

# Towards Implementing Consistency in OMG MOF-Based Repository Systems

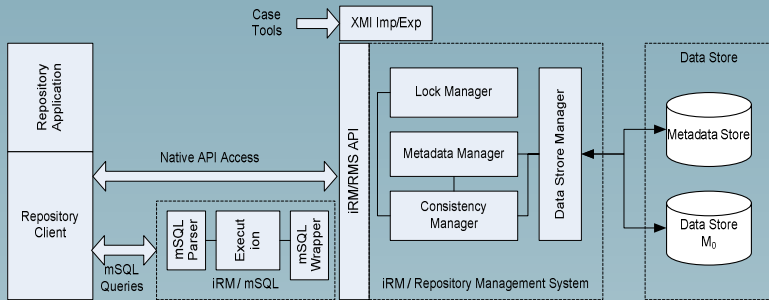
**Iliia Petrov, Marc Holze, Stefan Jablonski**

*ilia.petrov@informatik.uni-erlangen.de*

<http://www6.informatik.uni-erlangen.de/research/projects/irm/>

## Repository Systems

- Database Systems and metadata
- Repositories - Data stores with custom defined system catalogue
  - Management of data and metadata
    - Layered metadata architecture
    - $M_0, M_1, M_2, M_3$
  - Custom-defined system catalogue
  - Consistency among objects across metadata layers
    - Main contribution
    - Repository Transactions
- Declarative Query Language
- iRM – Prototype repository system



## Why is constancy an issue?

- Application1 deletes an attribute of a super-class of class C in an  $M_2$  model
- Application2 instantiates one of the sub-classes of C providing value for the deleted attribute
- Application3 counts the attributes of all  $M_1$  classes

## What do we need?

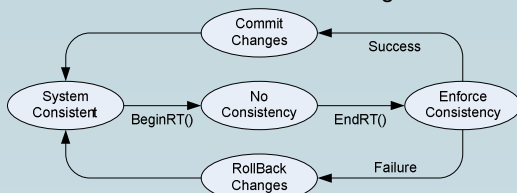
- Operational consistency
  - Concurrent multi-client access
  - Atomic sequences of repository operations
- Metadata Integrity
  - Well-formedness
  - Structural Integrity

## Solution: Enforce consistency within repository a transaction!

- Atomicity / Isolation / Consistency
- Durability – rely on the transactional capabilities of the underlying data store

## Atomicity

- Cooperative Atomicity – One logical operation comprises multiple RMS API operations
  - Example: Create a sub-class of a class in a package
- A demarcation mechanism needed → BeginRT / EndRT



## Isolation

- Locking
  - Extension of the multi-granularity locks
  - Short S and long X locks, 2PL compatible
  - Instance lattice
- A lock on an object on  $M_n$  can be placed if:
  - No dependent object on any underlying  $M_{n-p}$
  - Is locked with an incompatible lock

		Requested		
		X	S	
Present	$M_2$	X	-	-
	$M_1$	S	-	+
	$M_0$	X	-	-

## Consistency

- Well-Formedness → Syntax conformity, Immediate constraints
- Structural consistency → Do objects conform to their type definition
  - Creation: do newly created objects conform to their meta-objects
  - Modification:
    - Is a set of modification operations allowable
    - Can all instance objects be adapted to the modified meta-obj.
- Structural constraints
  - Hierarchies: containment hierarchy, generalization hierarchy
  - Associations, association ends and multiplicities
  - Attribute data types and optionality
  - Valid meta-object

Manipulation ( $M_2$ )	Model Element	Required Action ( $M_1$ )
Deletion	Package	Delete the package proxy (including all contents)
Deletion	Class	Delete the class proxy (including all instances)
Deletion	Association	Delete the association proxy (including all links)
Deletion	Attribute (classifier-level)	Remove the attribute from the class proxy
Deletion	Attribute (instance-level)	Remove the attribute from all instances
Change	Containment Hierarchy	Roll back the repository transaction
Change	Generalization Hierarchy	Roll back the repository transaction
Change	Attribute Type	Roll back the repository transaction if values are incompatible
Change	Association End Type	Roll back the repository transaction, if the new type is not a super-type of the old one
Creation	Attribute (classifier-level)	Add a new attribute to the class proxy
Creation	Attribute (instance-level)	Add a new attribute to all class instances (initialized with default values)

## Summary and Conclusions

- We defined the concept of repository consistency
- We proposed
  - Novel Algorithm
  - Novel concepts (consistency and locking)
  - Implementation