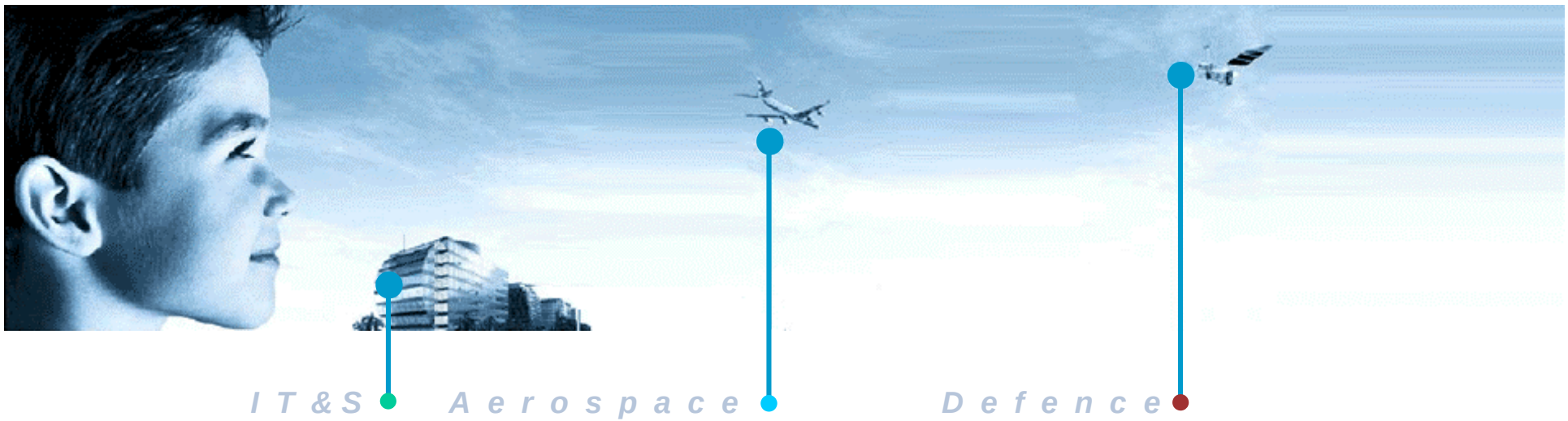
Data Distribution Service - DLRL



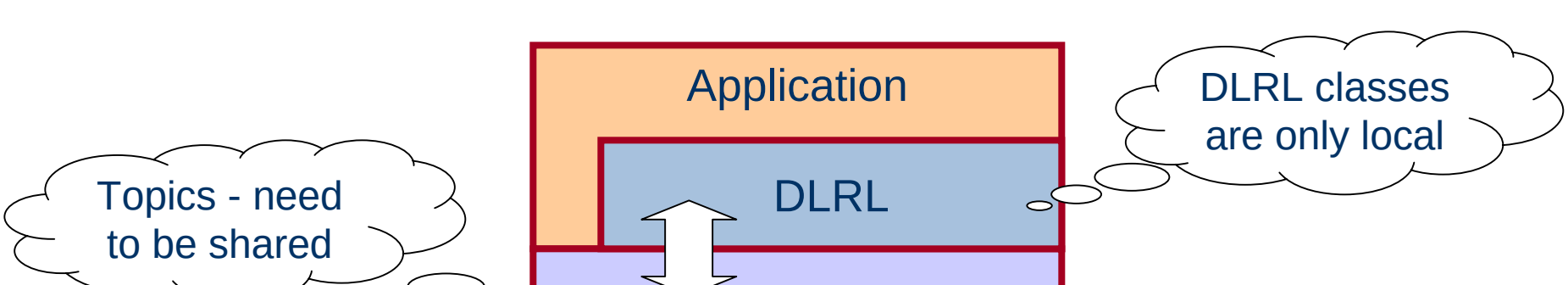
Virginie.Watine@fr.thalesgroup.com

13 July 2004

THALES

DLRL = Data Local Reconstruction Layer

- Purpose = provide a more 'natural' access to data
  - Object-oriented - using native language constructs
  - Management of graphs of objects
- Can be built on top of DCPS



- Object model can only be local (no need for a global one)

What is DLRL? (2/2)

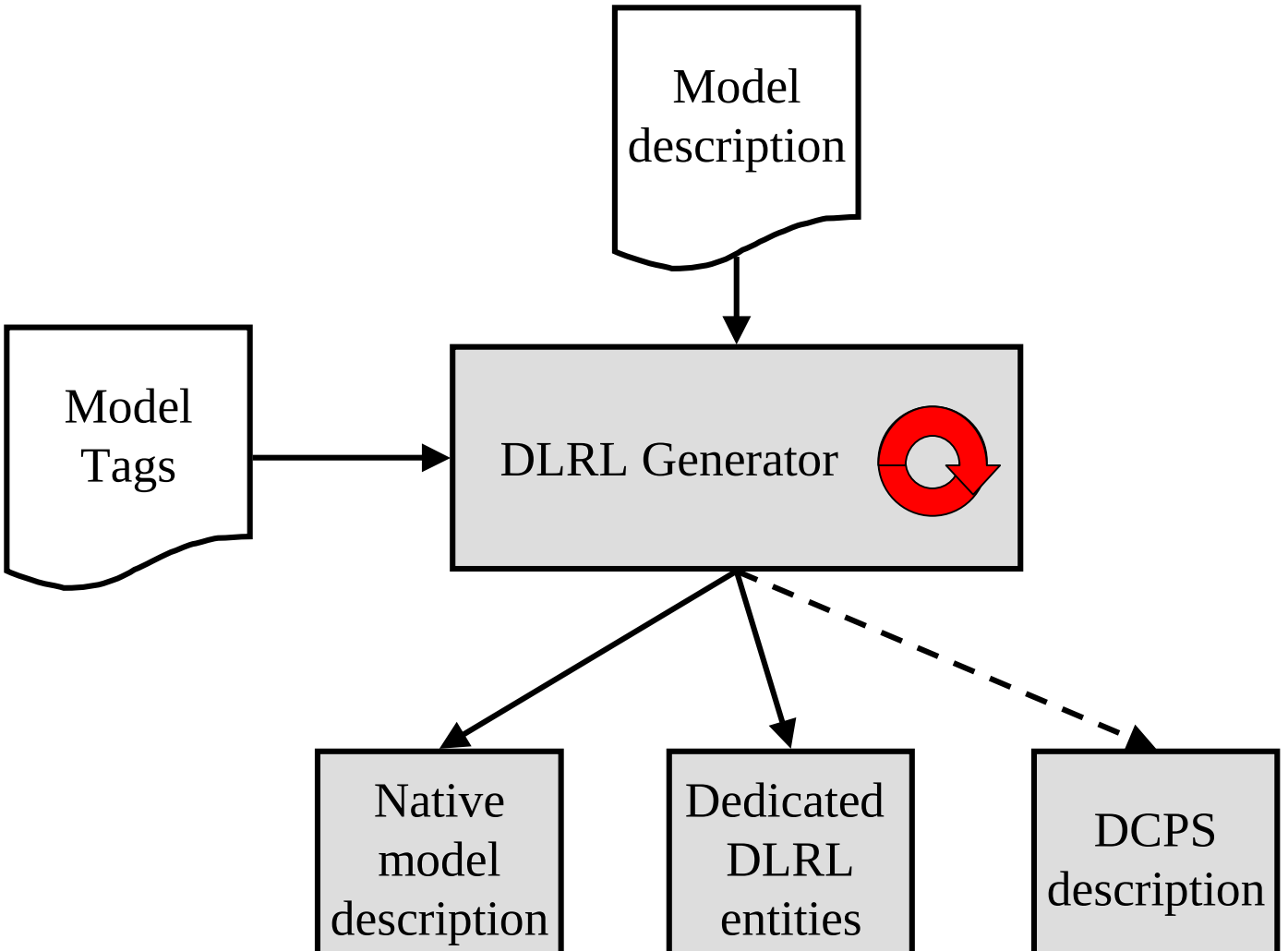
With DLRL, the application developer can:

- Describe classes of objects with their methods, data fields, relations
- Get some of those data members attached to DCPS entities
- Manipulate these objects (i.e., create, modify, delete) using the native language constructs that will, under the scene, activate accordingly the attached DCPS entities
- Have those objects refreshed transparently
  - Means are provided to manage potential conflicts between incoming updates and application-made modifications

Relies on a mapping between DLRL objects and DCPS entities

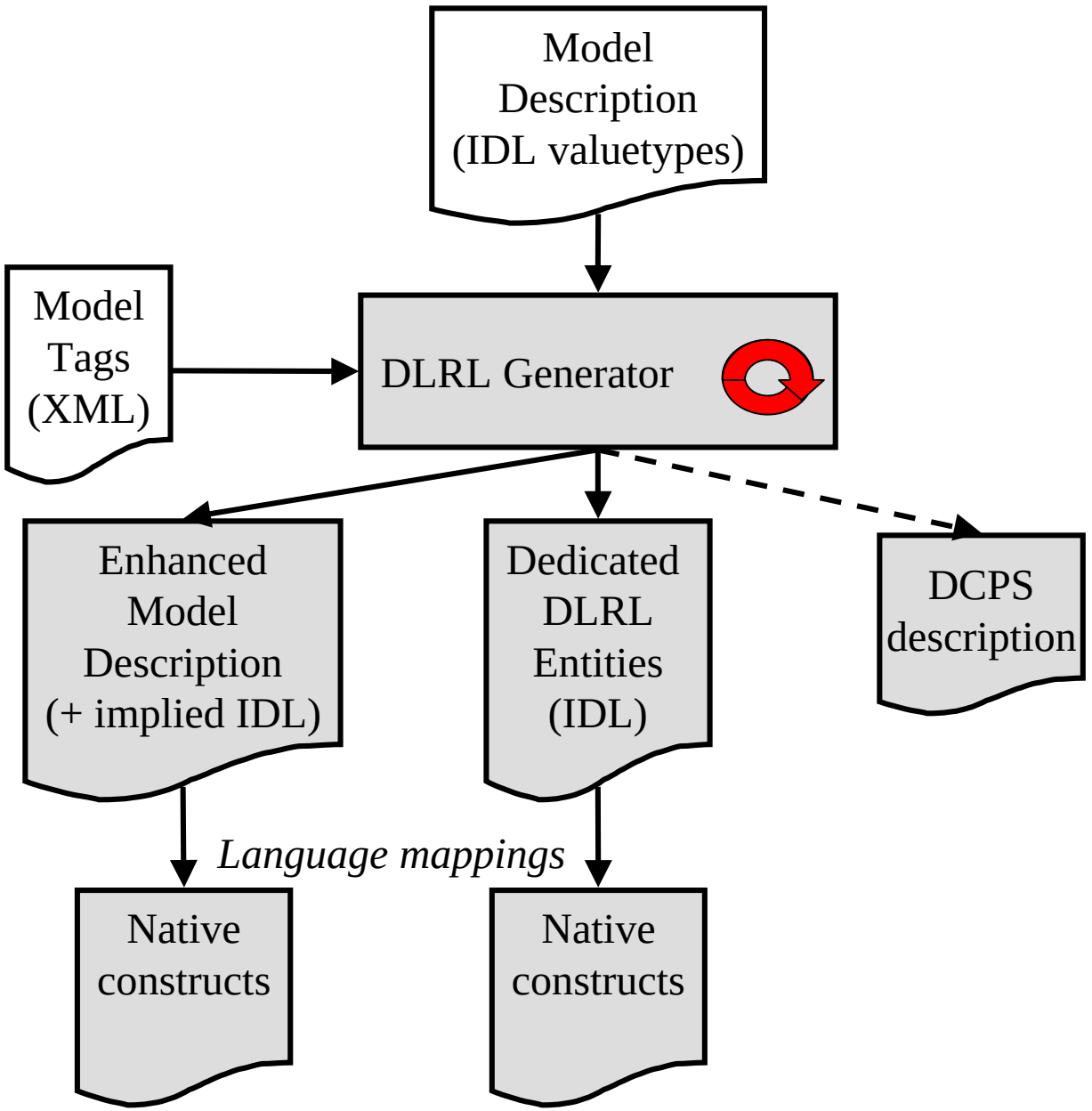
- Generated from a set of tags that annotate the DLRL model

How is the Mapping Indicated?



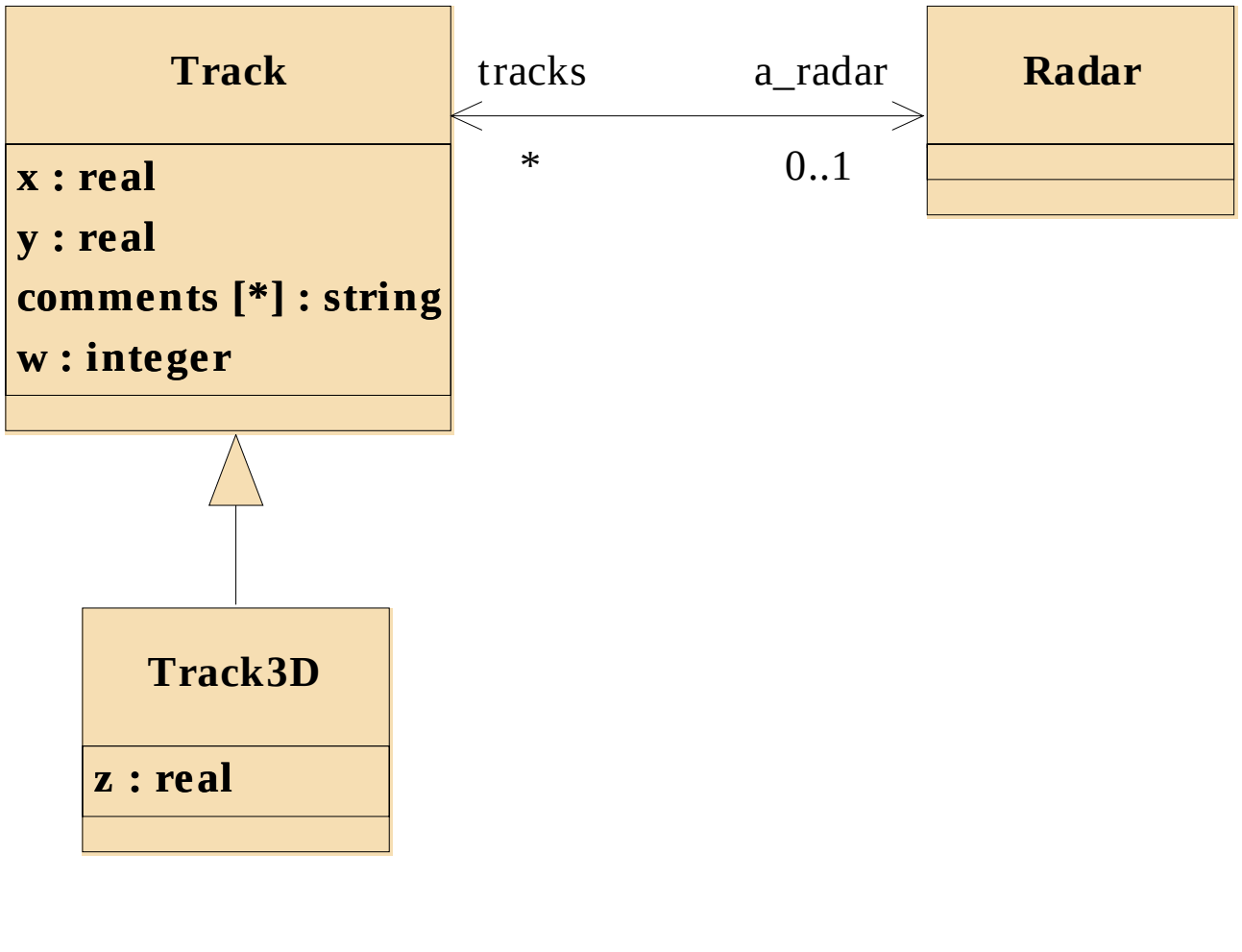
THALES

How is the Mapping Indicated (IDL PSM)?



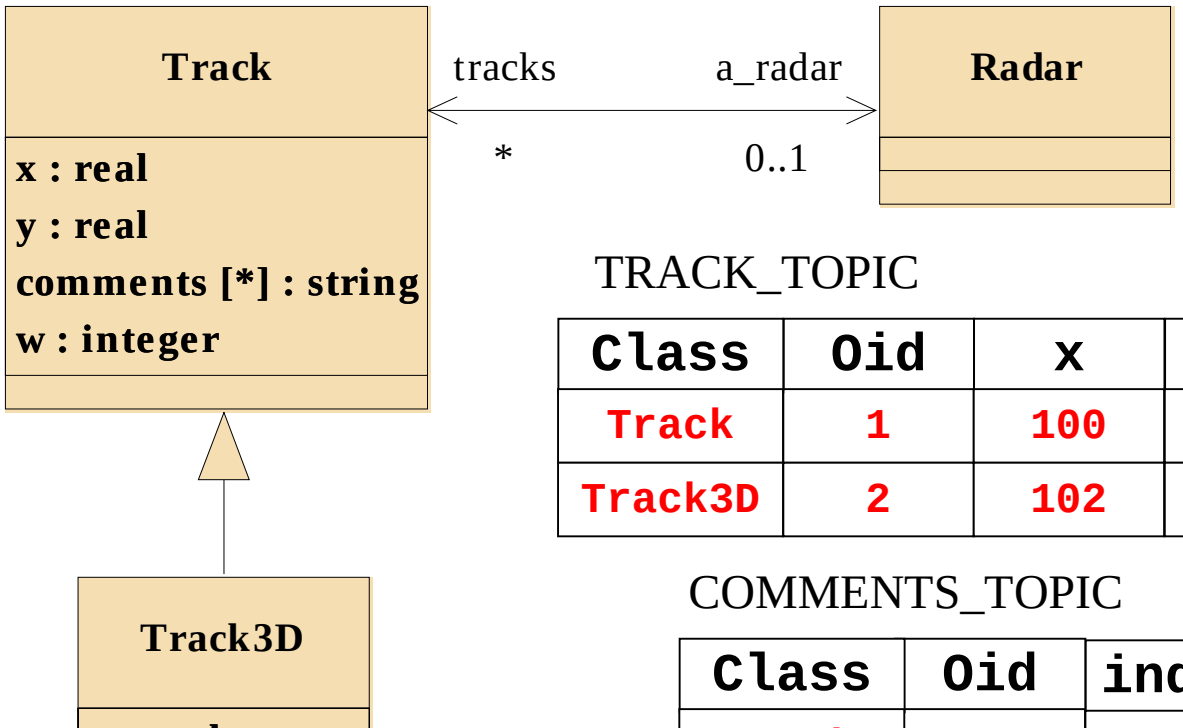
THALES

- DLRL objects are plain objects with:
  - Data members
  - Methods
- Only the data members are relevant to data distribution; they can be:
  - Attribute -> value
  - Relation -> DLRL object
  - Plain local data members (i.e., not involved in data distribution) are also supported
- Attributes and relations can be mono/multi valued
  - List or Map as collection types
- Two relations can form an association (inverse)
- DLRL classes can be organised by single inheritance



- The structural mapping describes the links between DLRL objects and DCPS data
  - E.g., in which 'topic' a given DLRL attribute will be put...
  - Close to Object to Relational mapping
  - Can handle existing DCPS models
- Set on an attribute basis
  - Needed for multi-valued attributes
    - collections as separate 'topics'
  - Needed also:
    - to handle inheritance nicely (extension tables)
    - for better flexibility (e.g., savings in memory and partial updates)
  - 1 DLRL class -> several DCPS topics
- Default rules exist

Example Mapped to DCPS



RADAR_TOPIC

Oid
11

TRACK_TOPIC

Class	Oid	x	y	radar
Track	1	100	200	11
Track3D	2	102	201	11

COMMENTS_TOPIC

Class	Oid	index	comments
Track	1	0	a comment
Track	1	1	another comment

T3D_TOPIC

Oid	z
2	300

RADAR-TRACKS_TOPIC

R_Oid	index	T_Class	T_Oid
11	0	Track	1
11	1	Track3D	2

THALES

Cache

- Isolates a set of related objects that will be managed consistently
- Corresponds to one DCPS Publisher and/or one DCPS Subscriber

CacheAccess

- Gathers a consistent set of object copies (clones)
- Allows to perform a consistent set of modifications (Write mode) or take a consistent picture for reading (Read mode)

ObjectRoot

- Root for any DLRL object

ObjectReference

- = what is strictly needed to designate an object (OID + class)

ObjectHome

- Root of typed manager of a set of objects (inside a Cache)

Collection

- Root for *multi-valued* attributes or relations
- two basis
  - List
  - Map

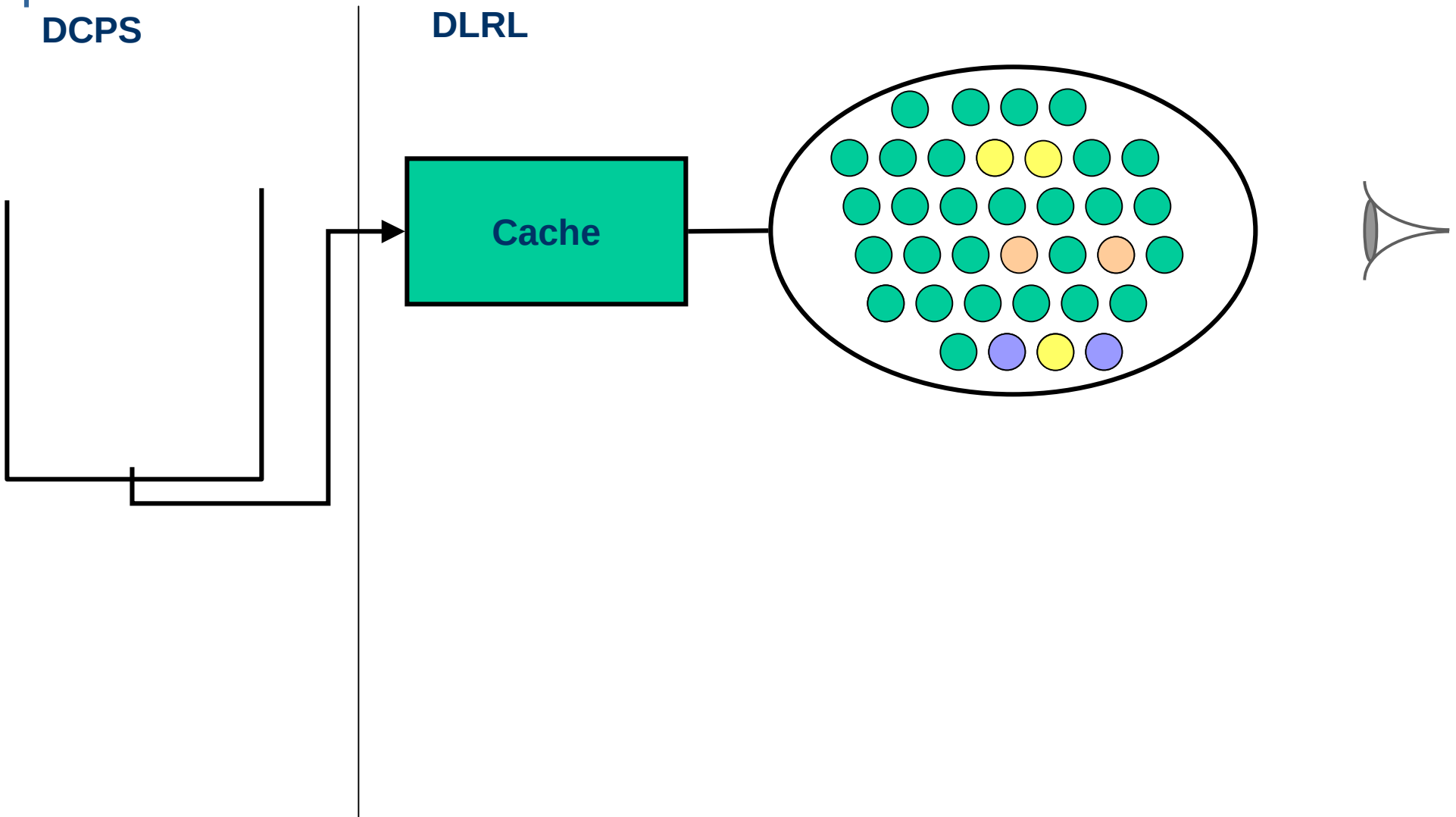
Relation

- Root for *relations* (mono or multi-valued)
- Represents an 'association end'
- Two relations can be managed consistently
  - To model a real association
  - When one relation is modified, the other is changed accordingly under the scenes

(Some DLRL entities are abstract roots to support typed derivations)

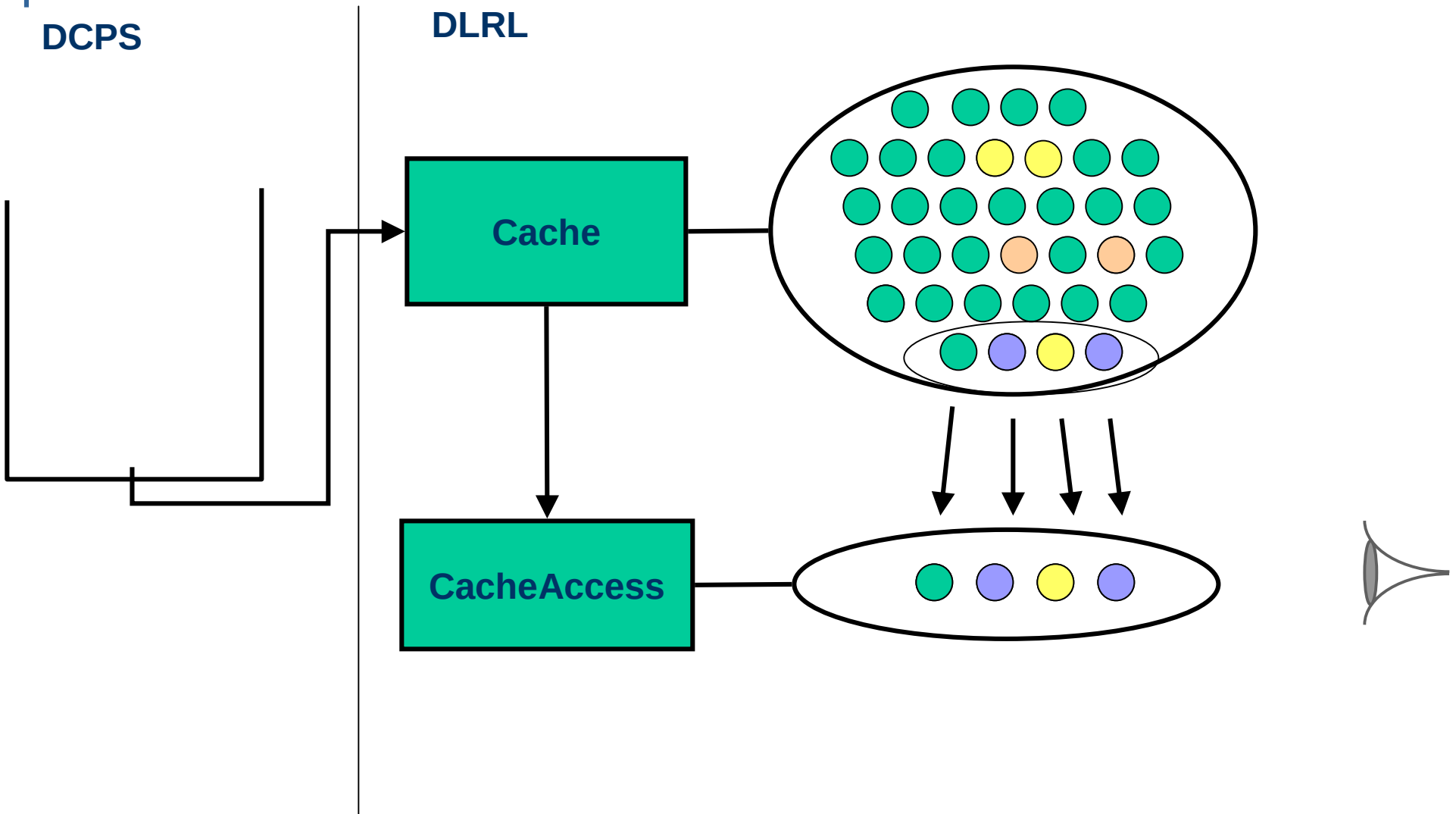
- Create a Cache (using the CacheFactory)
  - Passing a DCPS DomainParticipant and a CacheUsage as parameters
  - Creates under the scenes the DCPS Publisher and/or DCPS Subscriber
- Register the ObjectHomes to that Cache
- Call Cache::register_all_for_pubsub
  - Creates under the scenes the Topics, DataWriters and/or DataReaders
- *If needed, adjust QoS using DCPS*
  - Means are provided to retrieve the DCPS constructs
- Call Cache::enable_all_for_pubsub
  - Ready to read / write objects

CacheAccess in Read mode



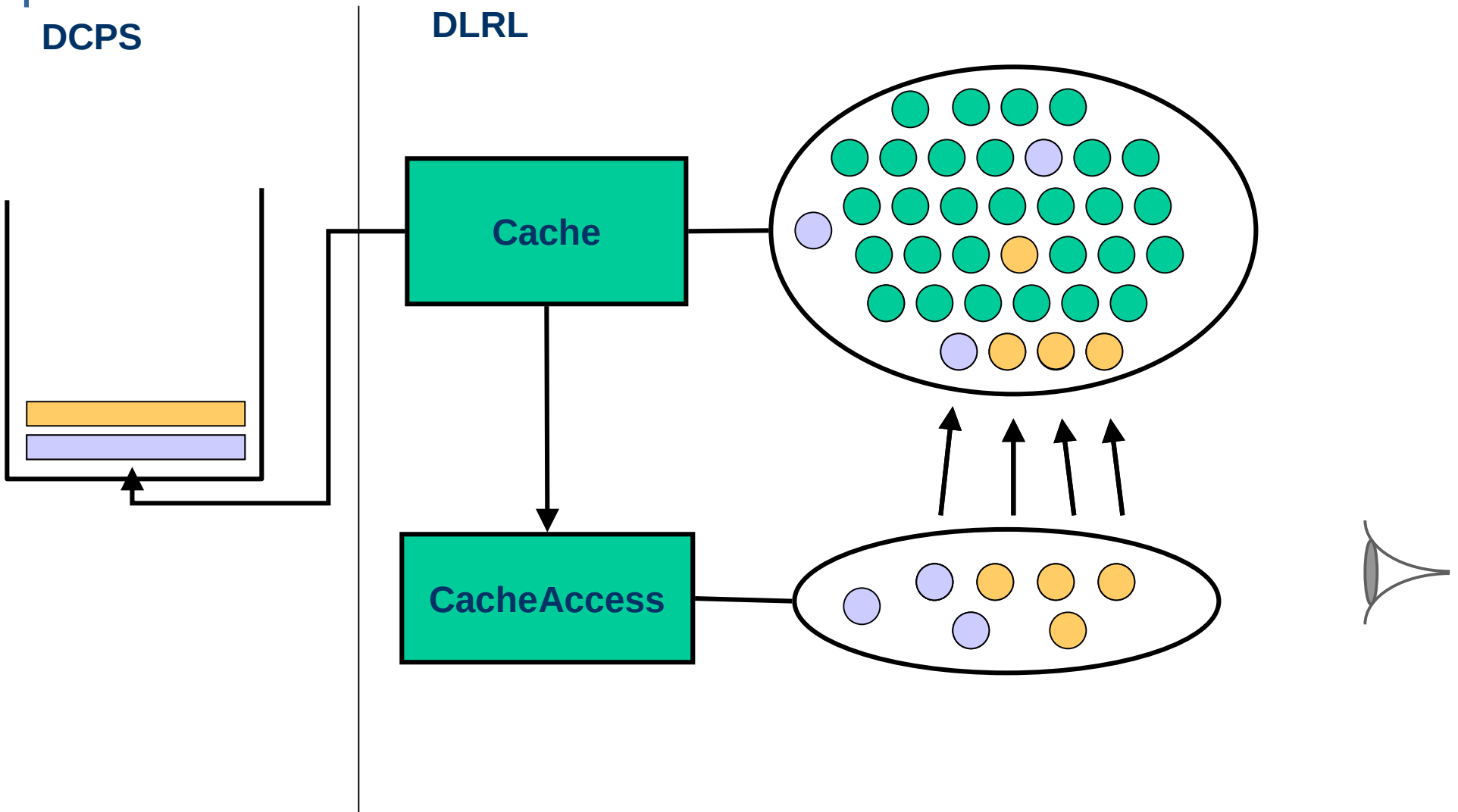
THALES

CacheAccess in Read mode



THALES

CacheAccess in Write mode



THALES

ObjectExtent

- Management objects that gather a set of objects of a given type
  - used to handle as a whole all instances of a given class or all members of a selection
- Can be filtered out (by means of an *ObjectFilter*) and/or modified (by means of an *ObjectModifier*) 'as a whole'

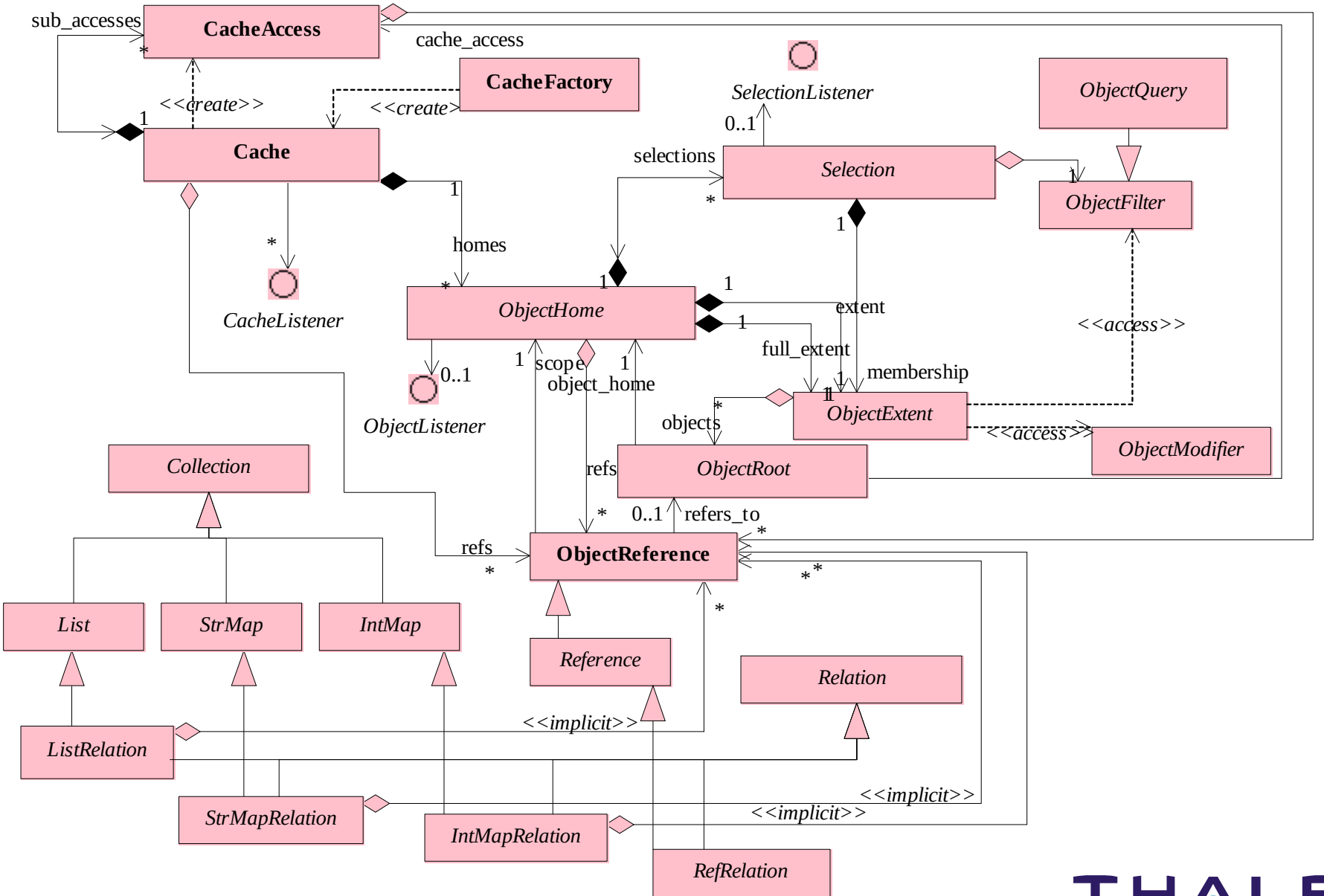
Selection

- 'Active' sub-set of the instances of a given class
  - Resulting of the application of an ObjectFilter
    - ObjectQuery as specialisation of an ObjectFilter
    - Manually and/or automatically refreshed
- Listeners can be attached
  - when an object enters / exits the selection
  - when a member of the selection is modified

Listeners - to notify incoming updates

- 3 kinds of listeners
  - CacheListener
    - When a bunch of incomings updates starts / ends
  - ObjectListener
    - When an object is created / modified / deleted
  - SelectionListener
    - When a member enters a selection / is modified / exits the selection
- Order of triggering
  - CacheListener::on_begin_updates
  - modifications are performed in the Cache objects
  - Object and selection listeners
  - CacheListener::on_end_updates

DLRL Entities (UML model)



THALES

DDS-DLRL is a layer on top of DCPS to ease the management of the data and to integrate it seamlessly in the application

- Object-orientation
- Management of graphs of objects

It promotes typed interfaces (as DCPS)

- By means of code-generation

It does not force a global object model

- The object model is local
  - better flexibility at system level
- Mapping to the DCPS data model