

Developing a RESTful Mixed Reality Web Service Platform

WS-REST'2010 @ WWW'2010, Raleigh, NC

April 26th 2010

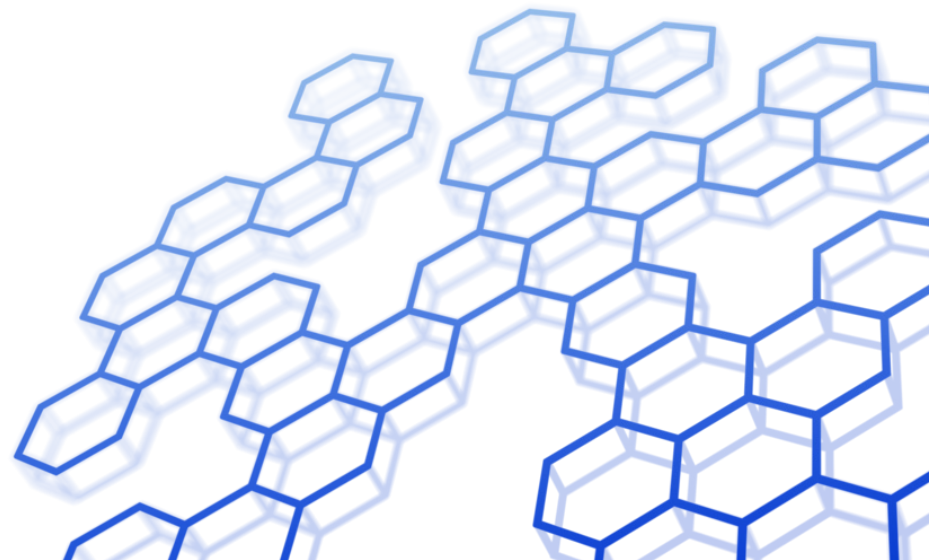
NOKIA

Petri Selonen

Principal Researcher (PhD)

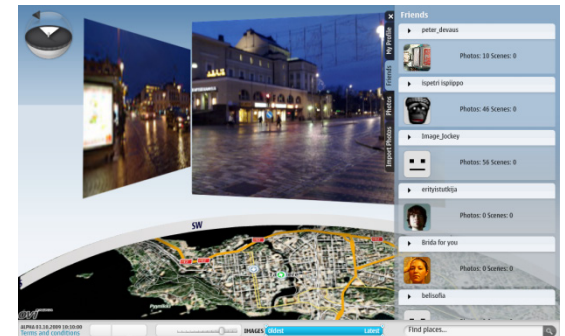
Nokia Research Center, Tampere-FI

petri.selonen@nokia.com



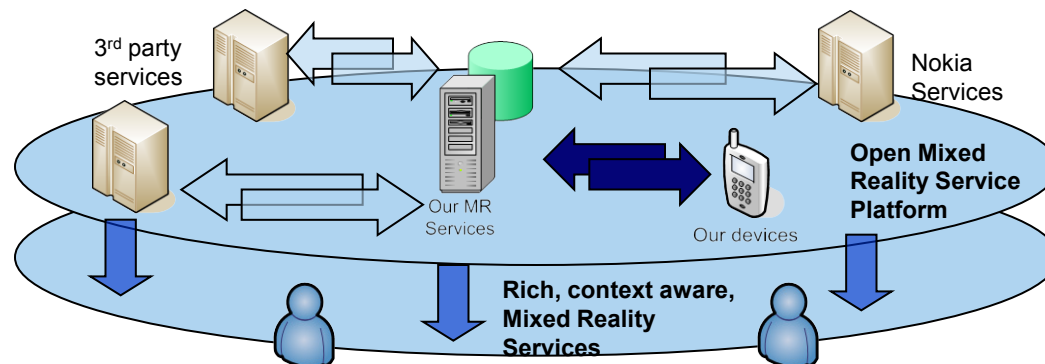
Background: Mixed Reality Solutions Web service

- **Mixed Reality** refers to the **fusion of real and virtual worlds** for creating environments where **physical and digital objects co-exist**
 - encompasses Augmented Reality and Augmented Virtuality
- The **MRS-WS** is a Web Service platform being developed at NRC serving Mixed Reality content for MR applications and solutions
 - 3D models, pointclouds, buildings, street-view panoramas, points of interest, annotations, photos, comments, tags, etc.
- ReST and Resource Oriented Architecture was chosen because the MRS-WS essentially serves geo-spatially arranged content



Initial Requirements

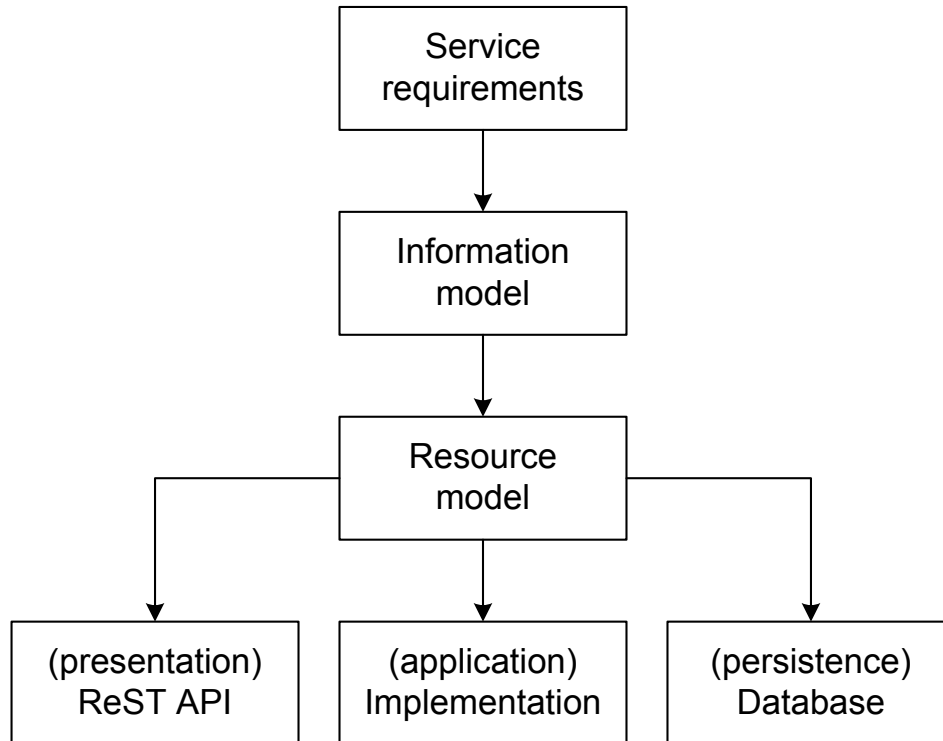
- Moving from silos and stand-alone backends to an **open service platform**
 - **common backend** for easily building **MR services** and applications
 - support for both **desktop and mobile clients**
 - **secure and scalable**
 - manage **social connections, context, content, maps, panoramas** etc.
 - provide **unified access to content** from external service providers for the user
- Allow **rich mash-ups** and support **3rd party developers** and **open innovation**
- Use standard and de-facto **Internet technologies**



People Challenges

- Development in a **program mode**
 - dedicated first-priority clients, demo driven operation, sprint iterations
 - multi-disciplinary, multi-continent research and development
 - ensuring building a platform while serving solution specific clients
- Developers seem to find Resource Oriented systems **hard to comprehend**
 - while simple in principle, **simple != easy**
- Lack of a commonly agreed, systematic and well-defined **approach for proceeding from service requirements to a RESTful Web Service**
 - how do you design RESTful and Resource Oriented Web services?
 - how do you facilitate communication with different stakeholders?

Proposal for a Solution: ReSTifying Approach



First iteration 2008: exploratory work on designing ReSTful Web Services for arbitrary functionality through *behavioral canonicalization*

Laitkorpi, M., Selonen, P. and Systä, T. Towards a Model-Driven Process for Designing ReSTful Web Services, [ICWS](#), 2009.

Second iteration 2010: developing ReSTful Web Services for *a priori* content-oriented systems

Selonen, P., Belimpasakis, P. and You, Yu: "Developing a ReSTful Mixed Reality Web Service Platform", [WS-REST](#), 2010

Selonen, P., Belimpasakis, P. and You, Yu: "Experiences In Building a ReSTful Mixed Reality Web Service Platform", [ICWE](#), 2010

Resource Model overview

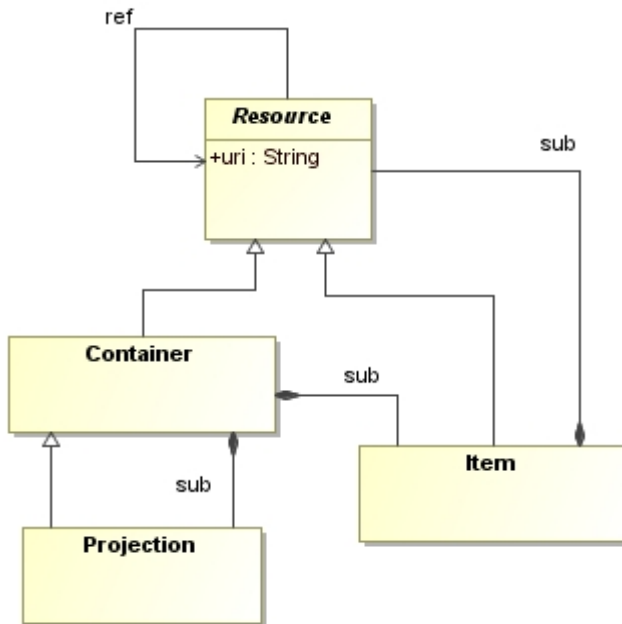
A **Resource Model** organizes the concepts of the domain model to addressable entities that can be mapped to elements of a ReST API, service implementation and database schema.

Items represent individual resources having a state that can be retrieved, created, modified and deleted

Containers can be used for retrieving collections of items and creating new ones

Projections are filtered views to containers

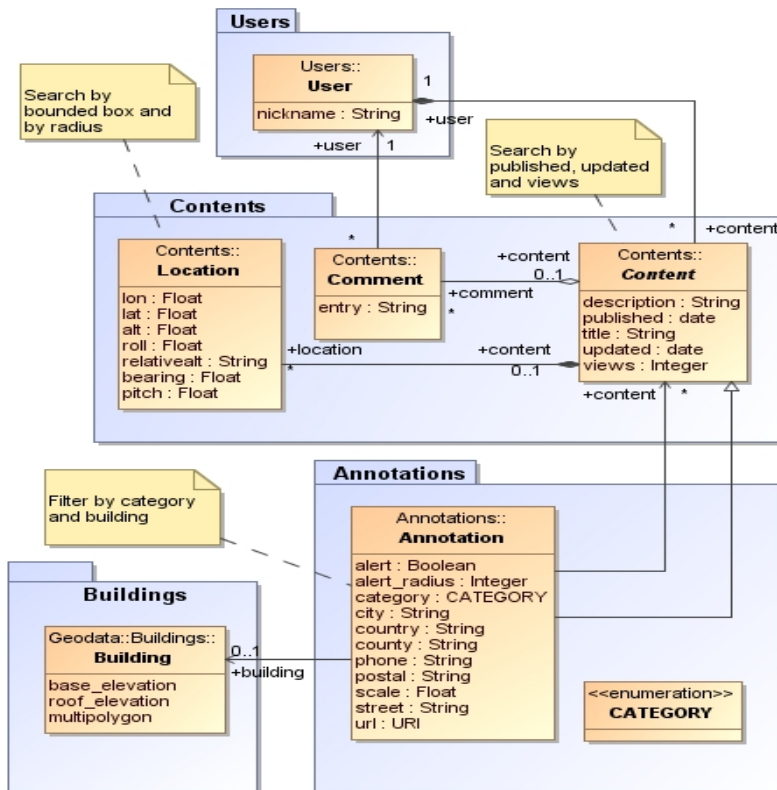
Resources can have subresources and links to other resources.



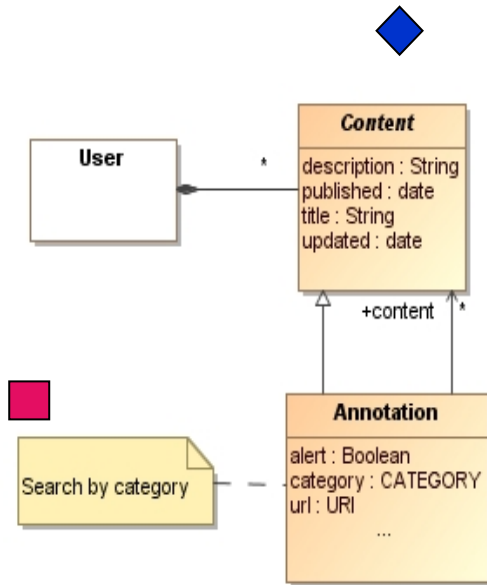
Example: Annotations

An Annotation

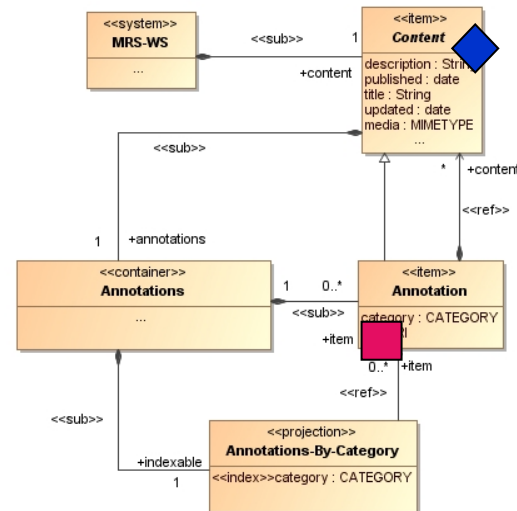
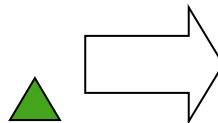
- has support for a location and spatial arrangement in 3D space, for user generated comments and for an owning user;
- has properties title, a description, creation and modification timestamps, street address and view count;
- can be searched by location in a bounded box;
- can be searched by location within a given radius;
- can be searched based on links to a particular building;
- can be searched based on selected categories only;
- can be searched based on most viewed elements;
- can be set to have been viewed by the client;
- can be attached content (eg. multimedia, binary, pointcloud) ; and
- can be arranged spatially based on a normal vector.



Information Model to Resource Model

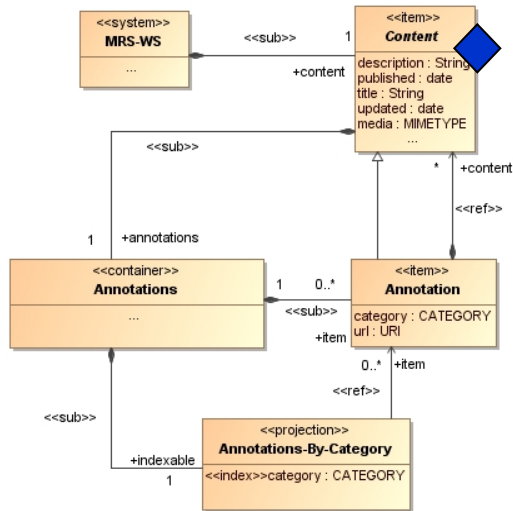


Information Model



Resource Model

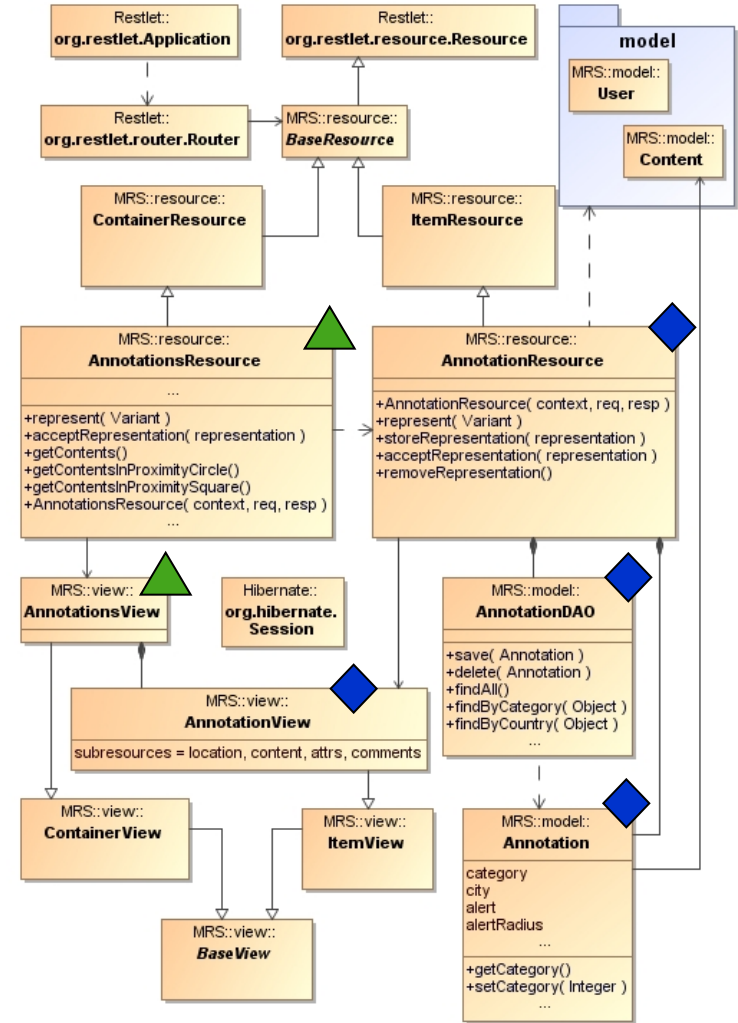
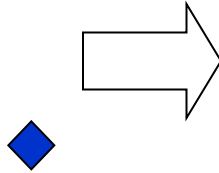
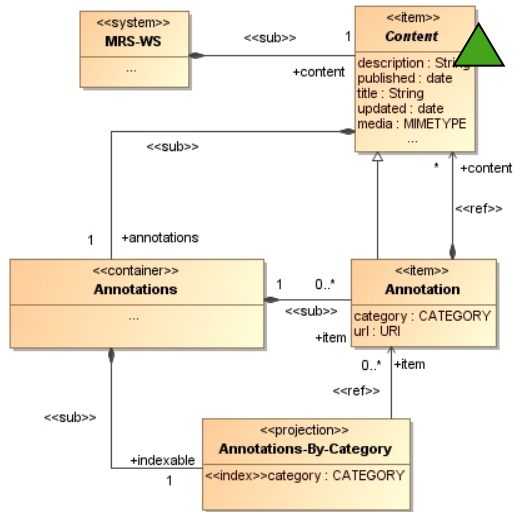
RM to ReST API (presentation)



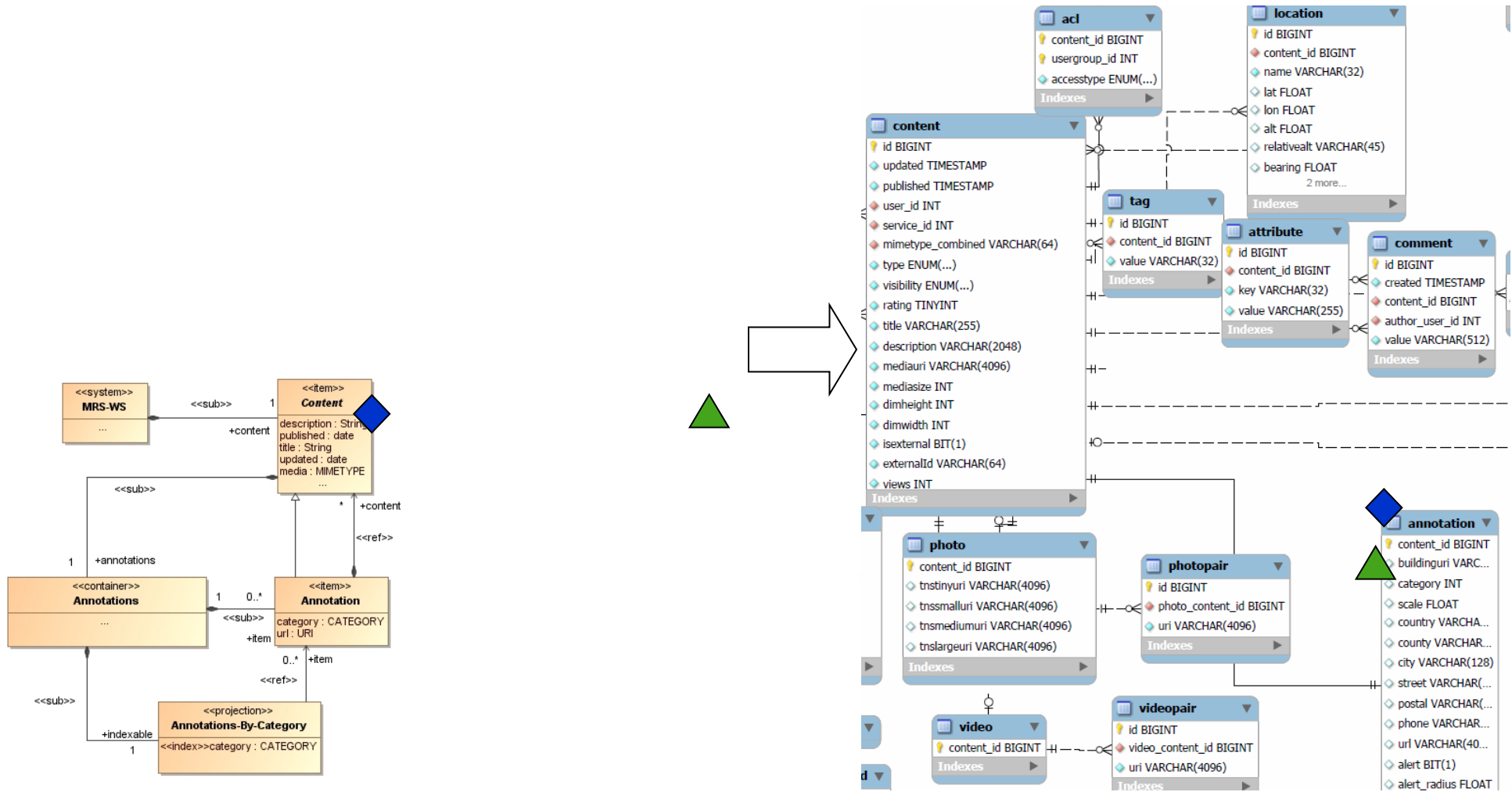
/content/annotations ◆
 /content/annotations/{annotation.id} ▲
 /content/annotations?
 category={annotation.category} ■

application/vnd.research.nokia.annotation ▲
 <annotation href="...">
 <id>4195042682</id>
 <updated>2009-12-18 04:01:13.0</updated>
 <published>2009-12-18 04:01:02.0</published>
 <title>For rent!</title>
 <description>I'm renting a 60m2 apartment here.
 Call me if interested.</description>
 <media />
 <category>1</category>
 <url>1.0</url>
 <contents>
 <content href="..." />
 </contents>
 </annotation>

RM to Implementation (application)



RM to Database (persistence)



MRS-WS Implementation Binding Summary

	API (Restlet)	Representation (XML/JSON)	Model (Hibernate, Java EE)	Persistence (MySQL)
Item	Restlet resource bound to the URI. Supported default operations are GET, PUT and DELETE.	Representation parsing/generation based on the item attributes. Subresources inlined per request basis.	A native Java object (POJO) generated for each item with a Hibernate Data Access Object and binding to database elements.	Items are rows in respective database table with columns specified by item attributes. References map to foreign keys.
Container	Restlet resource bound to the URI. Supported default operations are GET and POST.	Representation parsing/generation delegated to Items.	Basic retrievals to database, using item mappings.	Containers are database tables.
Projection	Implemented on top of respective Containers.	Representation generation delegated to Container.	Extended retrievals to database, using item mappings.	Stored procedures for more advanced database queries. Tables implied by Container.

Concluding Remarks

- The paper outlined a lightweight approach for developing RESTful and Resource Oriented Web services for content oriented systems
 - a common modeling notation for designing and communicating
 - mapping domain model concepts to ROWS implementation
 - guide the platform developers
- The approach emerges from the requirements of a real-world Web service platform development project
- Some identified benefits from REST, ROA and the approach
 - building a platform while supporting dedicated clients in rapid pace
 - targeting an architecture supporting extensibility
 - decouples clients from the service
 - Qt/C++, Java, Python, JavaScript/AJAX
 - Windows, Linux, Symbian, Maemo

Concluding Remarks, cont'd

- There are lots of topics outside the scope of the paper
 - **evaluating** the resulting **REST API**: URI hierarchies, representations, content types...
 - **idioms and patterns** for designing RESTful Web services
 - issues related especially to REST and **mobile clients**
 - roundtrips and latency
 - client processing overhead
 - capability of client side frameworks
 - balancing between server and client
 - “how do experts design RESTful Web services?”
- **What is the elevator pitch for conforming to REST and ROA?**
 - why should 1) managers, 2) service developers and 3) clients care?

Thank You

NOKIA



Observations

- In a perfect world, you could
 - evaluate the benefits and costs of adhering to the REST architectural style
 - have practices in designing and developing RESTful Web services
- But, most client developers don't give a toss about REST
 - just give the API to program against and give us the data...
 - ...and don't change anything in the future
- Most service developers don't give a toss about being RESTful
 - just copy and paste the code if it gets the job done
 - people think in terms of hiding data and exposing APIs
 - if simple operation is hard to RESTify, why bother?
- Most managers don't give a toss about you (nor REST)
 - interoperability-schmoperability – show me the money
- It's a cold world out there

