Mag. iur. Dr. techn. Michael Sonntag

IMS CPS & MD

Content Packaging and Metadata According to the IMS standards

Institute for Information Processing and Microprocessor Technology (FIM)
Johannes Kepler University Linz, Austria

E-Mail: sonntag@fim.uni-linz.ac.at
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

Introduction

- Basics
  - → Ideas behind learning material standards
  - → Areas of standardization
- CPS: Content Packaging
  - → Overall structure
  - → Elements
- MD: Metadata
  - → Overall structure
  - → Where to add
  - → Basic required elements
- Creating a course

Why learning material standards?

- Electronic learning materials (ELM) are more difficult to prepare than "ordinary" materials:
  - → Much higher standards (Video, Animations, Applets, ...)
  - → More difficult (various tools must be known)
  - → More options (books are linear, ELM must be a net, etc.)
- Avoiding stranded costs and increasing versatility
  - → The learning content should be independent from the system to "deliver" it; especially vendor independent
- "Homegrown" formats usually leave out many important aspects; retrofitting data into existing courses is difficult
- Comparability of learning materials increased
  - → Teachers can select which materials to use
  - → Students can use specific parts depending on preferences

Areas of standardization

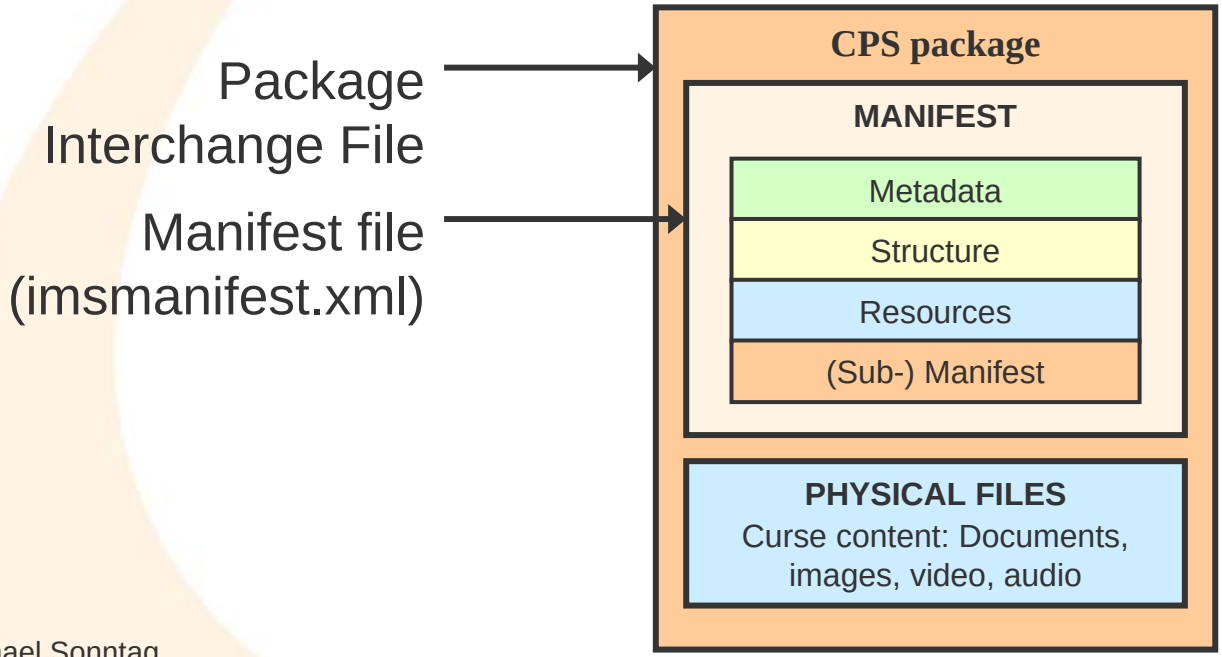
- Content packaging: Main item; the core of a course (CPS)
  - → To allow reuse of content in different "players"
- Metadata: Annotations of courses, elements, etc. (MD)
  - → To improve/ease the decision about (re)use
- Online tests: Immediate feedback and practice (QTI)
  - → Similar to CPS but for active content instead of "presentation"
- Learner profiles: Interests, levels of proficiency, ... (LD)
  - → For personalization and organization
- Sequencing: How to "play" a complete course (SS)
  - → Lecture, test, roleplay; prerequisites, ...
- Accessibility: Access for handicapped persons
  - → A kind of specialized metadata
- Repositories: Storing and finding courses

Content Packaging Specification

- Standard of the "Instructional Management Systems" (IMS) created by the Global Learning Consortium
  - → Independent forum (this is no software producer!)
  - → XML based specification
  - → Nowadays the de-facto standard (at least for import)
- Intended for interoperability of learning content and learning platforms of different manufacturers
- Metadata, course structure and paths to included files within the XML manifest are independent from actual content
- Advantages:
  - → Reusability
  - → "Platform independence"
  - → Base for international exchange of learning materials

CPS packages

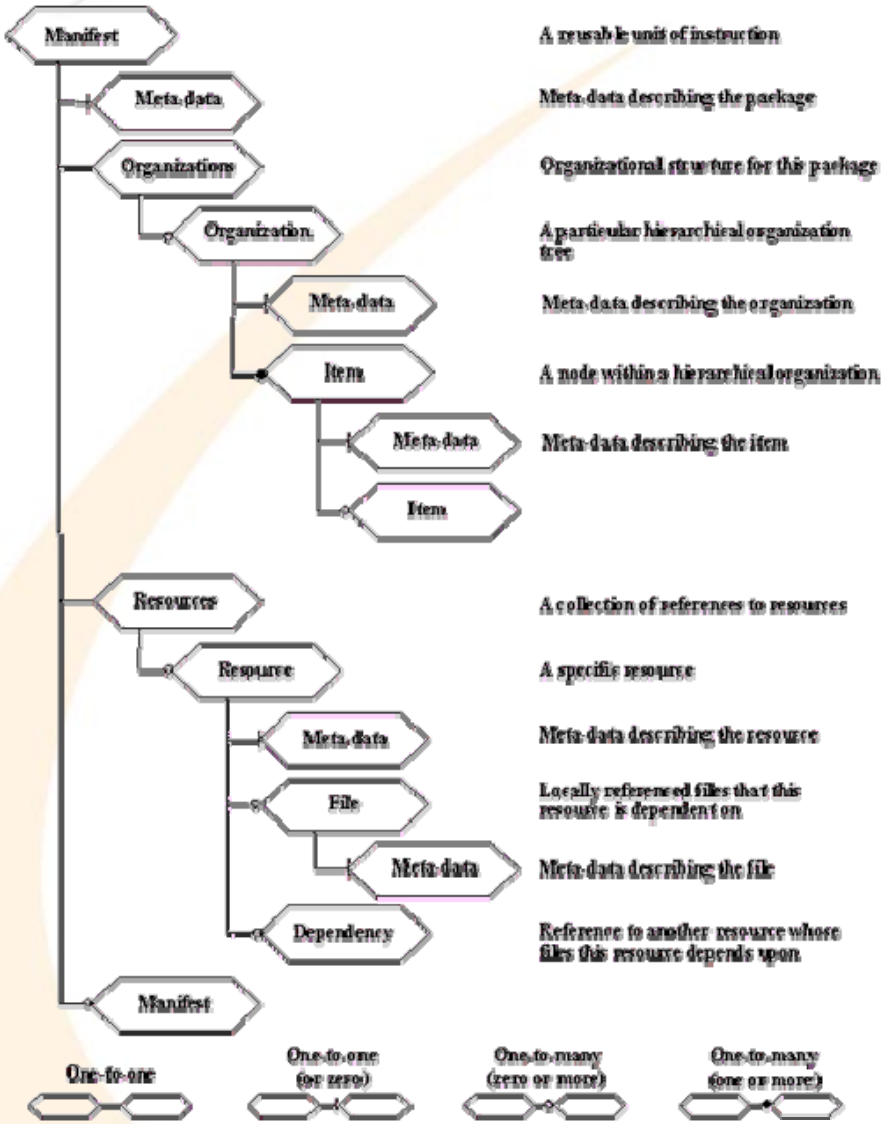
- CPS package = Learning package according to the CPS
- Structure of a CPS package:
  - → Manifest „imsmanifest.xml“ (XM file) in root directory
    - » Metadata
    - » Structure and references to the actual learning content
  - → Learning content in arbitrary formats



Structure of a manifest

- A manifest consists of these elements in exactly this order:
  - → Metadata (Optional): Describing the manifest (i.e. the course unit) as a whole
  - → Organizations (Required): How the package is organized
    - » Commonly used for the navigation
    - » Currently only defined structure is "hierarchical", which results in a tree-like structure (one root only, no cross-connections)
  - → Resources (Required): All the resources referenced in the organization(s) as a flat list
    - » Describes the actual files and additional required dependencies
      - – E.g. In the organization a webpage is included, the resource however also references images, applets, etc. on this page
  - → Submanifests(optional): For a hierarchical aggregation of manifests; e.g. creating a course from independent packages
    - » Standard: Only textual inclusion allowed
      - – Some tools support XInclude, however!

Structure of a manifest



- Metadata can be added to
  - » Take care where it actually applies to!
  - → Manifest
  - → Organization
  - → Item
  - → Resource
  - → File
- A (conceptual) resource consists of
  - → One or more files
  - → Dependencies on other resources
    - » E.g. a common set of files

Elements (1) organizations/organization

- A single manifest can contain several organizations
  - → This is especially useful if these represent the same content in different arrangement
    - » E.g. a subset for an introductore course
    - » E.g. tree vs. network
      - – Network is not supported, however (own specification needed)!
- An organization itself is the top-level container for the content tree (for hierarchical ones)
  - → Conceptually describes a "learning outcome"
    - » What the student should know after using the content

Elements (2)
item

- An item describes a single navigational element
  - → Conceptually a single learning resource
- Each item must/can have:
  - → Identifier (Req.): Unique (within complete manifest!) ID
  - → Identifier reference (Opt.): Reference to a resource
    - » Item without references are like folders: List of other items without any content associated with itself
  - → Title (Opt.): How this item is called/displayed
  - → Visibility info (Opt.): "Strange" ⇒ avoid it!
  - → Parameters (Opt.): These will be appended to the HRef
    - » E.g. for displaying the same form answer page with different form parameters
  - → Items (Opt.): For building the hierarchy!

Elements (3)
resources, resource, file, dependency

- Resources lists at least the resources used in the organizat.
- An individual resource need not be a single file
  - → Example: Webpage (html + image, applet, sound, video)
- Each resource must/can have:
  - → Unique identifier (Req.): Referenced from the organization
  - → Type (Req.): Hint for displaying (usually "webcontent")
  - → HRef (Opt.): URL for displaying the resource ("main" element)
  - → xml:base (Opt.): Base URL for files of this resource
- A file must have a HRef: The actual location of it
  - → Can be a local file or a web reference!
- Dependencies just refer to other resources
  - → The referenced resource is a container for files also required for this resource
    - » All files (and dependencies!) copied conceptually here

CPS Example (1)

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1" version="1.1" xmlns:... >
    <metadata>
        <schema>IMS Content</schema><schemaversion>1.1.3</schemaversion>
    </metadata>
    <organizations default="TOC1">
        <organization identifier="TOC1" structure="hierarchical">
            <title>Multimedia in webdesign</title>
            <item identifier="ITEM1" identifierref="RESOURCE1">
                <title>About multimedia</title>
                <item identifier="ITEM2" identifierref="RESOURCE2">
                    <title>Multimedia 1</title>
                </item>
                <item identifier="ITEM3" identifierref="RESOURCE3">
                    <title>Multimedia 2</title>
                </item>
                <item identifier="ITEM4" identifierref="RESOURCE4">
                    <title>Summary</title>
                </item>
            </item>
            <item identifier="ITEM5" identifierref="RESOURCE5">
                <title>Webdesign basics</title>
            </item>
        </organization>
    </organizations>
```

CPS Example (2)

```
<resources>
    <resource identifier="RESOURCE1" type="webcontent" href="intro.htm">
        <file href="intro.htm"/>
        <file href="background.gif"/>
    </resource>
    <resource identifier="RESOURCE2" type="webcontent" href="multimedia_1.htm">
        <file href="multimedia_1.htm"/>
        <dependency identifierref="RESOURCE_HOLDER"/>
    </resource>
    <resource identifier="RESOURCE3" type="webcontent" href="multimedia_2.htm">
        <file href="multimedia_2.htm"/>
        <dependency identifierref="RESOURCE_HOLDER"/>
    </resource>
    <resource identifier="RESOURCE4" type="webcontent" href="Summary.htm">
        <file href="Summary.htm"/>
    </resource>
    <resource identifier="RESOURCE5" type="webcontent" href="webdesign.htm">
        <file href="webdesign.htm"/>
    </resource>
    <resource identifier="RESOURCE_HOLDER" type="webcontent">
        <file href="image.gif"/>
        <file href="audio.mp3"/><file href="video.mpg"/>
    </resource>
</resources>
</manifest>
```

Metadata

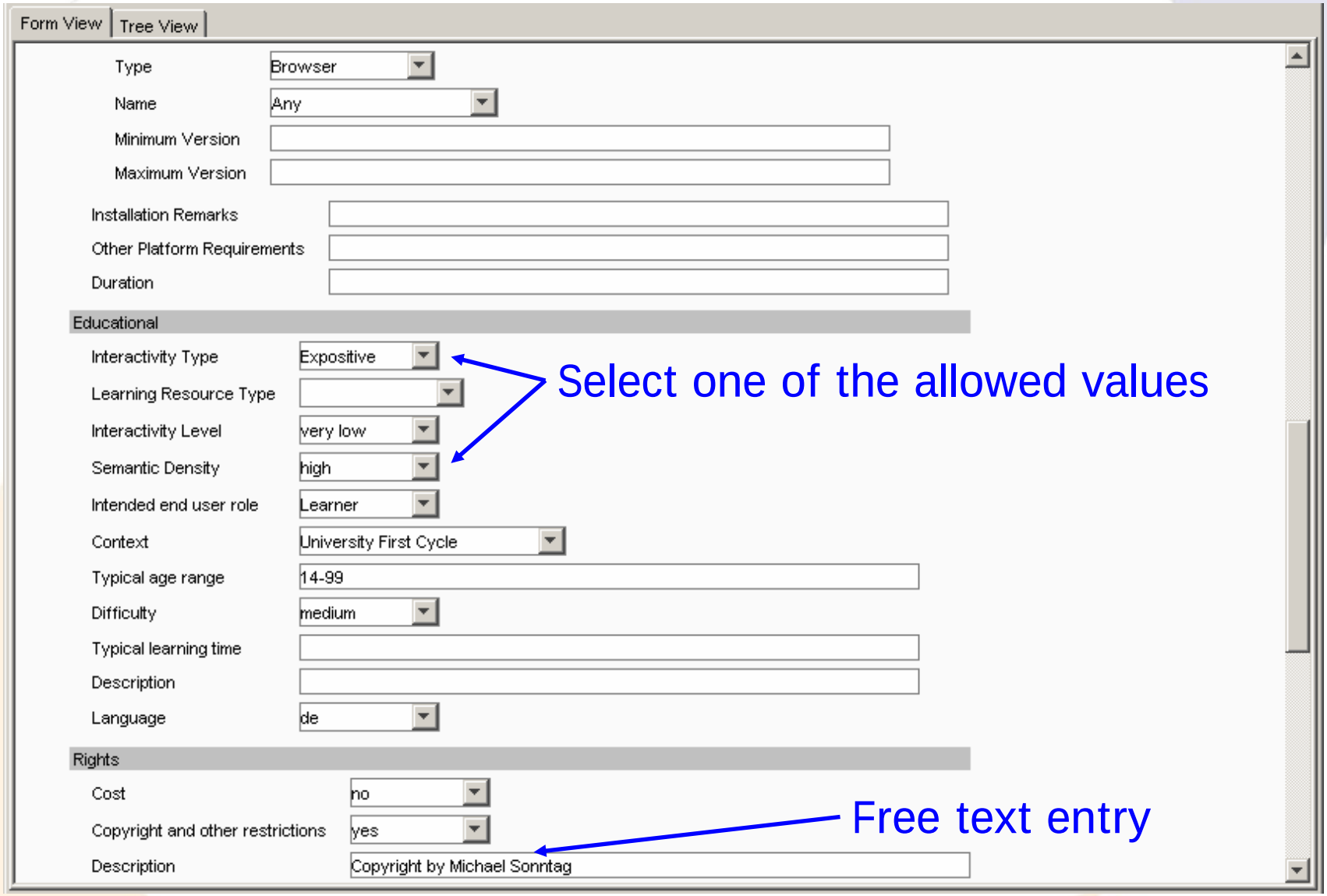
- Different standards exist
  - → IMS Learning Resource Meta-data
  - → IEEE LTSC LOM (Learning Object Metadata)
  - → BMBWK Metadata specification for electr. learning materials
- Information on the package, the resources used and the area of applicability (school types, age, ...)
  - → General and lifecycle informationen: Title, ID, version, status,...
  - → Metametadata: Metadaten schema, standard version, ...
  - → Technical information: Format, size, ...
  - → Pedagogical informationen: Interactivity type, -level, ...
  - → Rights information: Licenses, copyright, ...
  - → Dependencies/classifications: Req. resources, taxonomies, ...
  - → Comments: "User guide", ...

Directly contained within the manifest!

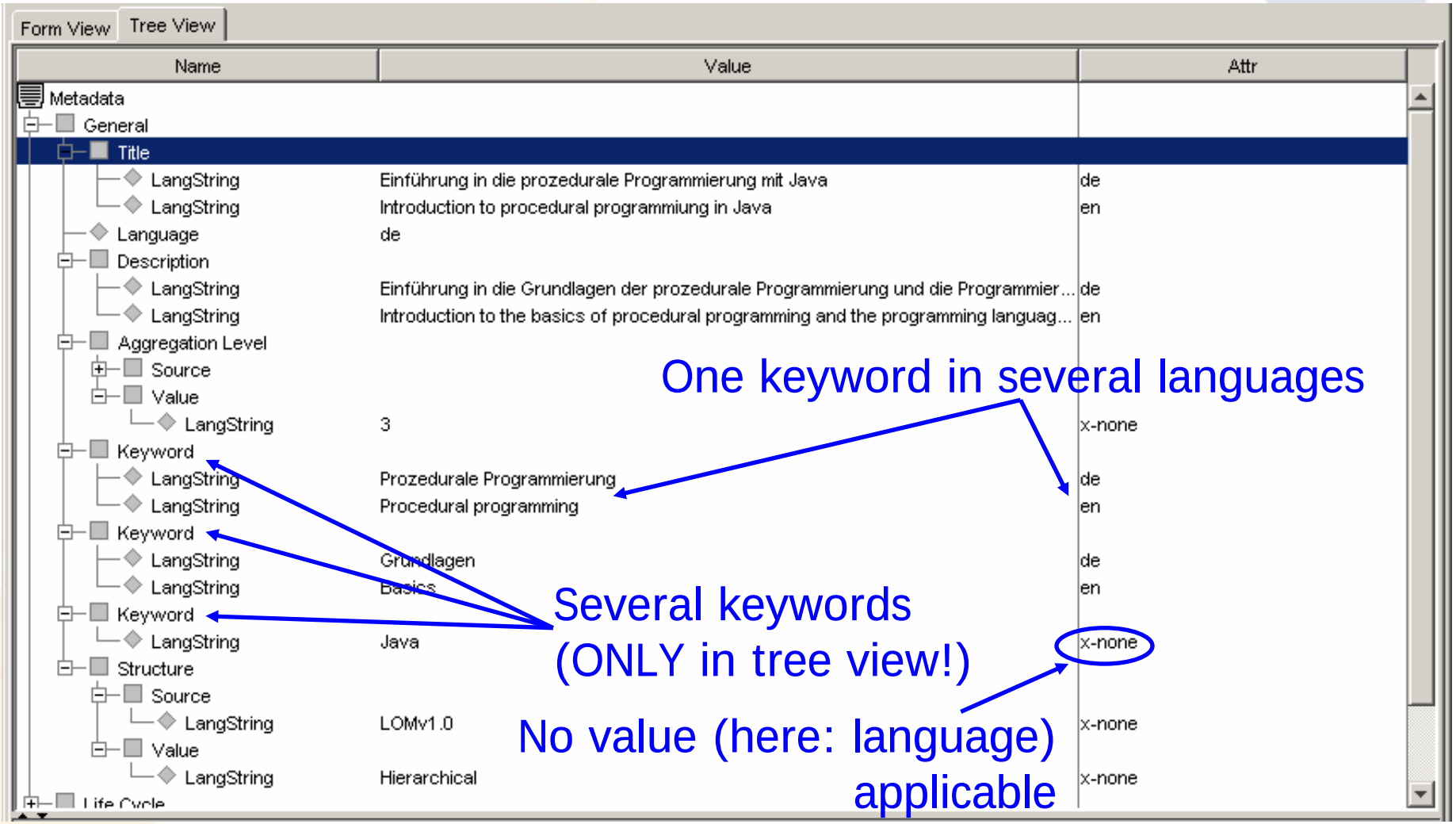
Metadata according to IMS (very similar to IEEE LOM)

- Metadata consists of (all optional):
  - → General: Title, language, description, ...
  - → Lifecycle: Version, status, contributors
  - → Metametadata: Schema, language, classification
  - → Technical: Format, size, technical requirements, platforms, ...
  - → Educational: Interactivity, context, age range, difficulty, etc.
  - → Rights: Cost (1/0), copyright (1/0), description
  - → Relation: Kind (e.g. hasPart, isBasedOn, requires), resource
  - → Annotation: Person, date, description
  - → Classification: Purpose, keyword, taxonomy, description
- Please note: E.g. Description appears several times
  - → The XML representation is the same, but the content is always different
    - » Same syntax, different semantics!

Form view of metadata



Tree view of metadata



Basic metadata

- The following elements should always be filled in for the whole course:
  - → Title: Name of the whole course
  - → Description: Short text on the course as a whole
    - » Topic, intentions, prerequisites, outcomes, ...
  - → Aggregation level: Is it a curriculum, a course, part of a course or an individual "resource"?
  - → Keywords: Some significant terms describing the content
  - → Classification: If applicable (e.g. course number at university)
  - → Structure: Hierarchical
  - → Lifecycle – Author: Who created the course
    - » More complicated if several creators or other participants!
  - → Full rights data: Trivial but important
  - → Relation or classification: Only if there is some!

Basic metadata

- → Full educational information: All elements must be filled in
  - » Please take care of the allowed values!
    - – These are from a closed taxonomy
- → Technical requirements: Required software including versions, installation remarks (if any)

- These elements should be present for all items:
  - → Title: Not necessarily as metadata; perhaps directly in CPS!
  - → Description: Short description of the item
  - → Keywords: Keywords associated with the item

- Guideline for metadata: Use the jCAPT editor and fill in everything you can imagine a value for
  - → Then use the tree view for adding more instances if needed!

Creating a course: Preparation

- Information/resources required in advance
  - → Raw material (documents, images, ...) must be available
  - → Content and size determined
  - → Structure of the content designed
  - → Base information for the metadata available/decided upon
- Possible targets for the course
  - » The course itself is independent of this; but take care of the usability of the resources!
  - → Presenting it on an Online Learning Platform
    - » E.g. WeLearn, the platform developed and used at the institute
  - → Offline presentation, e.g. on a CD-ROM
    - » Conversion e.g. through the Offline Converter
  - → Presentation on mobile devices
    - » Employing e.g. the viewer for Windows palmtops

Presentation possibilities

Online
Learning platform

WeLearn.System

CPS package

jCAPT Editor

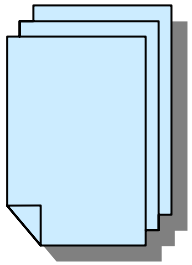
WeLearn.Mobile

Mobile
Devices

jCAPT Preview

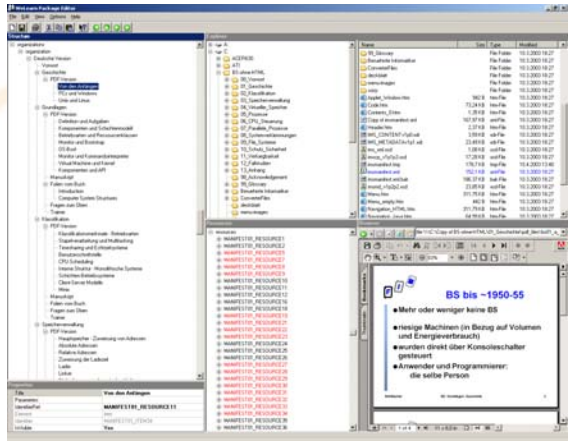
Offline Learning
CDs

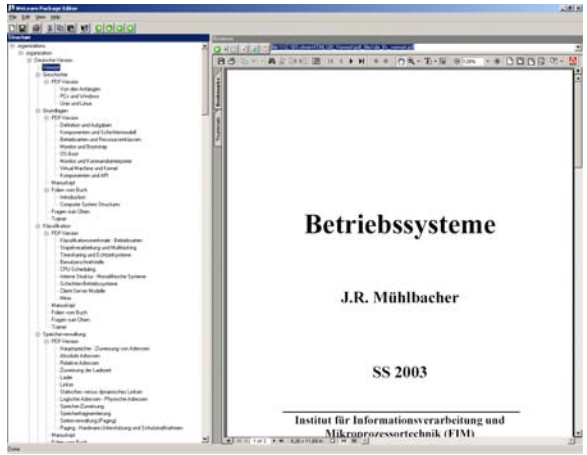
Creating a course: Assembly



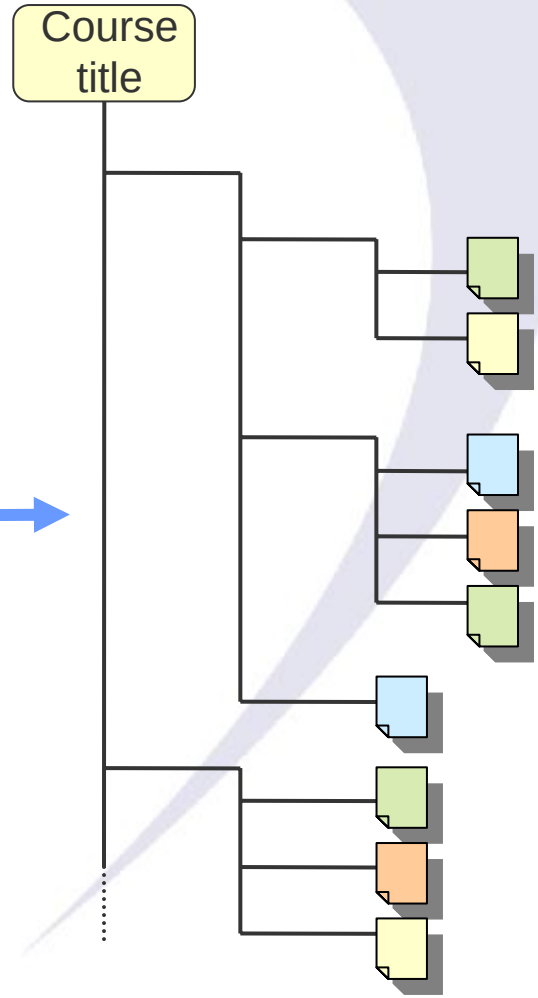
Learning materials

Structure





Editor



Course package

Creating a course: Assembly

- Using the jCAPT Editor
  - » Other products available (e.g. LRN editor)
  - → Importing the resources
    - » Adding additional files/dependencies
  - → Creating the navigation structure
    - » Using drag&drop
    - » Adding/correcting the title
  - → Inserting the metadata
    - » Separate editing mode, providing the categories available
      - – Some familiarity with the specification required!
  - → Preview of the result
- Result:
  - → Manifest according to the CPS standard
- Next step: Conversion and/or packaging into a ZIP file
  - → Preview functionality of jCAPT

Practical example

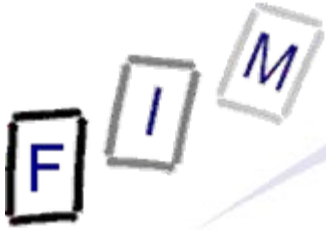
- We will create a new course from scratch
  - → The content material will be available, as explained before!
- Part of a Java course: Applets
  - → Import the resources
  - → Create an organization
  - → Define the course structure, including empty items
  - → Preview result
  - → Add some metadata
- Convert it to (D)HTML through jCAPT Preview
  - → No special configuration
    - » Look at the documentation if interested in other options!

Literature & Software

- IMS
  http://www.imsproject.org/
- IMS CPS
  http://www.imsproject.org/content/packaging/
- IMS MD
  http://www.imsproject.org/metadata/
- Reload Editor
  http://www.reload.ac.uk/
- Microsoft LRN Editor
  No longer available separate; part of Class Server now!
- WeLearn
  http://www.fim.uni-linz.ac.at/research/WeLearn/index.htm

Example courses/manifests

- Propaedeutics
  http://experience-weLearn.fim.uni-linz.ac.at/
- XML Techniques for E-Commerce
  http://www.sonntag.cc/teaching/XML_E-Commerce/Start.html
- Einführung in die prozedurale Programmierung mit Java
  http://www.sonntag.cc/teaching/JAVA-Kurs/
- Combined test cases (many submanifests, visibility, parameters, metadata, special characters, ...)
  See course homepage!



Questions?

Thank you for your attention!