WP4 – Knowledge base Design and Implementation

Prof. Dr. Geylani Kardas
(KocSistem)

WP4 Knowledge Base Design and Implementation

- Design and implement the ModelWriter's:
  - federated Knowledge Base itself, hosting multiple formalisms.
  - bi-directional text-model synchronization mechanism.
  - required API
  - a set of specialised modules (plug-ins) that exploit the Knowledge Base in ways that make the tasks of Technical Authors much more productive, e.g. consistency checks.
  - the collaborative functions linking and hierarchically organizing multiple ModelWriter KBs used by different Technical Authors on different sites.

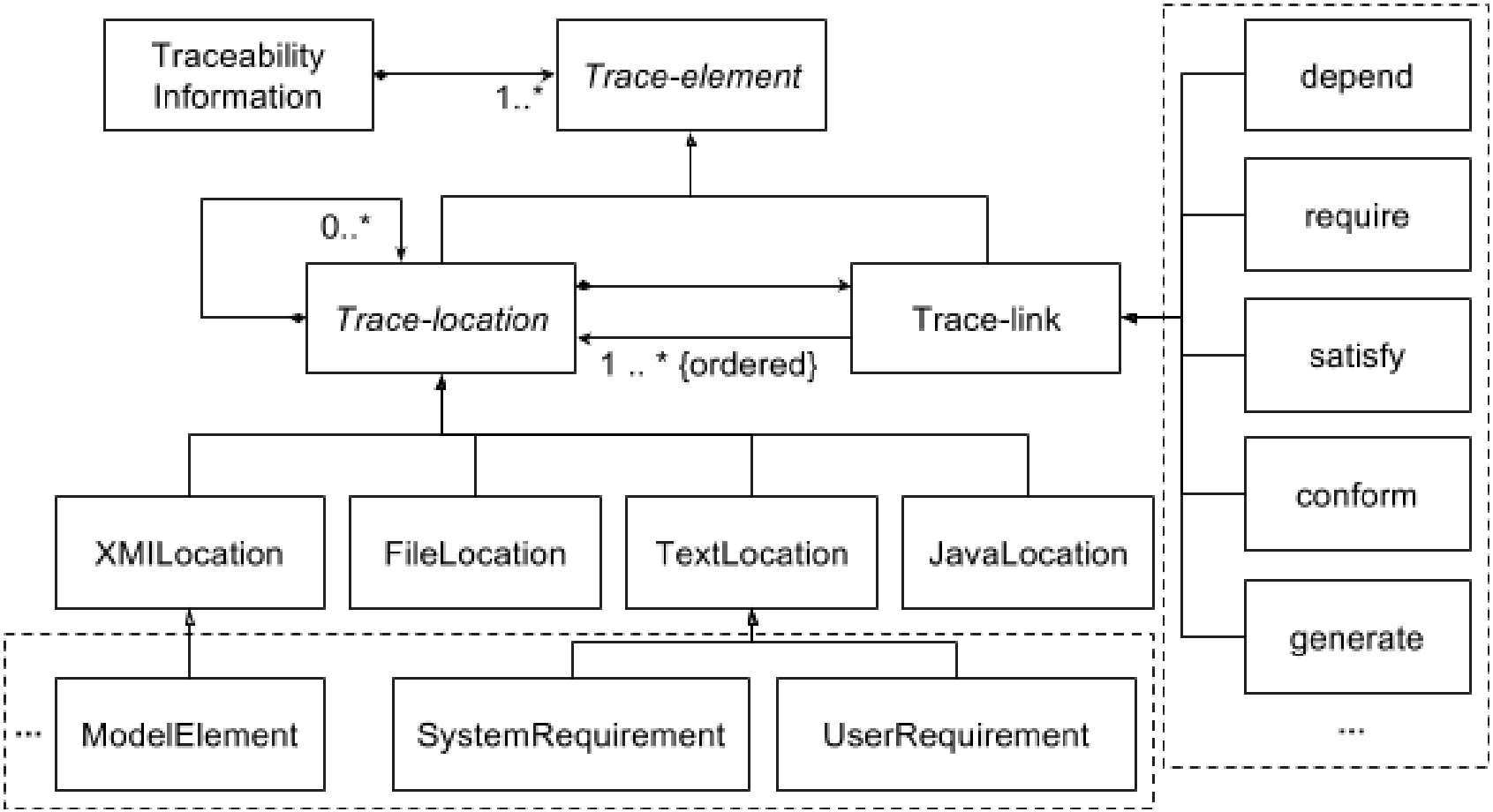
WP4 Knowledge Base Design and Implementation

- Plug-in #1 – This provides consistency and completeness checks within the same software lifecycle document, allowing automatic quality review of the content (meaning).

- Plug-in #2 – This provides consistency and completeness checks between related set of documents.

- Plug-in #3 – This provides semantic comparison between two versions of the same software lifecycle document (i.e. what conceptual changes have happened).

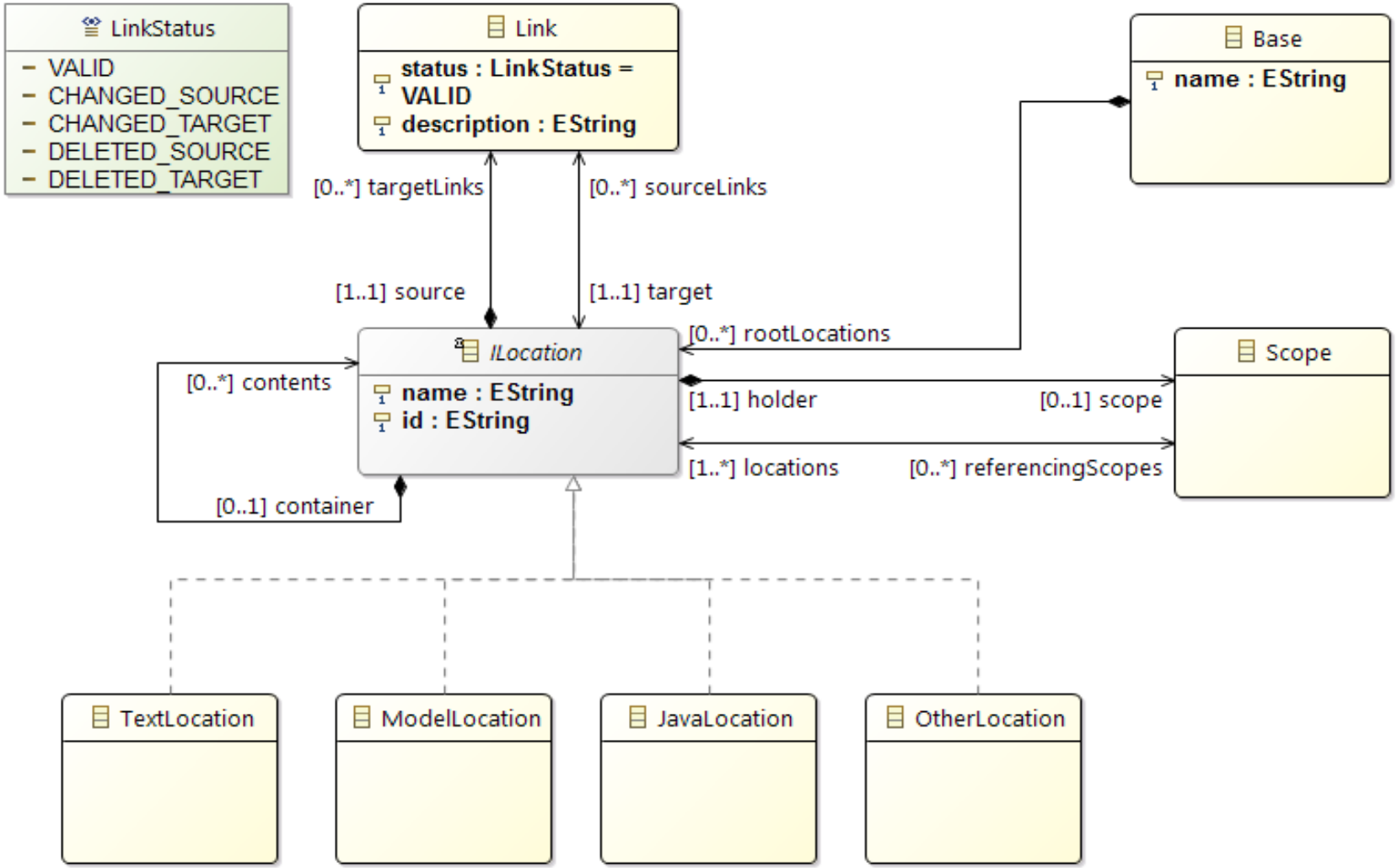
ModelWriter Core Model Approach



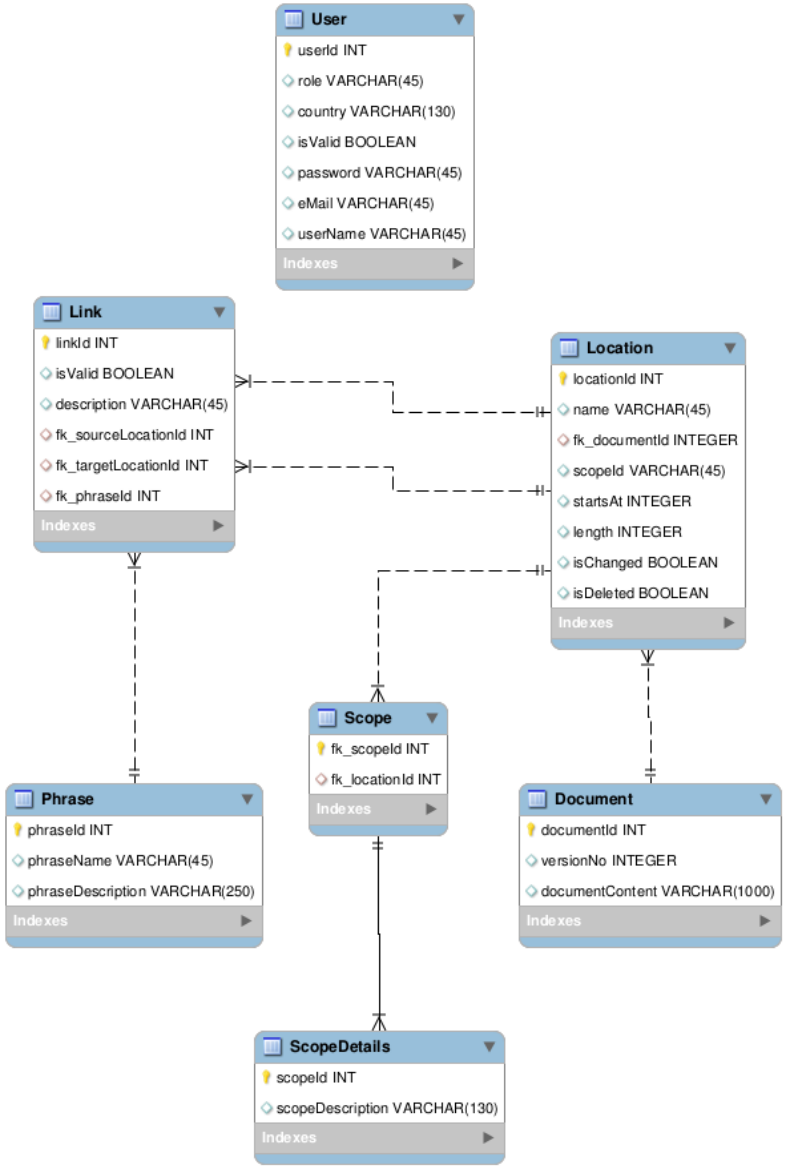
ModelWriter Core Model
State-of-the-art

	SotA Tools & Approaches	Consideration of Different Artefacts/ Heterogeneity of artefacts (internal or external models)	Traceability Management		
			Approach (Management of Traceability)	Definition of Formal Semantic for trace-links	Dynamic Configuration of Semantics of trace-links
Industrial Tools, Methods & Standards	SysML[1]	UML Elements	UML Profiling mechanism	-	-
	ReqIF[2]	Textual Requirements	Definition of XML Schema and extending its Data-Model	-	-
	IBM Doors[3]	Arbitrary between model elements	Creation of Relation Types	Transitivity of relations	-
	ModelWriter	Arbitrary between any model element or textual requirements	Basic Traceability Model extended with a Formal Specification	FORL (First-order Relational Logic)	+

ModelWriter Core Model Implemented by OBEO



Knowledge-base Design

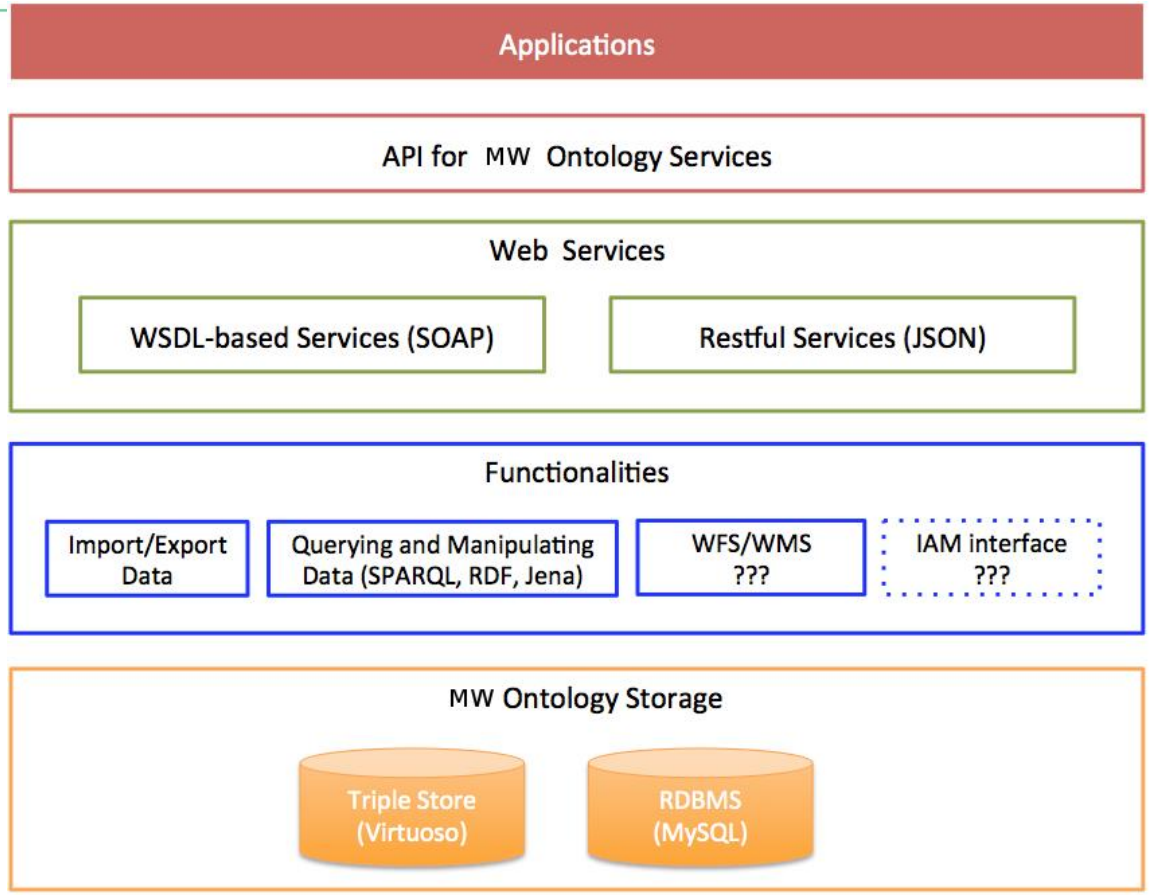


The class diagram of the knowledge base data structure representation.

MANTIS has prepared D4.1.1 Knowledge Base Design Document.

This document still is open to edit and to add contributions.

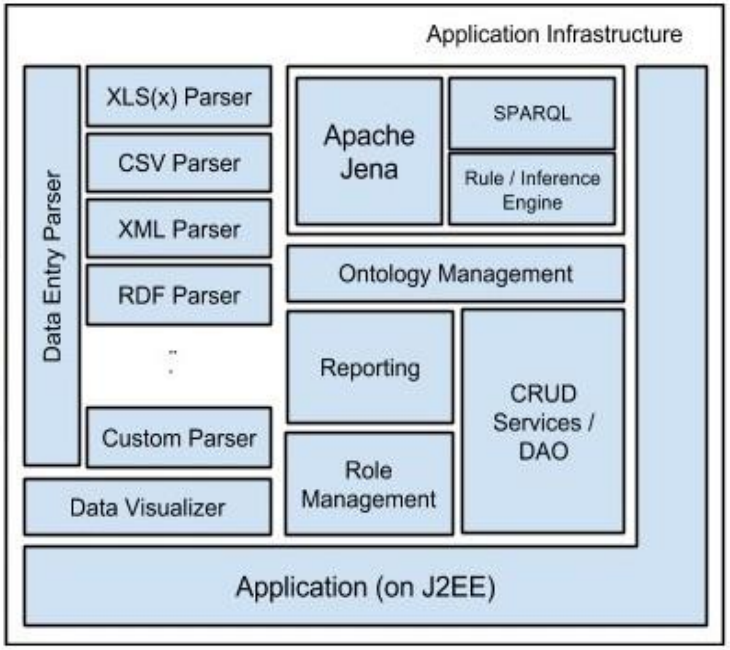
Knowledge-base Implementation
Ontology Infrastructure

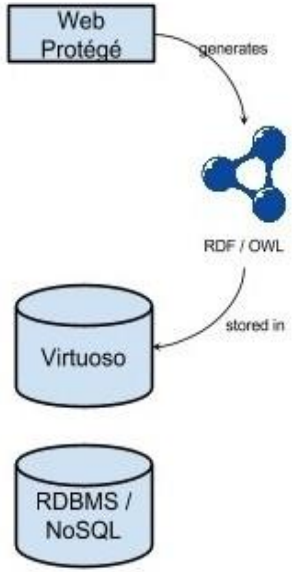


WMS/WFS services: standardized by Open Geospatial Consortium to use geographical attributes. **IAM interface** for identification and access management. Current platform uses OAuth2 authentication system (http://oauth.net/2/)

Web Feature Services (WFS)/Web Map Services (WMS) may be partially implemented. Identity and access management (IAM) services may not be implemented at all.

Knowledge-base Implementation
Ontology Infrastructure





The requests that arrives via web services are stored as triple in the Virtuoso Ontology Database and can be queried.
All processes that are based on ontology are done in Ontology Management Module (OMM). OMM is set up on parent Jena framework and inference engine can be utilized efficiently with SPARQL queries.
Also by using Web Protégé ontology can be generated in RDF/OWL formats and that ontology can be stored in Virtuoso.

Ontology Issues and Services

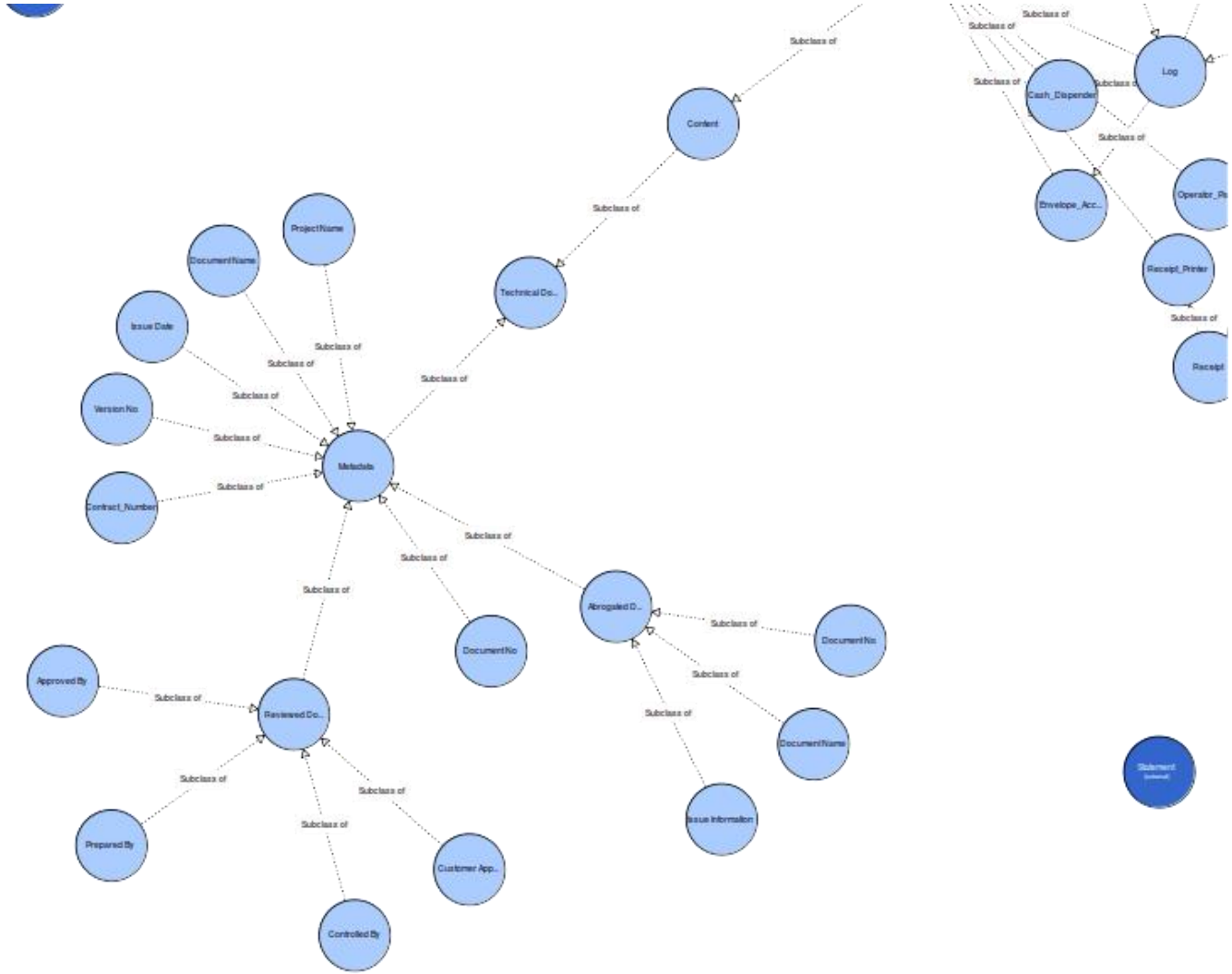
Ontology Issues

CRUD operations on ontologies as RESTFUL services
Using a sample design document, Mantis designed a document ontology
Ontology is hosted on Mantis servers
Manual RDF export
Automatically RDF export (working on)

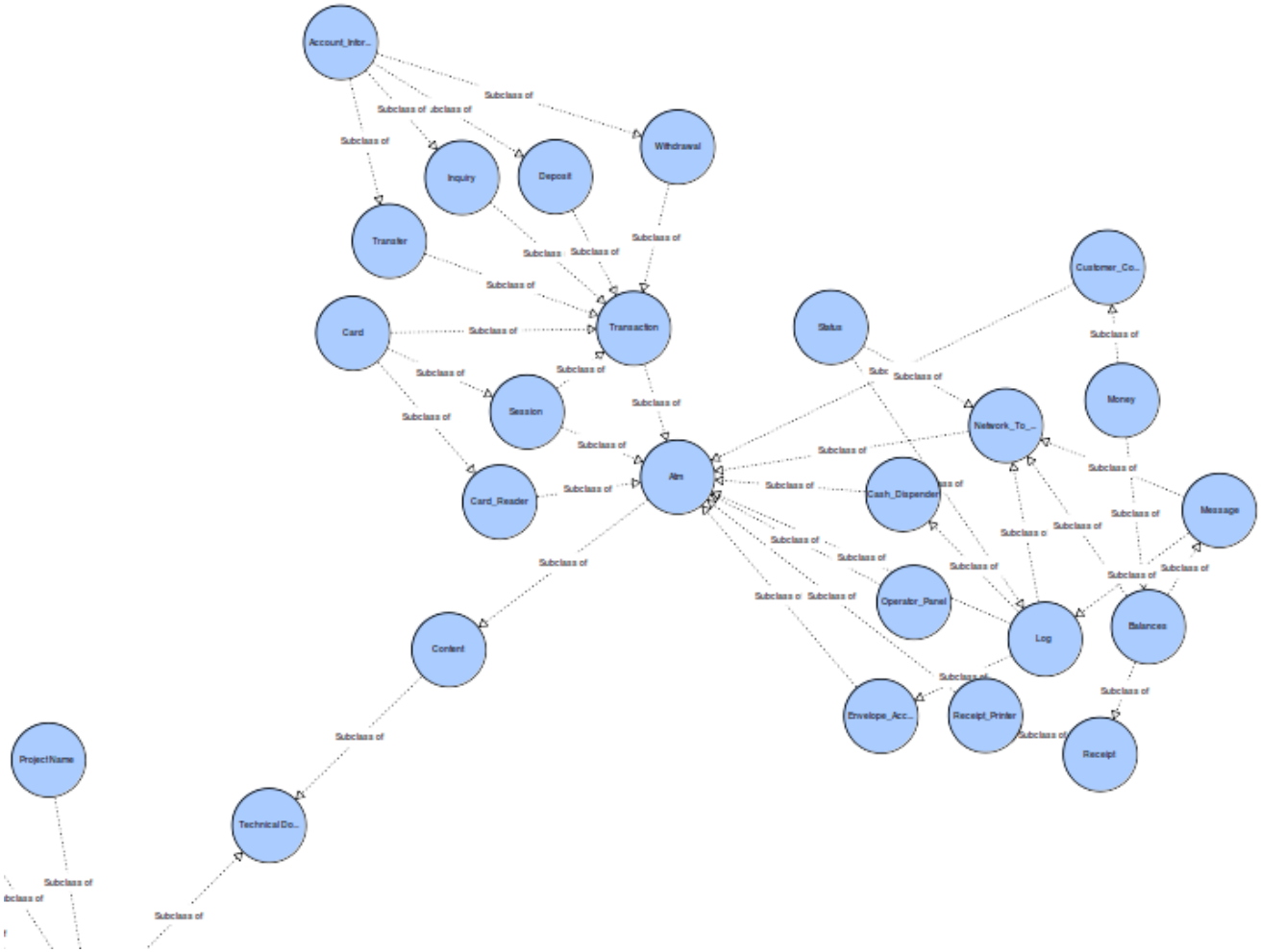
Ontology Services

insertTriple: This method inserts a new triple into an ontology.
ImportIntoOntology: This method imports triples into an ontology
exportOntology: This method exports a specific ontology
executeQuery: This method executes specific query
dropOntology: This method drops a specific ontology
removeTriple: This method removes specific triple(s)

Sample Document Ontology Model



Sample Document Ontology Model Instance



Thank you for your attention
We value your opinion and questions.