WP4 – Knowledge base Design and Implementation

Prof. Dr. Geylani Kardas
(KocSistem)

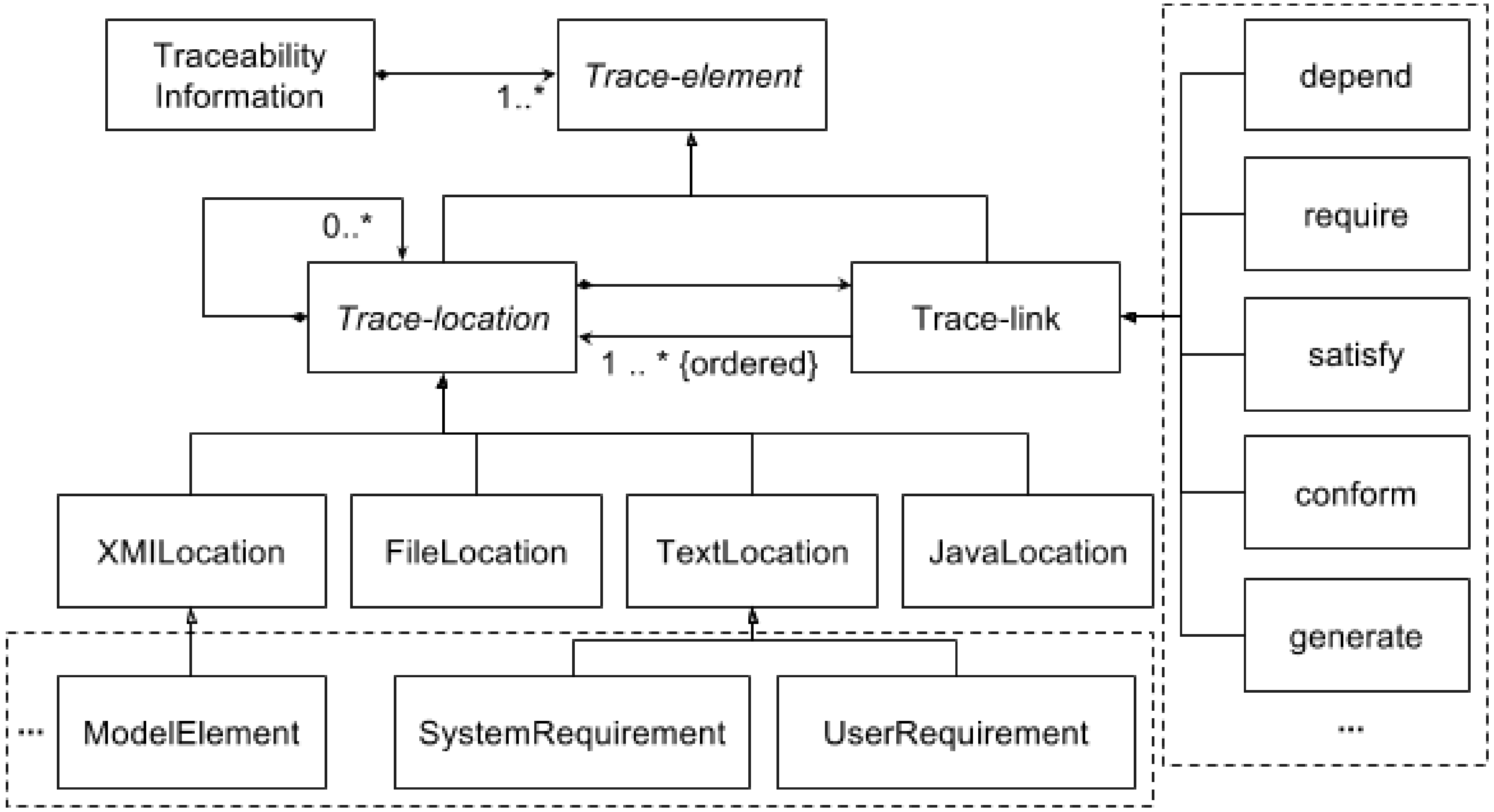
WP4 Knowledge Base Design and Implementation

- Design and implement the ModelWriter’s federated Knowledge Base itself, hosting multiple formalisms.
- Design and implement its bi-directional text-model synchronization mechanism.
- Design and implement its API.
- Design and implement 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.
- Design and implement 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).

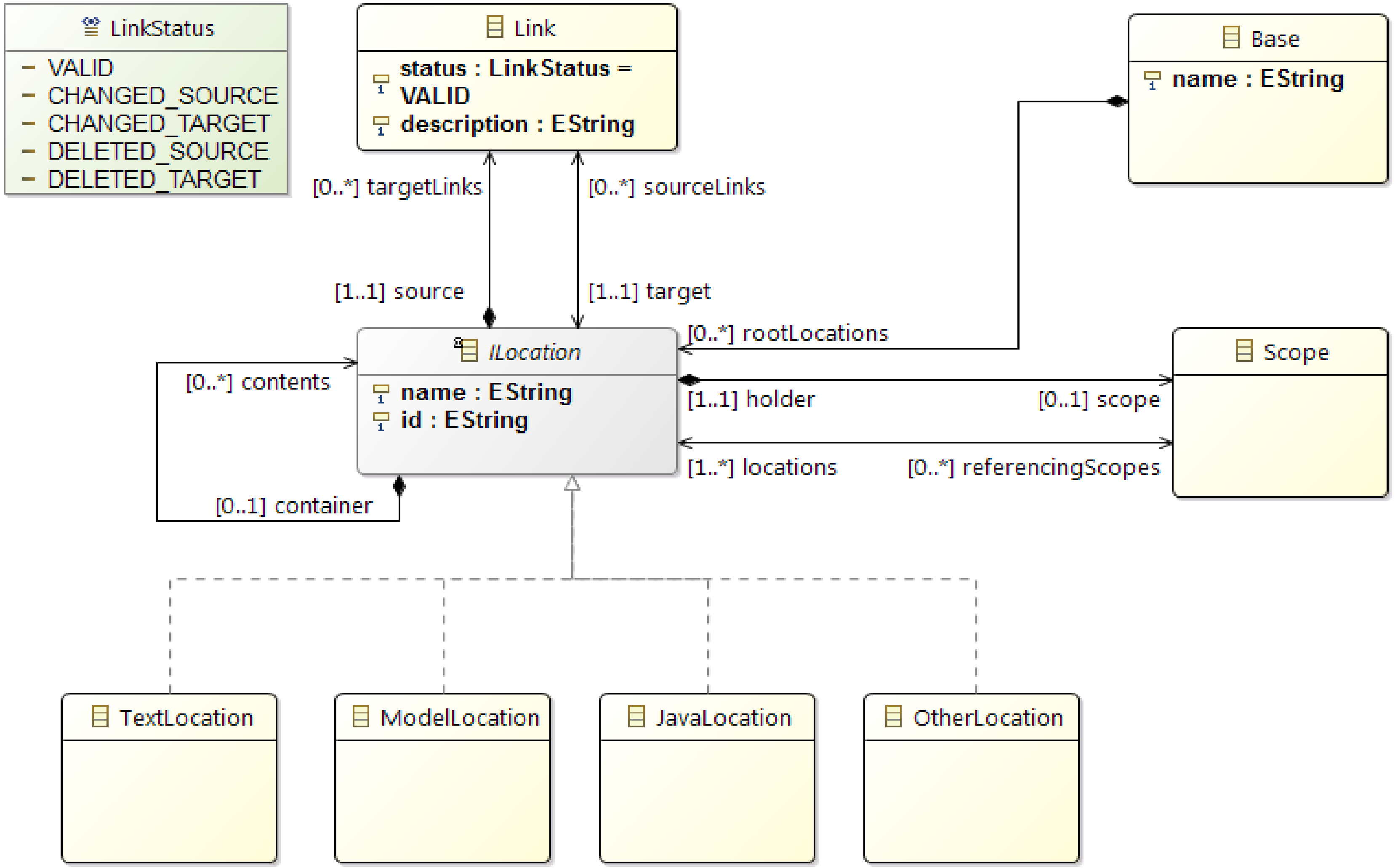
ModelWriter Core Model Approach



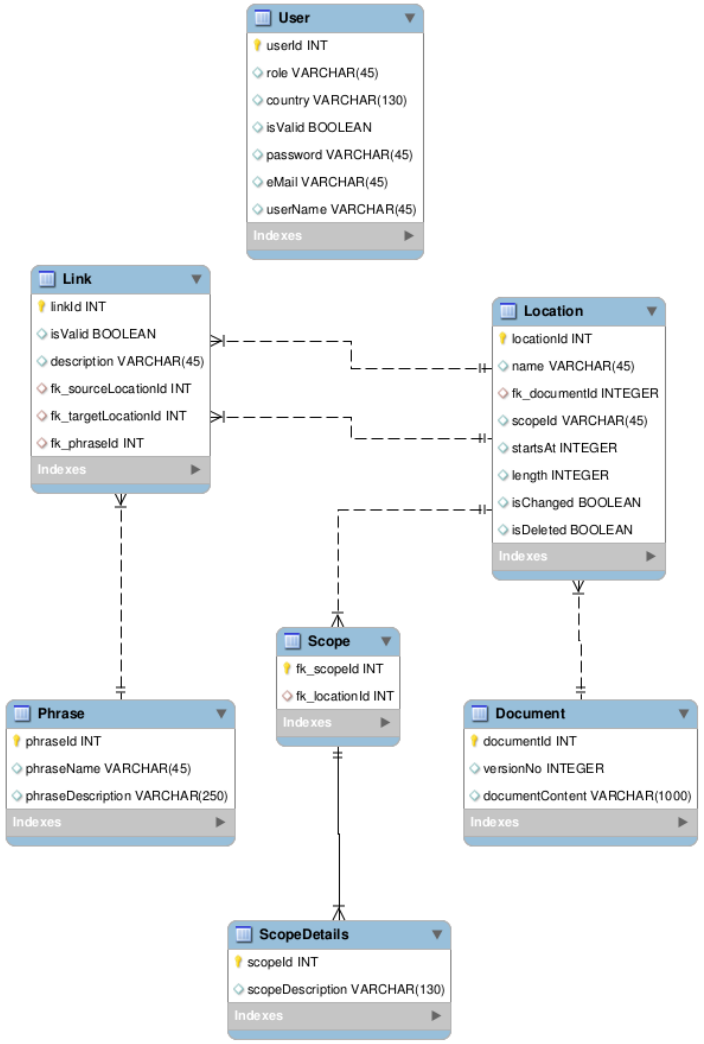
ModelWriter Core Model
State-of-the-art

	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	-
Approaches that provide Analysis Support about Traceability Information	TRIC, Goknil et. al[4]	Model-based Software Requirements and Architectural Models	Extending predefined Traceability metamodel	FOL (First-order Logic)	- (predefined semantics for each trace relation type defined in the metamodel)
	Sebatzadeh et. al.[5]	Model Elements (Homogenous Models)	Formal Specification	RML[6] (Relational Manipulation Language)	-
	Paige et. al. [7]	Model Elements	Case-specific Traceability Metamodel (EMF-based)	Epsilon Validation Language which is an extension of OCL	+
	Drivalos et. al.[8]	Traceability Metamodeling Language (TML)	a metamodelling language dedicated to defining traceability metamodels	-	-
	ModelWriter	Arbitrary between any model element or textual requirements	Basic Traceability Model extended with a Formal Specification	FORL (First-order Relational Logic)	+

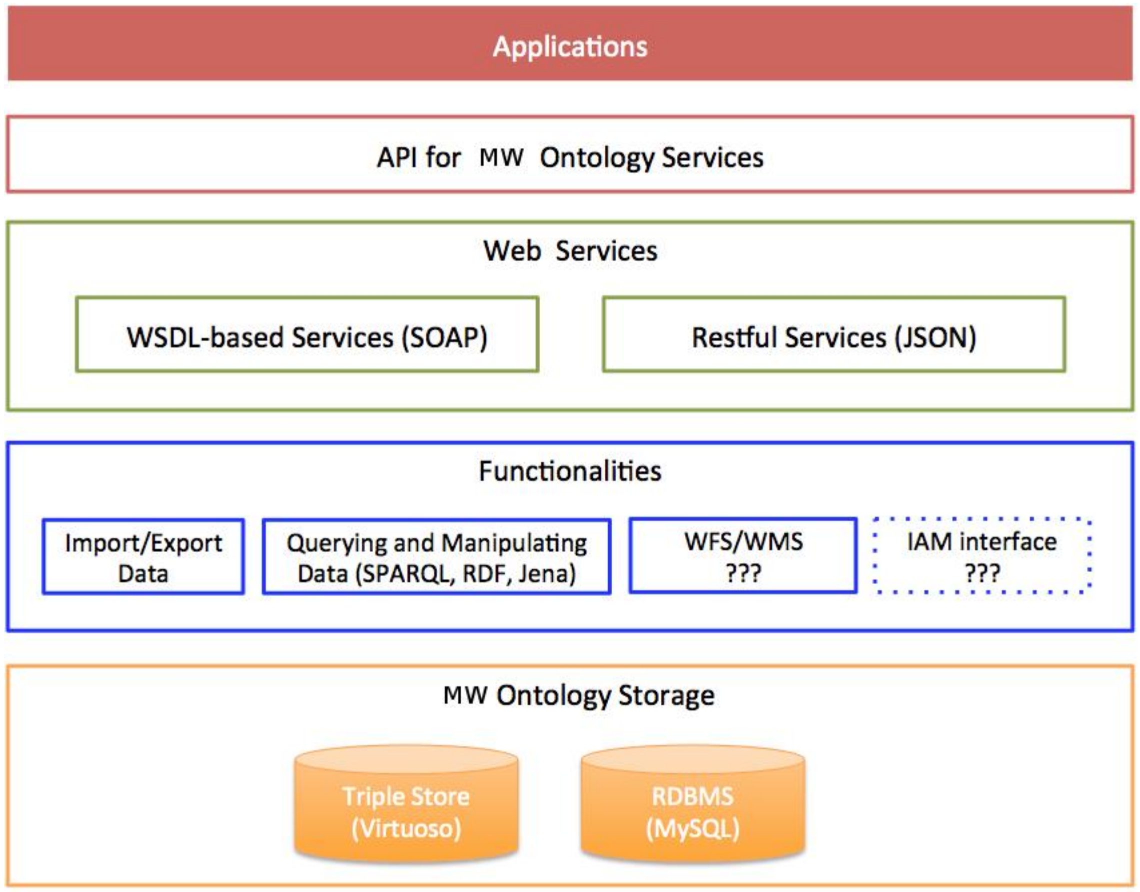
ModelWriter Core Model Implemented by OBEO



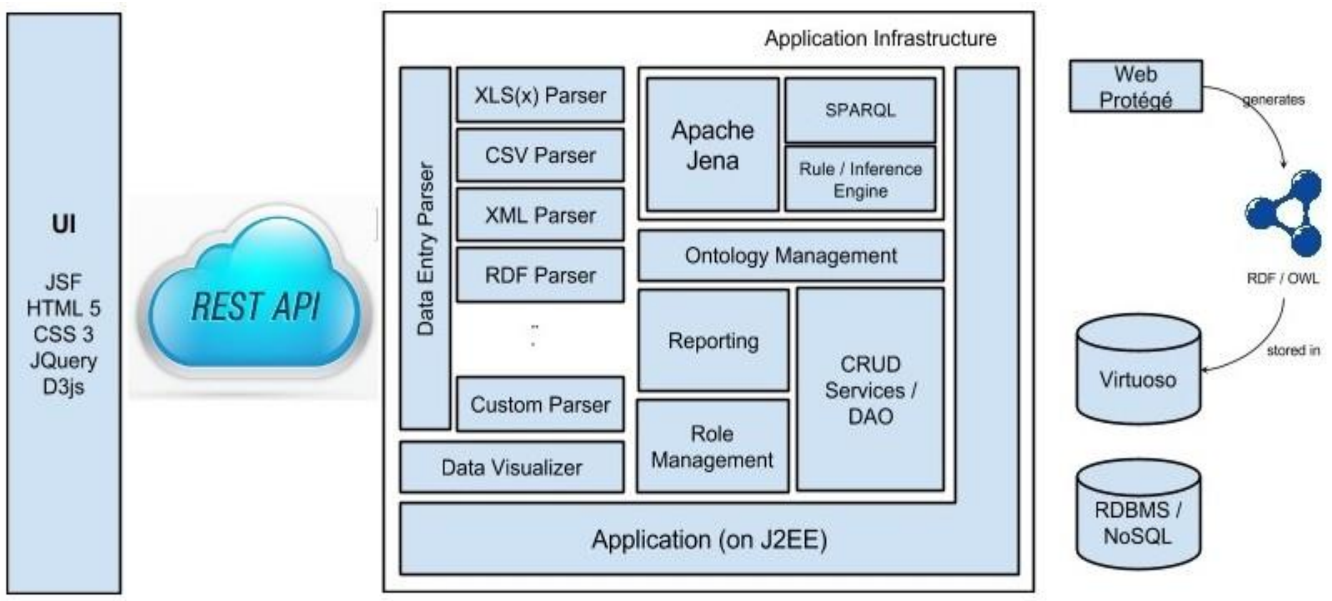
Knowledge-base Design



Knowledge-base Implementation
Ontology Infrastructure



Knowledge-base Implementation
Ontology Infrastructure



Ontology Issues and Services

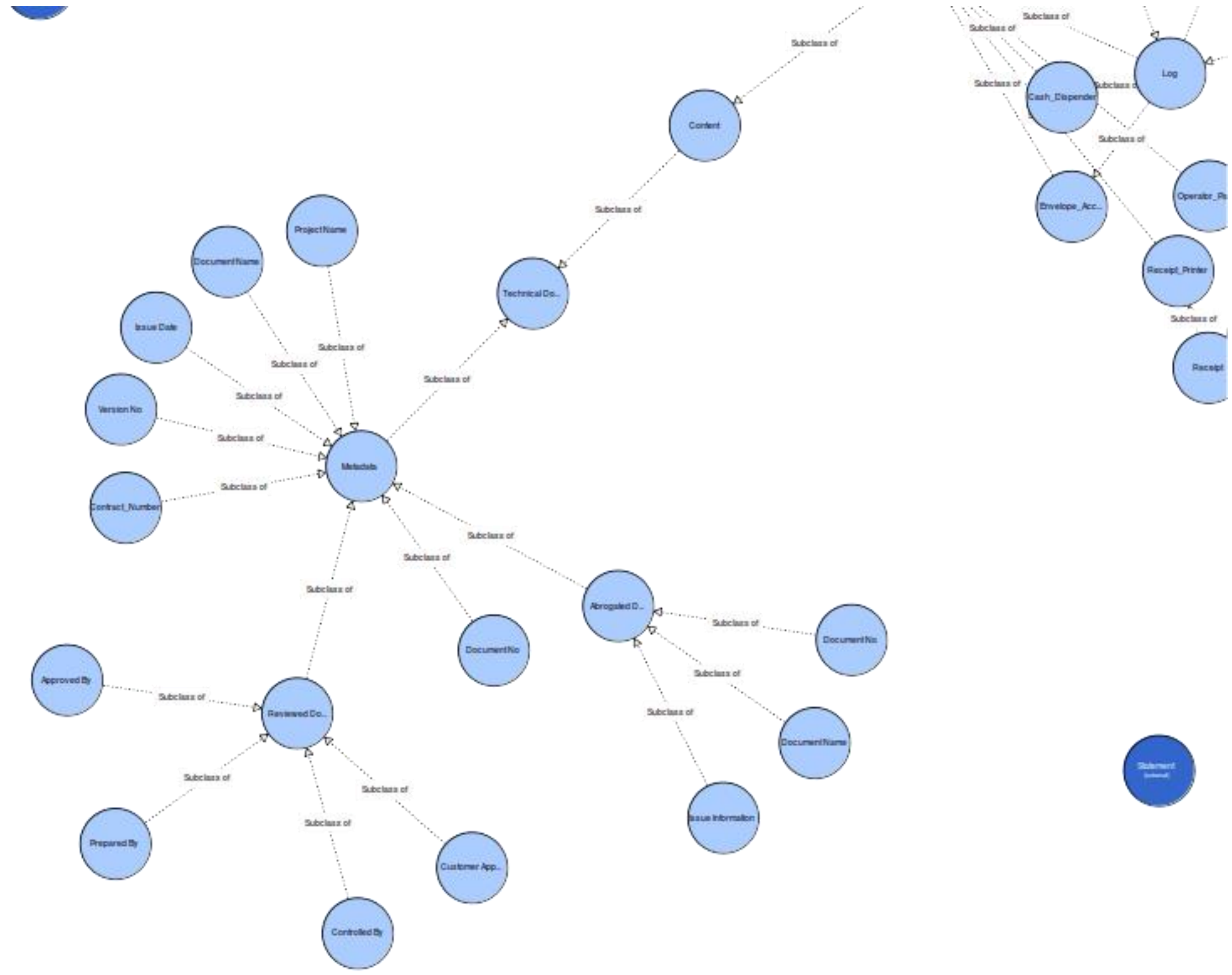
Ontology Issues

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)

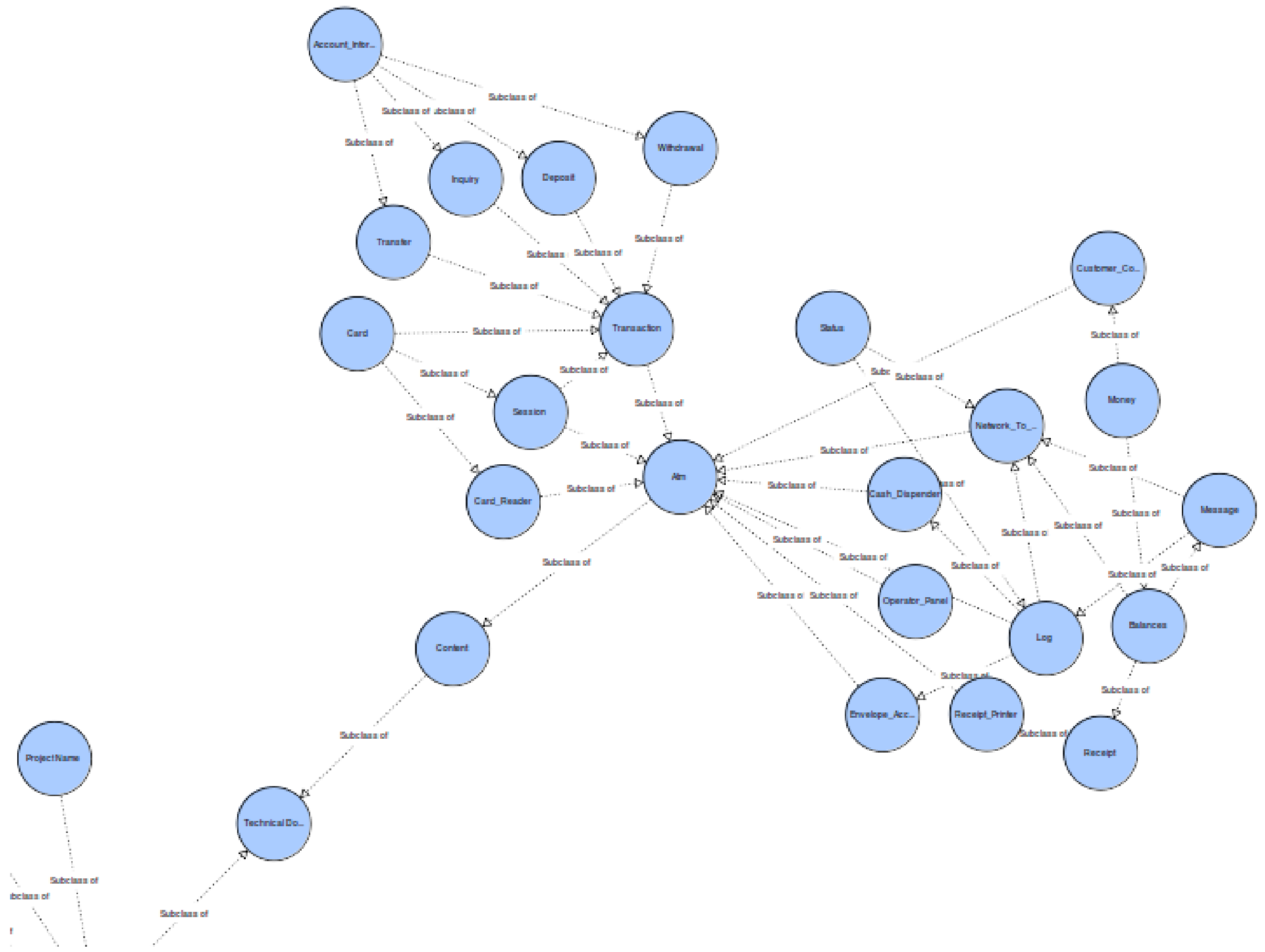
Ontology Services

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.

ITEA3