** – Enable element association with additional information via name-value pairs
- **XML Declaration** – Information for interpretation of the logical structure by a parser

Element

- **Element** – Element has a *Name*, *Start-* and *End-Tag* as well as *Content*.
- **Content**
  - **Unstructured** (character data)
  - **Structured**
  - **Mixed** (mixed content)
  - **Empty**

K03-01.xml | **K03-02.xml**

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

  <Unstrukturiert>
    <![CDATA[ Beliebige Zeichen & > < <C><c>//]]>
  </Unstrukturiert>

  <Strukturiert>
    <UnterElement>
      <UnterElement>...</UnterElement>
    </UnterElement>
  </Strukturiert>

  <Gemischt>
    Daten
    <UnterElement> Daten </UnterElement>
    Daten
  </Gemischt>

  <Leer></Leer> == <Leer/>

</Elemente>
```

Attribute

K03-01.xml | K03-02.xml | **K03-03.xml**

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

  <Element attribut="Wert" sprache="DE" Datum="11.04.2007"/>

</Beispiel>
```

- **Attribute** – Name-value pair
  - Value (for now) of type String
  - Order of attributes is irrelevant
- Element vs. Attribute
  - Attribute serves the sole purpose of transporting element's metadata
  - Compact notation, but inflexible – no nesting

XML Declaration

K03-01.xml | K03-02.xml | K03-03.xml | **K03-04.xml**

```
<?xml version="1.0" encoding="utf-8" standalone ="yes"?>
<?xml encoding="ISO-2022-JP"?>
```

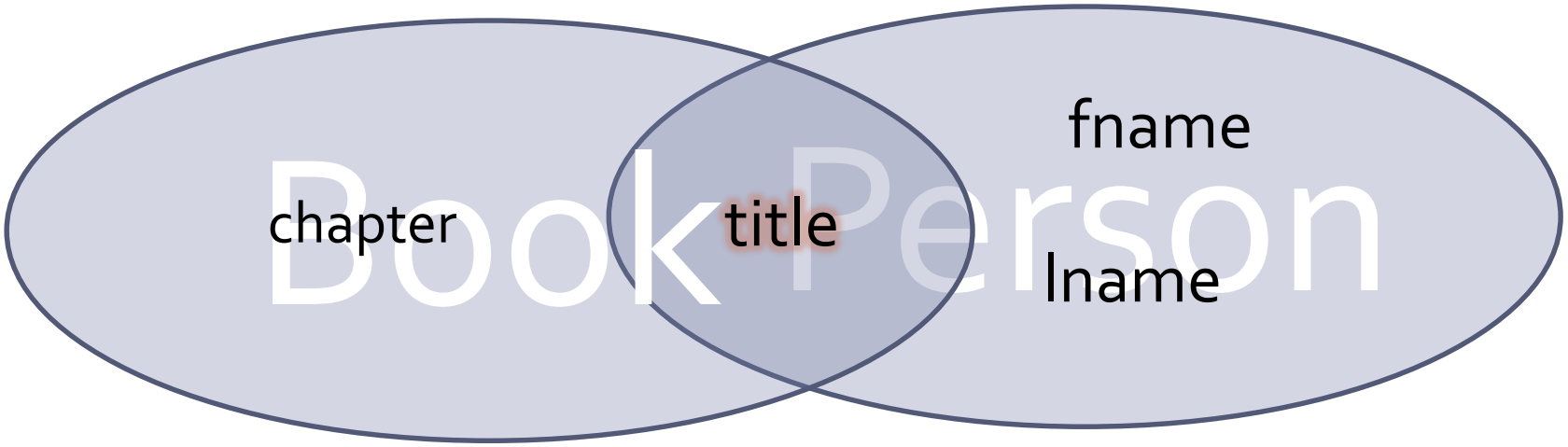
- **Declaration** – Provides instructions to the XML Processor (order is relevant!)
  - version (optional): XML Version used
  - encoding (mandatory): encoding of the XML Document
  - standalone (optional): "yes" means that there are no external Markup Declarations to process (apart from the Document itself)
- Declaration must occur at the beginning of the document
- **XML Processor** – the program that processes the XML document, i.e. a parser, and enables access to the content and structure of the XML document.

Rules for Well-Formedness

1. XML documents have at least one element
   - The first element is called "root"
2. Start- and end tags are in the same content
   - Right: <parent><child></child></parent>
   - Wrong: <parent><child></parent></child>
3. Each non-empty Start-tag must have a corresponding end tag
   - Case sensitivity
4. Naming conventions must be complied with
   - Names start with "_" or letters and can contain numbers
   - Not allowed are especially ":" and "=" in a name, as well as names starting with "xml"
5. Formatting (*white space) in text* is taken into account
6. Attribute names of an element are always unique

Namespace / 1

- *Problem:* Two elements – same name but different meaning?
  - **<title>** (*Book*) **<title>** (*Person*)



Namespace / 2

- *Idea:* Append a prefix to an element name
  - title → book:title and person:title
  - *Question*: Which prefix should be used?
- *Solution*: **Namespace concept**
  - **Namespace –** qualified elements and attributes with an URI (URI "addresses" the space of elements and attributes)
  - Namespace URI identifies resources, which contain the names of contexts (spaces) (doesn't have to exist)
  - Namespaces can be assigned prefixes (One or more prefixes as well as a default namespace/standard namespace)

Namespace / 3

- Example
  - Namespace NS1 contains the following names: title, description
  - Namespace NS2 contains the following names: title, fname, lname
  - Let the prefix be NS1=„http://example.org/Textdocument“
  - Let the prefix be NS2=„urn:schema:person“
  - Then <NS1:title> and <NS2:title> can be differentiated

Uniform Resource Identifier (URI)

- **Uniform Resource Identifier (URI)** – Generic concept for textual names and addresses.
- **Syntax** for identifiers **[RFC3986, earlier RFC1630 – cf. IETF !!!!!]**
  - **<uri> ::= <scheme>":"<scheme-specific-part>**
    - scheme: scheme name
    - Scheme-specific-part: scheme-specific identifier
- Forms:
  - **Uniform Resource Locator (URL)** – URI with specified resource access
    - Example: http://www.example.org or mailto:n@example.org
  - **Uniform Resource Name (URN)** – URI with a certain naming convention identifying resources
    - Example: urn:my:own-URI
- All these forms depend on organizations when it comes to name assignment, for example, Internet Assigned Numbers Authority (IANA)

Namespace in XML (1)

K03-01.xml | K03-02.xml | K03-03.xml | K03-04.xml | **K03-05.xml***

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns="urn:StandardNamespace"
      xmlns:ns1="http://example.org/Textdokument" xmlns:ns2="urn:schema:person">
  <ns1:title>Buchtitel</ns1:title>
  <ns2:title>Graf von</ns2:title>
  <element>im Default-Namespace</element>
</root>
```

- Namespace declarations
  - One or more per element
  - Children inherit all declarations from parents
- **Namespace restrictions (Qualified)** – Element is assigned to a namespace, i.e.
  - Qualified: Assignment via prefix
  - Qualified: Assignment via standard namespace

Namespace in XML (2)

03-06.xml | K03-01.xml | K03-02.xml | K03-03.xml | K03-04.xml | K03-05.xml

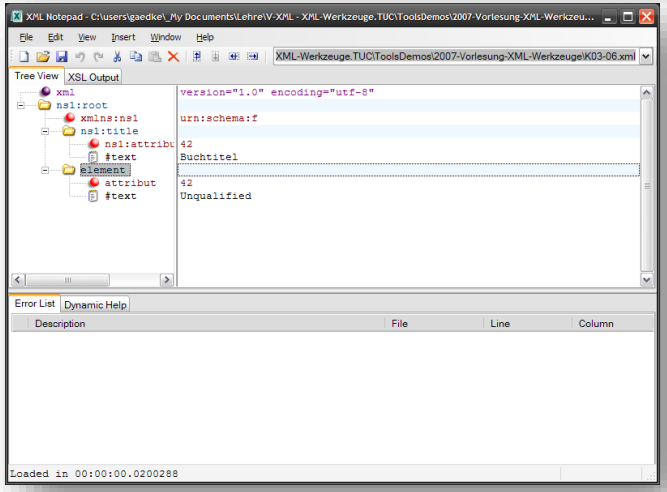
```
<?xml version="1.0" encoding="utf-8"?>
<ns1:root xmlns:ns1="urn:schema:f">
  <ns1:title ns1:attribut="42">Buchtitel</ns1:title>
  <element attribut="42">Unqualified</element>
</ns1:root>
```

- Qualified: Element „root“
  Unqualified: Element „element“
- Attributes can be assigned namespaces, but are often not, in order to achieve higher reusability (metadata association to the element) at the attribute level.

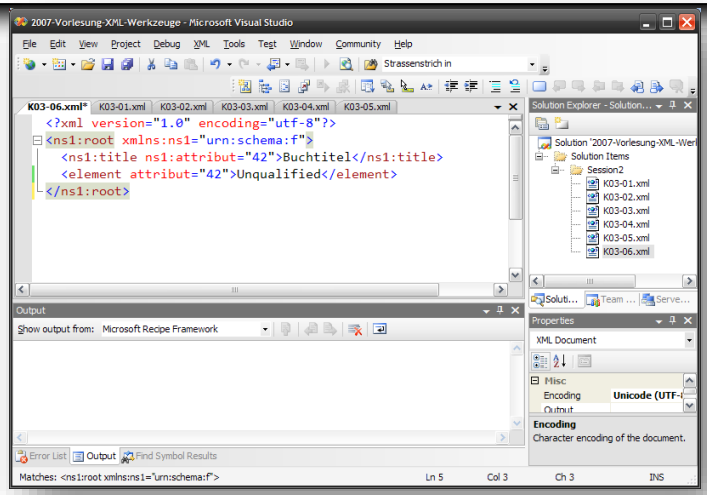
Authoring Tools

- Tools for XML data creation
  - **Text editors** such as vi, emacs, notepad
  - **XML editors** are specially designed to process various XML data
  - **Domain specific XML editors** provide a special user interface for processing of domain-specific data; they also offer the possibility of providing and maybe even importing internal data in XML
- Real-world XML editor examples
  - XML-Spy
  - XML-Notepad
  - XML editors in IDEs
  - New OpenOffice/Office products
  - Some websites, such as blogs with RSS export

XML Editor Examples

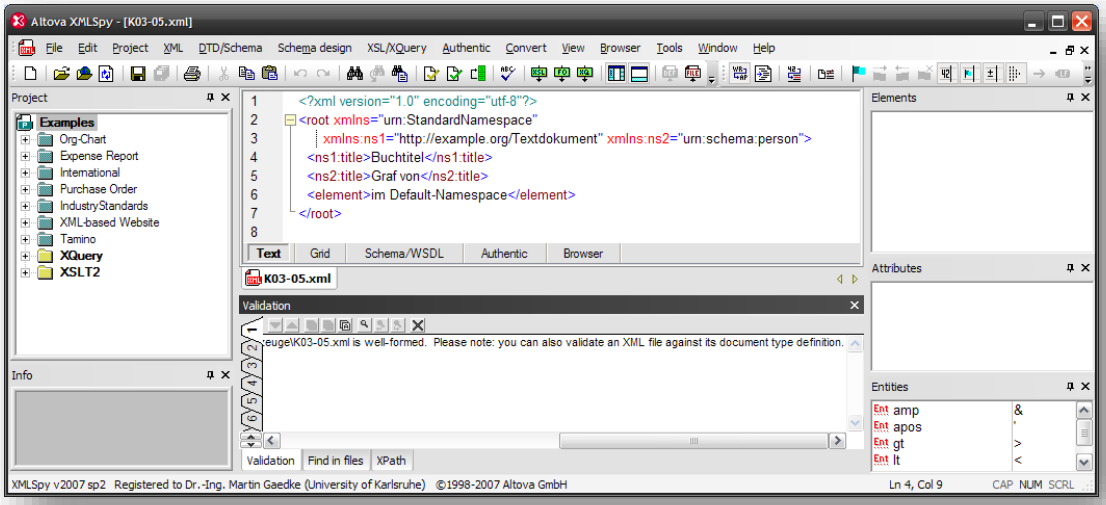


XML-Notepad



Visual Studio

Amazon/AWS



XMLSpy