Towards a Unified Runtime Model for Managing Networked Classes of Digital Objects

Kostas Saidis and Alex Delis
{saiko,ad}@di.uoa.gr

Department of Informatics & Telecommunications
University of Athens

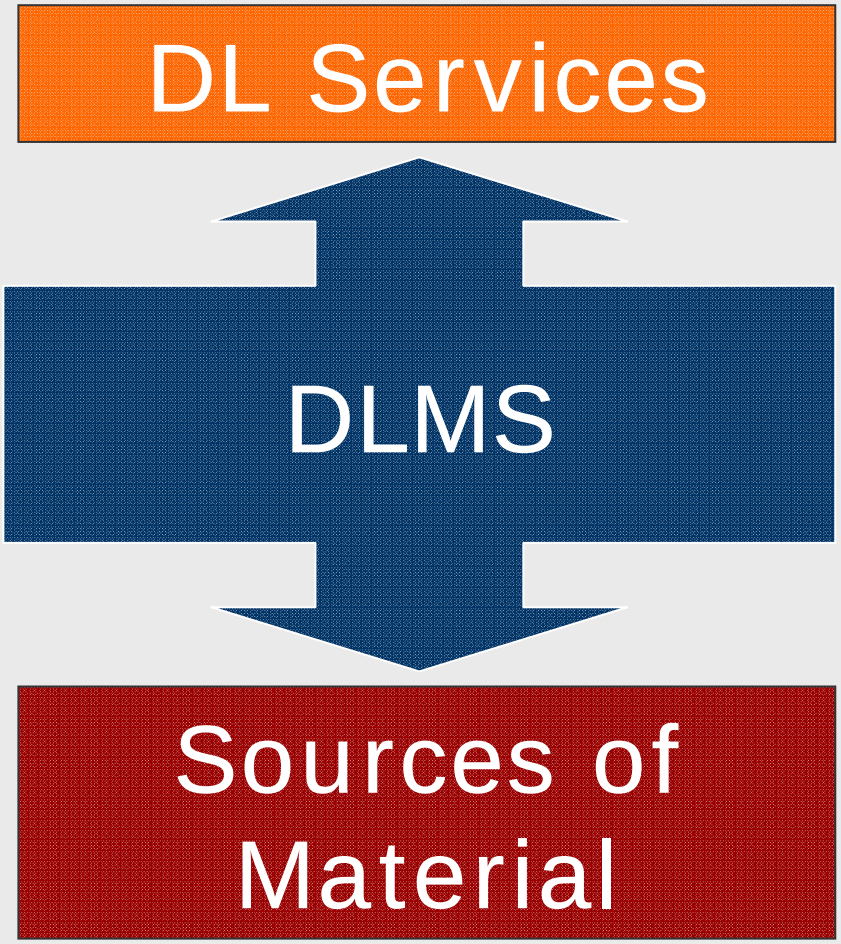
2nd DELOS Workshop on Foundations of Digital Libraries
ECDL 2007, Budapest, Hungary, September 20th 2007

The Goal

- Develop a general-purpose, reusable system that can act as a common runtime for developing any DL
- In DELOS terms, we discuss how to develop a DL Management System!

- What? Why? How?

The DLMS



DL Application Logic
Usage DO Model

DLMS
Logical/Runtime DO Model

DOStore
Physical/Storage DO Model

Highway to Hell

- Do we develop DLs in the **COBOL way**?

The COBOL Way

- Ad-hoc, tailor-made solution to specific use cases and scenarios
- Build a DL that supports a specific:
  - storage solution
  - set of digital material types
  - service provision environment
- Rebuild the DL when any of these change (new user requirement, new technology, etc)

Stairway to Heaven

- A Unified Runtime Model for DLs

A Unified DL Runtime Model

- Handle DL-specific deployment / development variations uniformly
- Operate atop heterogeneous storage solutions
- Handle semantically diverse types of material in a uniform manner
- Allow DL Application Logic to synthesize digital object information in any service provision environment of choice

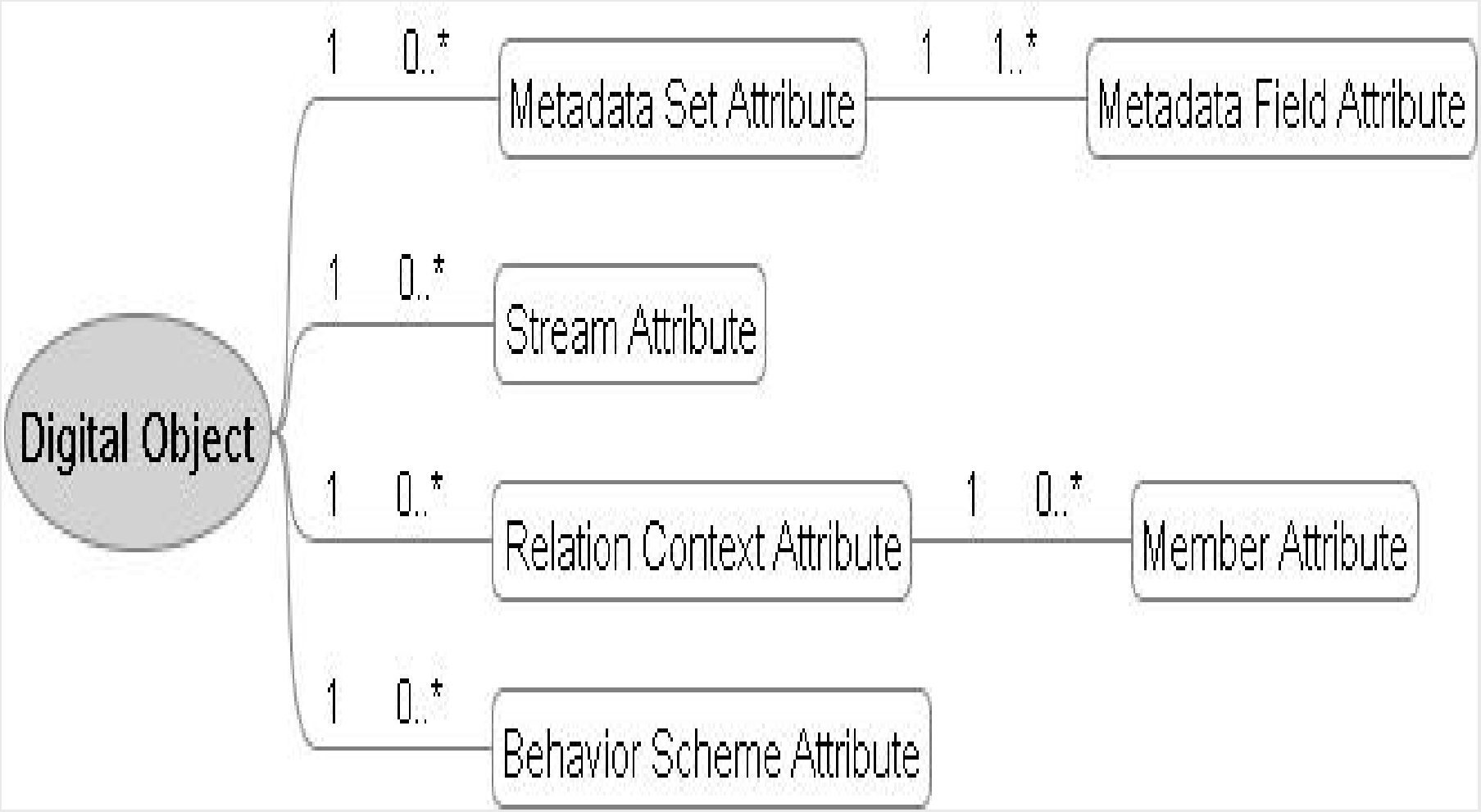
How to get there

- We identify the critical attributes of an effective DL Runtime Model:
  1. It should be based on a storage-independent logical model
  2. It should operate in a service-neutral manner
  3. It should provide powerful conceptual modeling capabilities to the DL designer
  4. It should be expressive and easy to use (productive) for the DL developer

1. Storage Independence

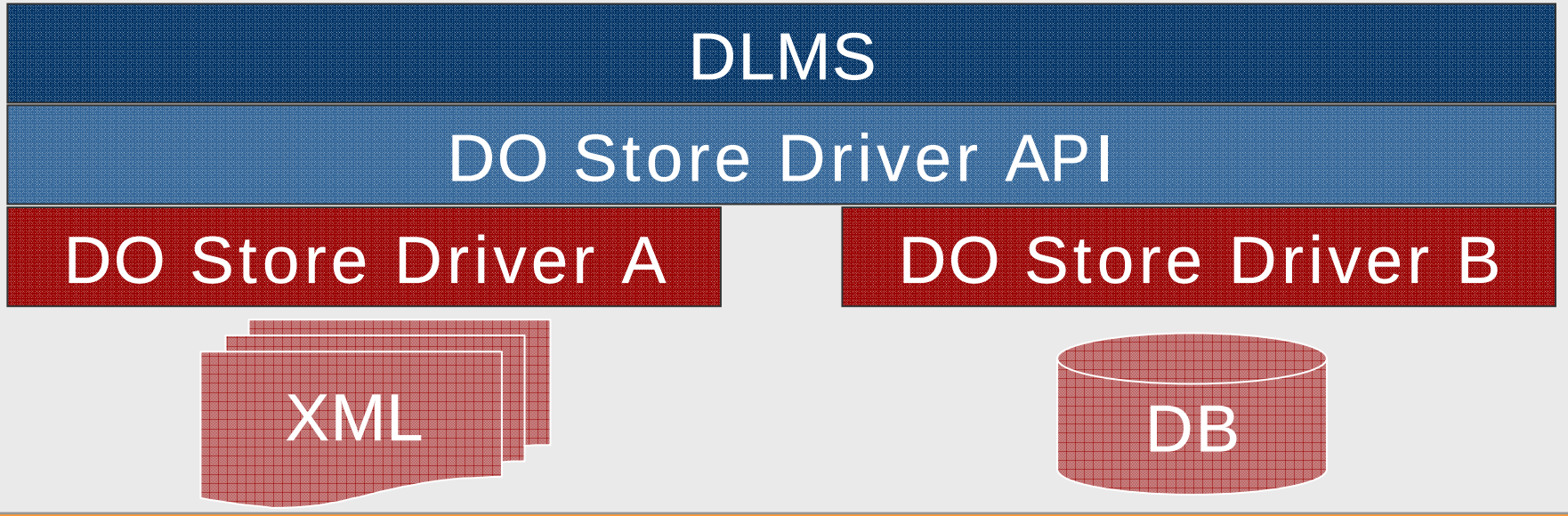
- A storage-independent Logical Model:
  - Allow DLMS to operate atop any DO Stores (databases, XML repositories, etc)
  - Offer a unified Logical View of heterogeneously stored DOs (local, remote, whatever)
  - Move DOs between DO Stores
  - DO Interoperation/Integration-ready!

Our Proposed Log. Model

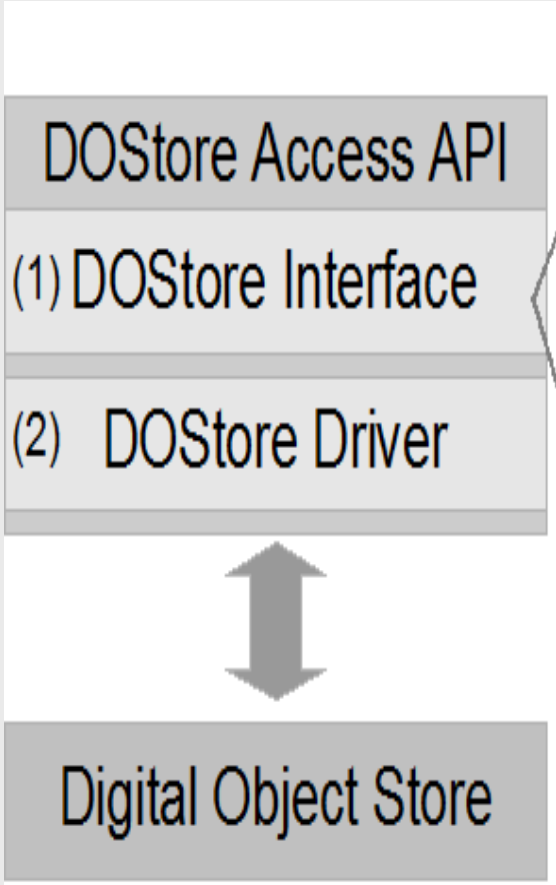


Masking Out Storage Variations

- We use the DO Store Driver notion
- “Translate” Diverse Physical Models into a unified Logical Model



DO Store Driver API




```
interface DOStore:
 boolean objectExists(doId);
 void addNewDO(dopId,doId);
 String addNewDO(dopId);
 String getDOPId(doId);
 MultilingualValue[] loadMetadataSet(doId,mdSetId);
 void saveMetadataSet(doId,mdSetId,fieldValues);
 String[] loadRelationMembers(doId,relId);
 void saveRelationMembers(doId,relId,ids);
 InputStream loadStreamContent(doId,streamId);
 void saveStreamContent(doId,streamId,stream,MIME);
 void saveStreamURL(doId,streamId,url);
 String getStreamURL(doId,streamId);
 String getStreamMIMEType(doId,streamId);
 long getStreamLength(doId,streamId);
```

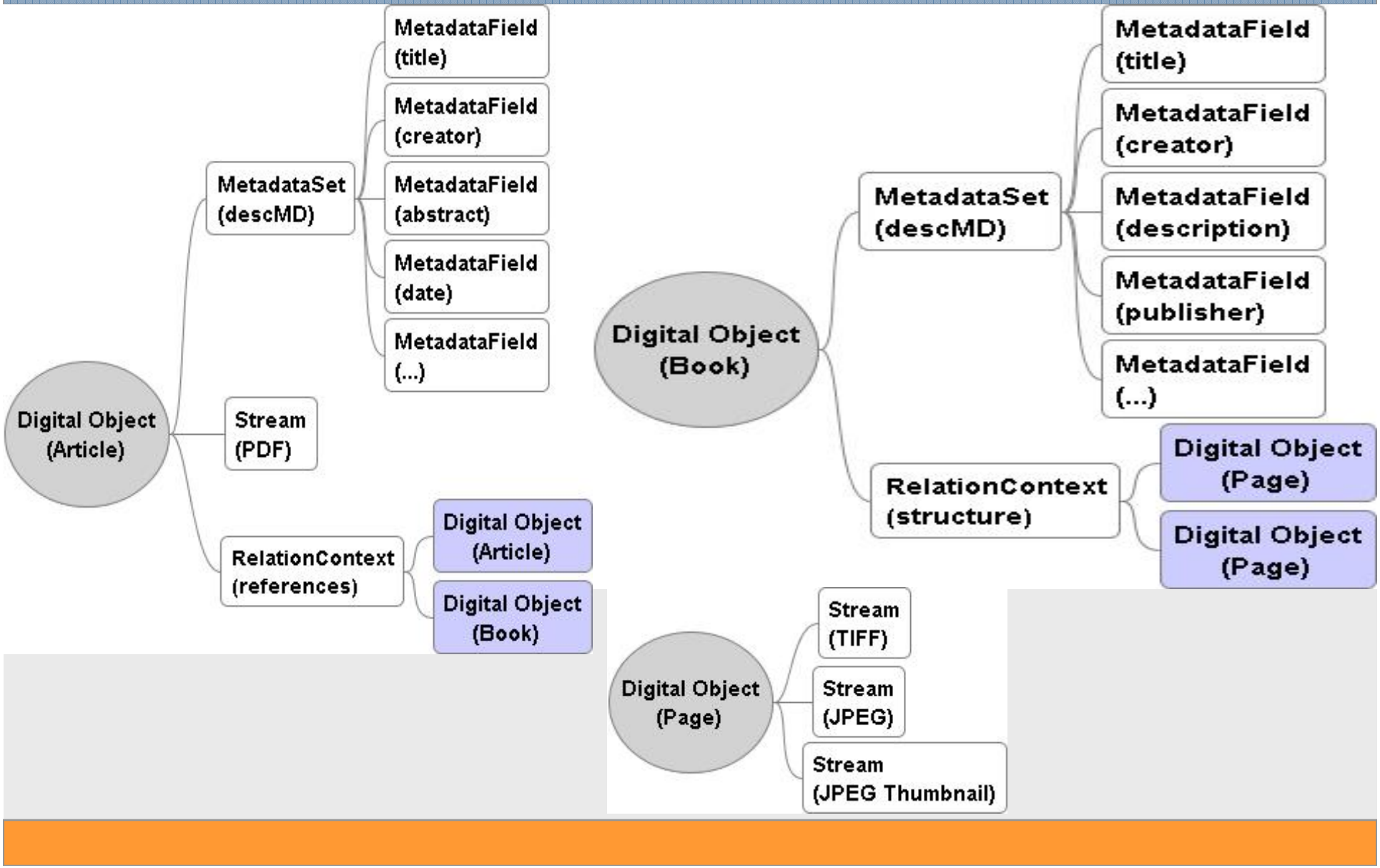
2. Modeling Power

- Represent semantically diverse DOs in a uniform manner (using a single “language”)
- Allow DL designer to use all four established abstraction principles:
  - Aggregation/Decomposition
  - Grouping/Individualization
  - Classification/Instantiation
  - Generalization/Specialization

DO Classes/Types

- DOs as compound entities comprised of metadata sets, streams, relation contexts and behavior schemes
- A self-contained definition of these attributes, viewed as DO meta-information, provides a DO Class/Type
  Digital Object Prototypes (ECDL 2005 & 2006, DLIB 5-6/2007)
- At runtime, DOs are treated as instances of DO Classes (automatically)
- Support Aggregation, Grouping, Classification/Instantiation

Example

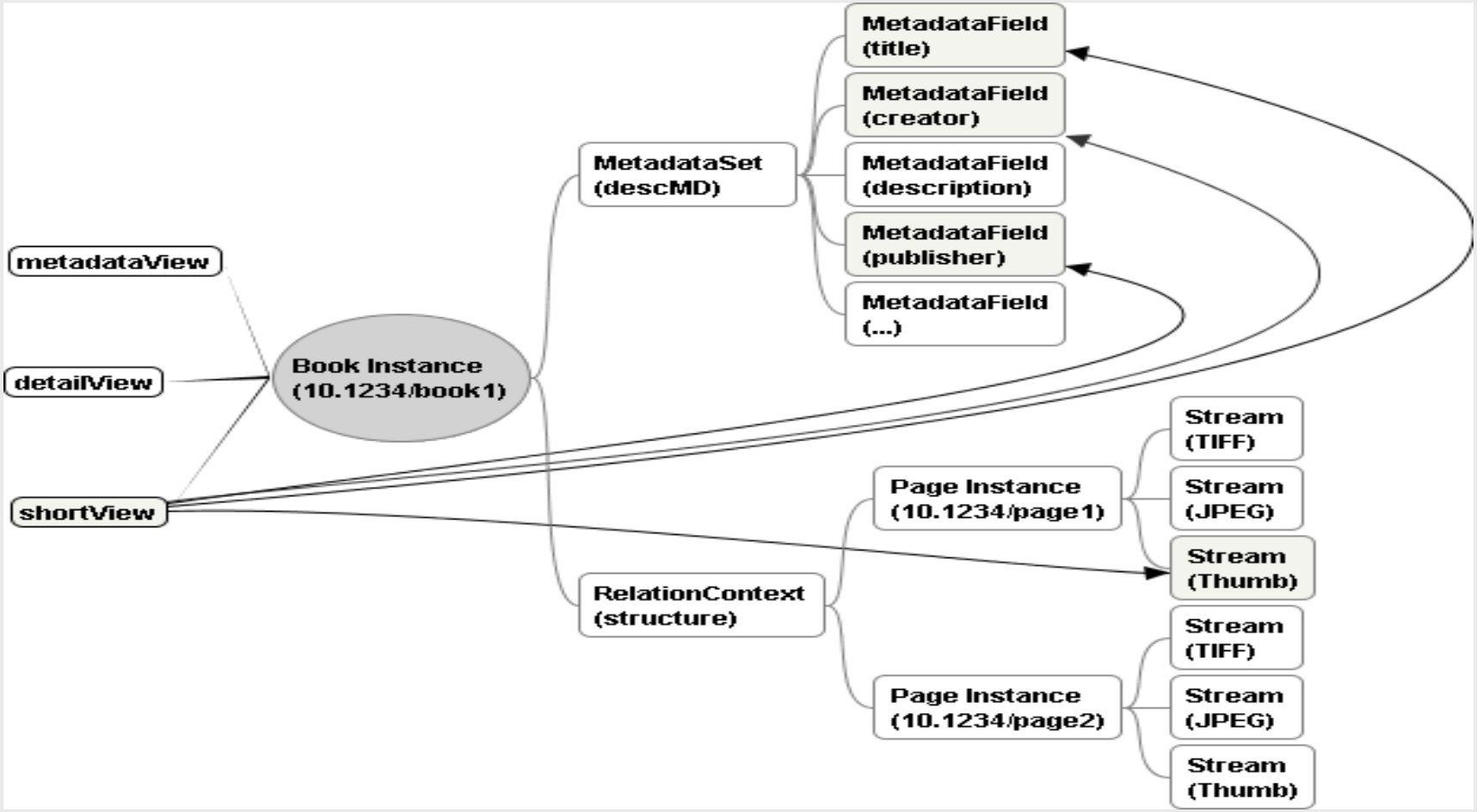

Digital Object (Article)
MetadataSet (descMD)
MetadataField (title)
MetadataField (creator)
MetadataField (abstract)
MetadataField (date)
MetadataField (...)
Stream (PDF)
RelationContext (references)
Digital Object (Article)
Digital Object (Book)
Digital Object (Book)
MetadataSet (descMD)
MetadataField (title)
MetadataField (creator)
MetadataField (description)
MetadataField (publisher)
MetadataField (...)
RelationContext (structure)
Digital Object (Page)
Digital Object (Page)
Digital Object (Page)
Stream (TIFF)
Stream (JPEG)
Stream (JPEG Thumbnail)

3. Service Neutrality

- A Runtime Environment that realizes the Logical Model:
  - Employ appropriate structures to stage DO information/data at runtime
  - Expose an API to access/modify such runtime structures
  - Cycle: Load / Wrap / Access & Modify / Unwrap / Serialize
  - Let the services decide the service provision details (e.g. protocols, user interfaces, etc)

Service Neutral DO Behavior

- Behavior Schemes: Projections on a DO's structure/namespace



4. Expressiveness

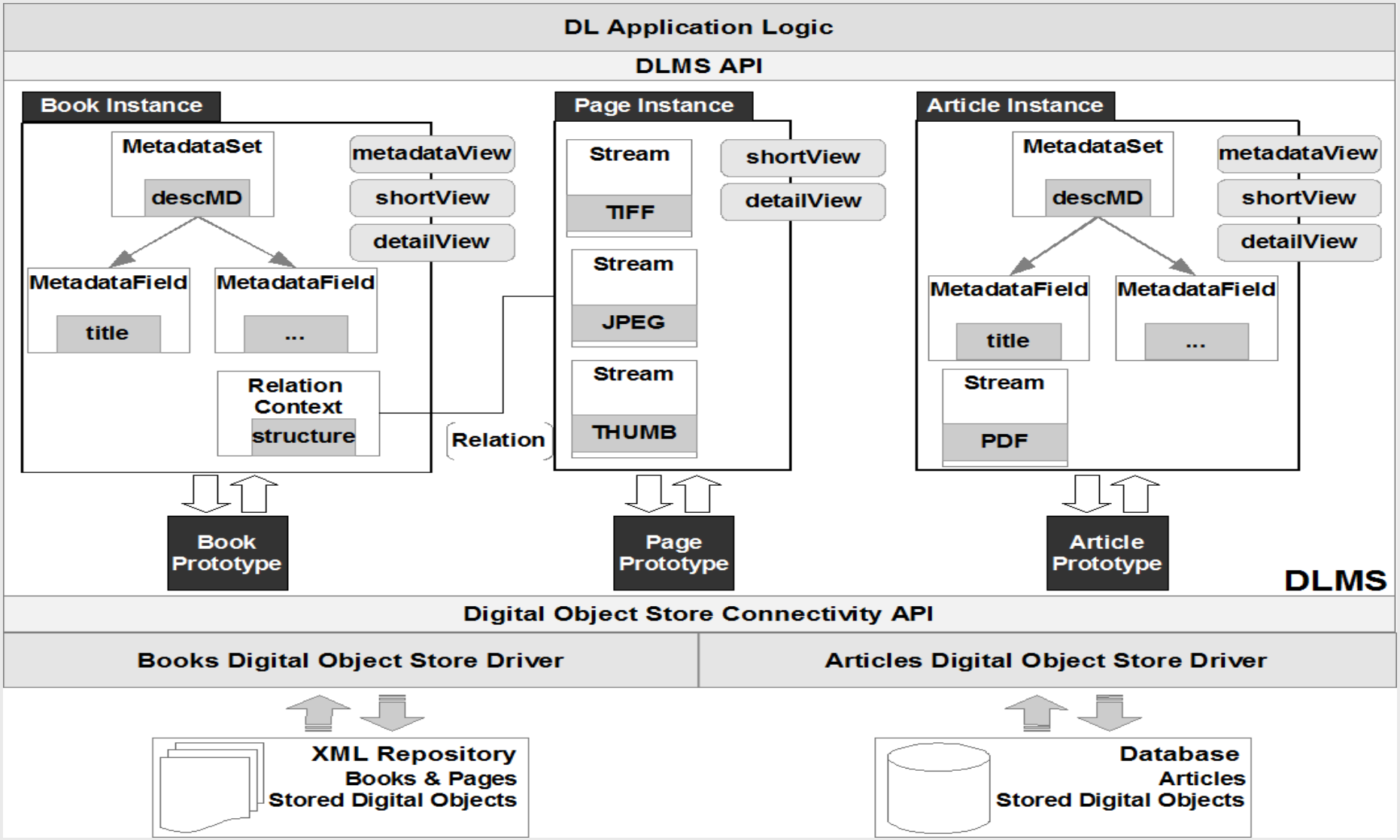
- Do more with less!
- A domain-specific DO Management “language”
- RDBMS acts as an SQL Interpreter (for the DB application developer)
- DLMS should be a DOML Interpreter (for the DL application developer)

Programming Example

```
1. DOInstance painting = DOPs.getInstance("painting","1000",paintingDriver)
2. MetadataSet mdSet = painting.getMetadataSet("descMD")
3. MetadataField title = mdSet.getField("title")
4. String value = title.getValue("en")

5. DOInstance book = DOPs.getInstance("book","10.1234/book1",bookDriver)
6. RelationContext relation = book.getRelationContext("structure")
7. foreach(id in relation.relationMembers())
8.  DOInstance page = DOPs.getInstance("page",id,bookDriver)
```

Our Proposal for the DLMS



Discussion

- Ref. Model
  - DLs should be viewed as applications build with the DLMS
  - The model will be finalized not when there is nothing more to add but when there is nothing more to take away
- DO Classes/Types
  - Think of them as guides to load/manage/store data at runtime – A DOP is not a way to store things
  - A stored digital object can have multiple types at runtime
- Future Work
  - DO Integration/Interoperation: DO Store Drivers can make DLs appear as remote sources of each other
  - Indexing / searching contradicts storage-independence
  - DOPs Inheritance – Reuse and Polymorphism

Thank God it's Over!

- Thank you for your patience!
- Comments? Questions?

- Email: saiko@di.uoa.gr
- An older version of our approach in action: http://pergamos.lib.uoa.gr/
- Public Release of DOPs framework: http://www.dops-framework.net