Pergamos: The University of Athens DL

- Over 1 million objects originating from 8 disparate collections
  - Folklore notebooks, Papyri, Historical Archive, Byzantine music manuscripts, Theatrical photos & brochures, Informatics research papers and dissertations, Medical images, Press articles

- Heterogeneous material, in terms of content type, metadata, structure, user requirements

- Mostly digitised material, requiring detailed cataloguing
Motivation

• Simplify & speed up the cataloguing process
  • Provide effective Web-based cataloguing interfaces
  • Automate content ingestion
• Decrease development time
  • Avoid custom coding for each content variation
  • Elaborate on reusable and configurable DL modules
• Provide the means to treat content variations (distinct types of digital objects) in a unified manner
Fedora Content Models

• “Used to describe the internal structure of a group of similar Fedora objects”
• Loosely defined and specify:
  • The number and types of datastreams
  • The number and types of disseminators
Content Model Issues

• Every Fedora-based Repository follows a different approach in the use of Content Models

• Content Models provide guidelines for the creation of DOs and are not interpreted by FEDORA – humans “manually” interpret Content Models
Digital Object Prototypes

• A Digital Object Prototype (DOP) is a Digital Object Type specification
• Provides a detailed definition of a DO’s
  • constituent parts (metadata, files, relationships)
  • private and public behaviors
• Introduced in *9th European Conference on Research and Advanced Technology for Digital Libraries (ECDL) 2005*
DOPs Summary

• DOPs provide the means to generate *user-defined types* of digital objects
• DOs automatically conform to their prototype – types are interpreted by the system (no custom coding required)
• It is easy to add new prototypes or modify existing ones (no custom coding required)
DOPs and Object Orientation

• DOPs can be conceived as a realization of the Content Model notion in a OO-like fashion (we argue that “similar internal structure” = type)
• Instance-of
  • uoadl:100 is a Session object
• Encapsulation
  • DO Private & Public Behaviors
• Inheritance (under development)
DO Instances and Stored DOs

- DOs are conceived as instances of their respective prototype
  - DO instances automatically conform to the Prototype’s specifications
- Fedora Objects (Stored DOs) are treated as serializations of DO instances
  - DOPs enforce conformance of Stored DOs to their DOP at the Repository layer
Instantiation / Serialization

Instantiation

Stored Digital Object → Digital Object Prototype → Digital Object Instance

Serialization

Stored Digital Object ← Digital Object Prototype ← Digital Object Instance
A DO Instance

Session Digital Object Prototype

Public Interface (public behaviors)
- browseView
- zipView
- detailView

Private Behaviors
- Load / Serialize
- Validate
- Batch Ingest

Session Digital Object Instance

Private Data (Files, Metadata Sets, Relations)
- CONTAINER
- HQ
- EAD
- WWW
- DC
- THUMB

“uoadl:1209” Stored Digital Object
DOPs and Collections

• DOPs “live” in the context of a collection (collection pertinent scope)
• Collections are also user-defined
• Collections can contain other collections (hierarchical namespace)
• The Digital Library is the collection of all the collections
What’s in a DOP

- Metadata sets & elements specification
- File types / conversions / ingestions specification
- Allowed relationships
- Private & Public behaviors
Behaviors

• Encapsulation: Private & Public behaviors
• Private behaviors are executed by the instance “behind the scenes” on the occurrence of certain events in the instance’s lifecycle (the object itself triggers selected private functionality)
• Public behaviors are exposed to clients (clients trigger a selected functionality)
Metadata Elements

- DOPs specify:
  - the individual metadata sets (e.g. DC)
  - the individual elements (fields) that constitute each set
  - The possible mappings among elements of different sets
Metadata Elements II

• DOP-based automatic loading / serialization of metadata sets (private behavior)

• Behavioral characteristics of metadata elements:
  • isMandatory, isHidden, isRepeatable,
    • Available to high level services (enabling them to adapt to type-specific requirements of each element)
  • defaultValue, validationAction (private behavior)

• DOP-based mappings among metadata elements (private behavior)
Files Specification

• DOPs specify:
  • The individual supported files and their allowed formats
  • The information required to convert files from one format to another
  • Batch file ingestion / automatic DO creation process for each file
Files Specification II

- Automatic file conversions (e.g. TIFF to JPEG, TIFF to JPEG thumbnail)
  - Supported actions: convert, resize, convertAndResize
- Batch file ingestion / DO creation:
  - A zip file containing TIFF images generates the respected DOs
Relationships

• DOPs specify:
  • The structural relationships the object is allowed to participate (e.g. child-of, or is-parent-of)
## Collections and Prototypes in Pergamos

<table>
<thead>
<tr>
<th>Collection</th>
<th>Sub collections</th>
<th>DOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>dl</td>
<td>theatre, papyri, medical, folklore, histarch</td>
<td>-</td>
</tr>
<tr>
<td>dl.theatre</td>
<td>-</td>
<td>album, photo</td>
</tr>
<tr>
<td>dl.papyri</td>
<td>-</td>
<td>papyrus</td>
</tr>
<tr>
<td>dl.medical</td>
<td>-</td>
<td>image</td>
</tr>
<tr>
<td>dl.folklore</td>
<td>-</td>
<td>notebook, section, page</td>
</tr>
<tr>
<td>dl.histarch</td>
<td>senate</td>
<td>-</td>
</tr>
<tr>
<td>dl.histarch.senate</td>
<td>procs</td>
<td>-</td>
</tr>
<tr>
<td>dl.histarch.senateprocs</td>
<td>-</td>
<td>folder, session, page</td>
</tr>
</tbody>
</table>
### dl.histarch.* Prototypes

<table>
<thead>
<tr>
<th>Prototype</th>
<th>Metadata</th>
<th>Files</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder</td>
<td>- Qualified DC&lt;br&gt;- EAD-like&lt;br&gt;- EAD to DC mappings</td>
<td>none</td>
<td>-contains session or page</td>
</tr>
<tr>
<td>session</td>
<td>- Qualified DC&lt;br&gt;- EAD-like&lt;br&gt;- EAD to DC mappings</td>
<td>zip</td>
<td>-contains page</td>
</tr>
<tr>
<td>page</td>
<td>none</td>
<td>-HQ image&lt;br&gt;-Web image&lt;br&gt;-Thumbnail</td>
<td></td>
</tr>
</tbody>
</table>
### dl.folklore.* Prototypes

<table>
<thead>
<tr>
<th>Prototype</th>
<th>Metadata</th>
<th>Files</th>
<th>Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>notebook</td>
<td>- Qualified DC</td>
<td>none</td>
<td>-contains chapter or page</td>
</tr>
<tr>
<td>chapter</td>
<td>-Qualified DC</td>
<td>zip</td>
<td>-contains page</td>
</tr>
<tr>
<td>page</td>
<td>none</td>
<td>-HQ image</td>
<td>-Web image</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Thumbnail</td>
<td></td>
</tr>
<tr>
<td>Prototype</td>
<td>Metadata</td>
<td>Files</td>
<td>Relations</td>
</tr>
<tr>
<td>-----------</td>
<td>----------------</td>
<td>----------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>papyrus</td>
<td>Qualified DC</td>
<td>-Original Image</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Processed image</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Web image</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Thumbnail</td>
<td></td>
</tr>
</tbody>
</table>
### dl.theatre.* Prototypes

<table>
<thead>
<tr>
<th>Prototype</th>
<th>Metadata</th>
<th>Files</th>
<th>Relations</th>
</tr>
</thead>
</table>
| album     | - Custom (theatrical play metadata)  
            - DC  
            - Custom to DC mappings | none        | contains photo         |
| photo     | Qualified DC  
            - Original Image  
            - Web image  
            - Thumbnail | none        | none                   |
A hierarchy of DO instances
Session DOP

<prototype id="Session">
  <MDSets>
    <!-- Metadata definition -->
    <MDSets id="dc">
      <label lang="en">Dublin Core Metadata</label>
      <datastream id="dc" MDtype="descriptive" loader="gr.uoa.dl.core.xml.StandardLoader" serializer="gr.uoa.dl.core.xml.DCSerializer"/>
      <fields>
        <field id="dc:date" isMandatory="true" isRepeatable="false" isHidden="false" validation="gr.uoa.dl.core.validation.SimpleDateFormat">
          <label lang="en">Date</label>
        </field>
        <field id="dc:identifier_physical" isMandatory="true" isRepeatable="false" isHidden="true">
          <label lang="en">Call number</label>
        </field>
        ...
      </fields>
    </MDSets>
    <MDSets id="ead">
      <label lang="en">EAD like Metadata</label>
      <datastream id="EAD" MDtype="descriptive" loader="gr.uoa.dl.core.xml.StandardLoader" serializer="gr.uoa.dl.core.xml.EADSerializer"/>
      <fields>
        <field id="did:unitid"/>
        ...
      </fields>
    </MDSets>
    <mappings>
      <mapping id="identifier">
        <from="ead.did:unitid"/>
        <to="dc:dc:identifier_physical"/>
      </mapping>
      ...
    </mappings>
  </MDSets>
</prototype>

<files>
  <!-- Files definition -->
  <file id="zip" type="container" datastream="ZIP">
    <label lang="en">ZIP file</label>
    <mime-type id="application/zip"/>
    <hatchingast targetTypeId="page" targetFileId="highQuality"/>
  </file>
  
  <files>
    <relations>
      <!-- Relationships definition -->
      <structure>
        <childType id="age"></childType>
      </structure>
      <references allowCustomURL="false" allowCustomDO="true"/>
    </relations>
    <behaviours>
      <!-- Behaviours definition -->
      <schemes>
        <scheme id="browseView" isDefault="true">
          <label lang="en">Short View</label>
          <element id="MDSets.dc:dc:identifier"/>
          <element id="MDSets.dc:dc:title"/>
          <element id="MDSets.dc:dc:date"/>
        </scheme>
        <scheme id="zipView">
          <label lang="en">Short View</label>
          <element id="MDSets.dc:dc:title"/>
          <element id="files.zip"/>
        </scheme>
        <scheme id="detailView">
          <label lang="en">Detail View</label>
          <element id="MDSets.dc:dc:identifier"/>
          <element id="MDSets.dc:dc:title"/>
          <element id="MDSets.dc:dc:identifier_physical"/>
          <element id="MDSets.dc:dc:date"/>
          ...
        </scheme>
      </schemes>
    </behaviours>
  </files>
<files>
  <file id="highQuality" type="primary" datastream="HQ">
    <label lang="en">High Quality Image</label>
    <mime-type id="image/tiff">
      <conversion target="web" task="convertAndResize" hint="scale:0.6, quality:0.7">
        <mime-type id="image/jpeg" converter="gr.uoa.dl.core.conv.ImageConverter"/>
      </conversion>
    </mime-type>
  </file>
  <file id="web" type="derivative" datastream="WEB">
    <label lang="en">Web Image</label>
  </file>
  <file id="thumb" type="derivative" datastream="THUMB">
    <label lang="en">Thumbnail Image</label>
  </file>
</files>
## On Behaviors

<table>
<thead>
<tr>
<th>FEDORA Behaviors</th>
<th>Prototype-based Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>All behaviors are public</td>
<td>Distinction between private and public</td>
</tr>
<tr>
<td>Are defined in each DO separately</td>
<td>Are defined <strong>once and in one place</strong> (in the Prototype)</td>
</tr>
<tr>
<td>Operate on the datastreams</td>
<td>Operate on the atomic elements of a DO</td>
</tr>
<tr>
<td>Invoked directly on the DO</td>
<td>Invoked as in Object Oriented Dynamic Method Dispatch</td>
</tr>
<tr>
<td>Require the a priori existence of datastreams</td>
<td>Instantiation (new, “empty” DO)</td>
</tr>
<tr>
<td>Exposed as Web services</td>
<td>Web services will be of use after the DL has been built</td>
</tr>
</tbody>
</table>
Next steps

• Implement OO inheritance (type A derives from type B)
• Offline prototype validation process (in beta phase)
• Release current DOPs implementation under an open source license (in the near future)
• Develop a repository-independent DOPs implementation
References
